Computational Statistics
https://doi.org/10.1007/s00180-025-01637-y

ORIGINAL PAPER

# A simple and efficient method for sampling mixture models based on Dirichlet and Pitman-Yor processes

Mame Diarra Fall[1] · Éric Barat[2]

Received: 11 September 2024 / Accepted: 27 April 2025
© The Author(s) 2025, corrected publication 2025

**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 typically 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., see Müller and Quintana 2004; Müller and Mitra 2013; Müller et al. 2015; Ghosal and van der Vaart 2017). The Dirichlet process mixture (DPM) model (Ferguson 1983; Lo 1984) is by far the most popular BNP model. In recent years, models going beyond the

✉ Mame Diarra Fall
diarra.fall@univ-orleans.fr

Éric Barat
eric.barat@cea.fr

1 Institut Denis Poisson, UMR CNRS, Université d'Orléans, Orléans, France

2 Université Paris Saclay, CEA, List 91120, Palaiseau, France

DPM have been proposed in the literature (Favaro and Walker 2013; Nieto-Barajas et al. 2004; Favaro and Teh 2013; Griffin and Walker 2011).

Let $\mathbf{X} = (X_1, \dots, X_n)$ 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 $\sigma$-finite measure $\lambda$; the mixing distribution $H$ is a discrete random probability measure. In this paper, we will mainly focus on the following two cases: (1) $H \sim \mathrm{DP}(\alpha, G_0)$, i.e. $H$ is distributed according to the 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 $d = 0$. The PYM is an interesting alternative to the DPM, allowing greater flexibility in modelling.

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

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

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

$$H(\cdot) = \sum_{j=1}^{\infty} \tilde{w}_j \delta_{\boldsymbol{\theta}_j^*}(\cdot), \tag{3}$$

where $\delta_{\boldsymbol{\theta}^*}(\cdot)$ is the Dirac measure giving the mass 1 at $\boldsymbol{\theta}^*$; the weights are constructed according to the so-called GEM distribution as follows: $\tilde{w}_1 = v_1$, $\tilde{w}_j = v_j \prod_{i=1}^{j-1}(1 - v_i)$, 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 $H$, the PYM provides a flexible model for clustering items of various kinds in a hierarchical setting without explicitly specifying the number of components.

Since $H$ is discrete, some $\boldsymbol{\theta}_i$ will be identical. Observations with the same parameter $\boldsymbol{\theta}_i$ can be considered to belong to the same component. We can then introduce latent indicator variables that assign observations to the mixture components. Let $c_i$ be such that $c_i = j$ if $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j^*$ and $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*, \dots)$ denote the unique values among $(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n)$. The parameter $\boldsymbol{\theta}_i$ takes the value $\boldsymbol{\theta}_j^*$ with probability $\tilde{w}_j$. The model (2) can be expressed hierarchically as follows, using the stick-breaking representation:

$$\begin{aligned} X_i|c_i, \boldsymbol{\Theta}^* &\overset{ind}{\sim} p(x_i|\boldsymbol{\theta}^*_{c_i}) \quad i = 1, \dots, n \\ c_i|\tilde{\mathbf{w}} &\overset{iid}{\sim} \sum_{j=1}^{\infty} \tilde{w}_j \delta_j(\cdot) \\ \tilde{\mathbf{w}}|d, \alpha &\sim \mathrm{GEM}(d, \alpha) \\ \boldsymbol{\theta}^*_j|G_0 &\overset{iid}{\sim} G_0. \end{aligned} \tag{4}$$

In the following we will note $\mathbf{c} = (c_1, \dots, c_n)$ the indicator variables.

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 marginalised out, whereas in conditional methods, it is explicitly represented.

Marginal methods can be further divided into conjugate and non-conjugate models. Conjugacy means that the mixture kernel $p(\cdot|\boldsymbol{\theta})$ and the base distribution $G_0$ form a conjugate pair. In this case, the computation of conditional posterior distributions is simplified and can be performed analytically (Neal 1991; Escobar 1994; West et al. 1994; Escobar and West 1995; Bush and MacEachern 1996).

In non-conjugate models, however, posteriors can not be easily calculated. The sampling scheme is more difficult and requires sophisticated techniques (MacEachern and Müller 1998; Walker and Damien 1998; Green and Richardson 2001). The reader is referred to Neal (2000) for a more complete overview and discussion of these methods. Neal (2000) 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 temporary auxiliary components.

An alternative to marginal methods are conditional algorithms, which explicitly represent the mixing measure using, for example, its stick-breaking representation (Ishwaran and James 2001; Walker 2007). The challenge with conditional approaches is to deal with the countably infinite representation of $H$ in Eq. (3). In Ishwaran and James (2001), the authors resort to an approximation and truncate the mixing measure to a deterministic value. An alternative that avoids hard truncations was proposed by Muliere and Tardella (1998), 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 Arbel et al. (2019). The idea of random truncation, but avoiding the introduction of truncation errors as in the previously cited papers, was developed in Papaspiliopoulos and Roberts (2007) with a Metropolis-Hastings sampling scheme, and in Walker (2007) using the *slice sampling* strategy. The latter algorithm was improved by Papaspiliopoulos (2008) and Kalli et al. (2011).

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, as they must condition on all previously allocated data. These incremental updates mean that marginal samplers are not easily parallelisable. This is detrimental when working with large datasets, as the allocation step is the most time consuming part of the algorithm. Another drawback of marginalising over the mixing measure is that computing posterior conditionals requires additional sampling steps (Ishwaran and James (2001); Gelfand and Kottas (2002); Arbel et al. (2016)). 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 marginalisation, leading to a significant reduction in the dimensionality of the parameter 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 parallelise to take advantage of recent parallel computing hardware architectures, which is particularly useful for large data sets.

However, the simplicity of the representation comes at the cost of slower mixing. In fact, in this representation, the weights are explicitly defined by the prior and the components are represented with a size-biased order on their labels. This means that components with lower labels have higher prior probabilities than components with higher labels. As a result, components are not interchangeable and the cluster prior labels contribute to the posterior sampling. In this situation, the sampler must efficiently mix cluster labels to avoid clustering bias. The authors in Porteous et al. (2006) 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 cluster labels, this additional step seems to be the only way to improve the mixing over clusters (see also Papaspiliopoulos 2008). In contrast, marginal methods using the Pólya urn representation sample in the space of equivalence classes over exchangeable cluster labels, where cluster identities are arbitrary and insignificant. This is the appropriate space for the sampler, since cluster labels are irrelevant.

It is worth mentioning that there are hybrid samplers that cannot be classified as either marginal or conditional (Lomeli et al. 2015; Dubey et al. 2020). We also point out that some conditional samplers, instead of using the stick-breaking representation for the underlying mixing measure, exploit other constructive representations. For instance, the use of the so-called Ferguson and Klass representation (Ferguson and Klass 1972) of independent increment processes has been considered in the literature (see for example Griffin and Walker (2011) and Nieto-Barajas and Prünster (2009), 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 turn out 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 cluster labels, where cluster 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 Caron et al. (2014), Donnet et al. (2018), or they have inspired other authors as Canale et al. (2022).

The rest of the paper is structured as follows. In Sect. 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 Sect. 3. We conclude the paper in Sect. 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 the cluster labels as in marginal methods, thereby collapsing the model into a lower dimensional space, while preserving the 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 updating of Pitman-Yor processes under the class of Poisson-Kingman models introduced by Pitman (2003). The following proposition summarises this posterior characterisation.

**Proposition 1** *Pitman (1996b), Corollary 20 Let $H \sim \mathrm{PY}(d, \alpha, G_0)$ 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,*

$$H|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \overset{d}{=} \sum_{j=1}^{k_n} w_j \delta_{\boldsymbol{\theta}^*_j} + r_{k_n} H_{k_n}, \tag{5}$$

*where*

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

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

The posterior characterisation (5) allows us to work on the space of cluster equivalence classes and, due to exchangeability, to integrate out the order of cluster labels as in marginal samplers. Indeed, Pitman showed in Pitman (1996a) the equivalence between the exchangeability of the random partition generated by sampling from a discrete distribution and the symmetry of the law characterising the limiting frequencies of the occupied components given the data. We can easily check that exchangeability is ensured in Eq. (5), since it combines a Dirichlet distribution and an unconditional Pitman-Yor process (independent of the observed data). Being based on this representation, the samplers we propose in Sects. 2.1 and 2.2 live in the space of equivalence classes on cluster labels. These labels are therefore exchangeable and no label permutation is required. This property has important consequences for algorithm mixing, as we shall see in the comparative study. In contrast, in conditional algorithms using the stick-breaking representation, exchangeability is lost when using the usual updating rule:

$$H(\cdot)|\theta_1, \dots \theta_n = \sum_{k\in\mathbf{c}^*} \tilde{w}_k^* \delta_{\theta_k^*}(\cdot) + \sum_{k\notin\mathbf{c}^*} \tilde{w}_k \delta_{Z_k}(\cdot), \tag{6}$$

where $\mathbf{c}^* = (c_1^*, \dots, c_{k_n}^*)$ are the unique values of the classification variables $\mathbf{c} = (c_1, \dots, c_n)$. The weights $\tilde{w}_k^*$ are constructed as follows: $\tilde{w}_1^* = v_1^*, \tilde{w}_2^* = v_2^*(1 - v_1^*), \dots, \tilde{w}_n^* = v_n^* \prod_{i=1}^{n-1}(1 - v_i^*)$ 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 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 characterisation (5), we propose a Gibbs sampling scheme to sample from the posterior of a Pitman-Yor mixture model and, as a special case, the Dirichlet process mixture. To simulate $H_{k_n}$, the independent part of the posterior given in (5), we propose two variants of the sampler. The first makes use of a thresholded version of the "Slice efficient dependent" of Kalli et al. (2011). The second is based on a truncation of the process, as originally suggested in Ishwaran and James (2001).

### 2.1 Variant 1: exchangeable thresholded slice sampler

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

$$f(x_i, u_i) = \sum_{k=1}^{\infty} w_k\, p(x_i|\boldsymbol{\theta}_k^*)\mathcal{U}(u_i|0, \xi_k), \tag{7}$$

where $\xi_k$ is a variable such that for all $k$,

$$\xi_k = \min\left(w_k, \zeta\right), \tag{8}$$

with $\zeta \in [0,1]$ and is independent of $w_k$. Here, $\zeta$ is a threshold we propose to improve the mixing properties of the sampler. The threshold $\zeta$ can be a random or a 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. This is reminiscent of the augmentation introduced in Neal's Algorithm 8. A typical deterministic value of $\zeta$, which gives 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 assigned data. It can be expressed in the two-parameter case as

$$\zeta = \frac{(\alpha + d\,\mathbb{E}_{\alpha,d}(K_n))(1-d)}{(\alpha+n)(\alpha+1)}, \tag{9}$$

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

$$\mathbb{E}_{\alpha}(K_n) = \sum_{i=1}^{n} \frac{\alpha}{\alpha+i-1} \approx \alpha \log\left(1+\frac{n}{\alpha}\right). \tag{10}$$

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,

$$\mathbb{E}_{\alpha,d}(K_n) \approx \frac{\Gamma(\alpha+1)}{d\,\Gamma(\alpha+d)} n^d. \tag{11}$$

Coming back to the slice sampling formulation, using (8), we can rewrite (7) as follows,

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

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 characterisation (5), 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 Kalli et al. (2011), we jointly sample $\mathbf{w}, \mathbf{u}|\mathbf{c}$. The full conditional distributions involved in the sampling 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 sample 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 $k \leq k_n$:

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

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

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

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

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

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

   Set $k^* = \min\big(\big\{k : r_k < u^*\big\}\big)$. 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 applies only 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 the classification variables requires the computation of a normalising constant, which is feasible using auxiliary variables, since the choice of $c_i$ is from a finite set,

   $$c_i|\mathbf{w}, \mathbf{u}, \boldsymbol{\Theta}^*, \mathbf{x} \overset{\text{ind}}{\sim} \sum_{k=1}^{k^*} w_{k,i}\, \delta_k(\cdot), \tag{12}$$

   where $w_{k,i} \propto \mathbf{1}\big(w_k > u_i\big) \max\big(w_k, \zeta\big) p\big(x_i|\boldsymbol{\theta}_k^*\big)$, and $\sum_{k=1}^{k^*} w_{k,i} = 1$. Note also that, to speed up the 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,

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

   - Sampling parameters for unallocated components from their priors,

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

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 (5). 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 (5) is approximated by

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

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 $k^* = k_n + M$. Note that unlike Ishwaran and James (2001), here the truncation is not at the model specification stage, but at the sampling stage. 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

$$w_{k,i} \propto w_k\, p\big(x_i|\boldsymbol{\theta}_k^*\big) \quad \text{and} \quad \sum_{k=1}^{k^*} w_{k,i} = 1.$$

- Sample $w_k$ for $k \leq k_n$:

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

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

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

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

- Sample components parameters using

$\star$ the density proportional to

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

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

$\star$ 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 Algorithm comparisons

In this section, we carry out a comparative study on different data sets, both real and simulated. We will compare the two variants of the proposed conditional sampler (called "Slice exch. thres." and "Trunc. exch."), with two other conditional samplers: the efficient Slice sampler proposed by Kalli et al. (2011) ("Slice efficient"), and the truncated blocked Gibbs sampler of Ishwaran and James (2001) ("Truncated"). Note that "Slice efficient" is referred to as "Slice efficient dependent" in Kalli et al. (2011), as opposed to its independent version which uses a deterministic slice function.

We will also compare ourselves to Neal's Algorithm 8 ("Algo. 8", Neal 2000), known as the best algorithm for marginal methods. Since the latter is a marginal algorithm, it is not included in the comparison of conditional methods discussed here. Its results are therefore provided for information, to give an idea of the performance to be expected.

### 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 $G_0$ is a normal-gamma base measure i.e, $G_0(\mu, \sigma^{-2}) = \mathcal{N}(\mu|\eta, \kappa^{-1}) \times \mathcal{G}(\sigma^{-2}|\gamma, \beta)$ where $\mathcal{G}(\cdot|\gamma, \beta)$ denotes the Gamma distribution with mean $\gamma\beta$. For comparison purposes, we use the same real and synthetic data sets as in Kalli et al. (2011).

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

   The corresponding densities are shown in Fig. 1. To measure the performance of the algorithms on small and large data sets, we generated $n = 100$, $n = 1000$ and $n = 10,000$ draws from each of these two mixtures.
2. The real data are
   - Galaxy data: these are the radial velocities (in $10^3$ km/s) of 82 distant galaxies diverging from our own. This is a popular data set in density estimation problems, and was first described in Roeder (1990).
   - S&P 500: this consists of 2023 daily index returns from 1980 to 1987. This is another dataset commonly used in density estimation and volatility studies of financial assets (Jacquier et al. (2004)). The data set is unimodal, asymmetric, and heavy-tailed.

### 3.2 Algorithm performance

We monitored the convergence of two quantities: the deviance of the estimated density and the number of occupied clusters. The deviance is defined as

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

where $n_j$ is the size of cluster $j$. Deviance is a useful function of several model parameters that avoids simulating additional weights in the usual slice sampling and marginal approaches, since the weights are replaced by empirical cluster sizes.

The performance of the competing samplers in their stationary regime was assessed by looking at the integrated autocorrelation time (IAT) for each monitored quantity. IAT is defined in Sokal (1997) as,

$$\tau = 1 + 2\sum_{\ell=1}^{\infty} \rho_\ell,$$

where $\rho_\ell$ is the sample autocorrelation at lag $\ell$. 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., Neal 2000; Green and Richardson 2001; Papaspiliopoulos and Roberts 2007; Kalli et al. 2011). 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 $2\tau$ greater than in independent sampling (Sokal 1997). If we denote by $\tau_j$ the integrated autocorrelation time produced by algorithm $j$ for a given quantity, then $\tau_1/\tau_2 = k > 1$ means that algorithm 1 requires $k$ times more iterations than algorithm 2 to produce the same Monte Carlo error (Papaspiliopoulos and Roberts 2007). Thus, when comparing two alternative Monte Carlo algorithms for the same problem, the most efficient one is the one that produces the smallest IAT because it provides better estimates.

However, calculating IAT is difficult in practice. Following Sokal (1997), an estimator of $\tau$ can be obtained by summing the estimated autocorrelations up to a fixed lag $L$,

$$\hat{\tau} = 1 + 2\sum_{\ell=1}^{L} \hat{\rho}_\ell. \tag{13}$$

One can also estimate the standard error of $\hat{\tau}$ using the following formula from Sokal (1997),

$$\operatorname{std}(\hat{\tau}) \approx \sqrt{\frac{2(2L+1)}{N}\tau^2} \tag{14}$$

where $N$ is the Monte Carlo size, which must be sufficiently large. A common value of $L$ is

$$L = \min\{\ell : |\hat{\rho}_\ell| < 2/\sqrt{N}\}. \tag{15}$$

If the sampler is run for a sufficient number of iterations, this ensures that $L$ is the smallest lag for which we would not reject the null hypothesis $H_0 : \rho_\ell = 0$.

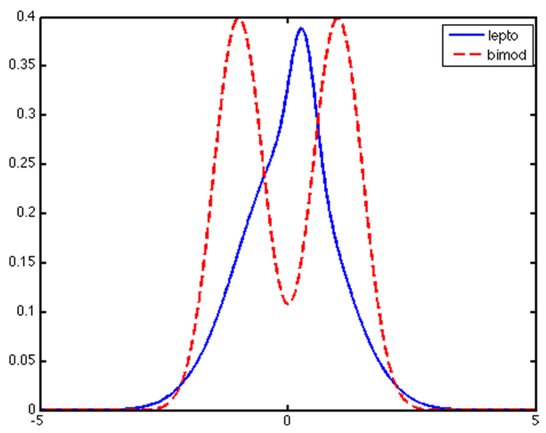

**Fig. 1** Bimodal (bimod) and unimodal leptokurtic (lepto) mixtures

## 3.3 Algorithm 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 dataset length and each parameterisation are given in Tables 1-2.

Second, we investigated the behaviour of competing algorithms in a PYM power-law case. In our experiments, we considered several 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 number of clusters. Note that for $n \leq 1000$, the prior expected number of clusters in the PYP is between those of the two DPs, and for $n > 1000$ this expectation is higher than that of the DP($\alpha = 5$). This is consistent with the Eqs. 10 and 11, and with Fig. 2 which shows realisations of $K_n$, for $n$ ranging from 1 to 10, 000 for the three processes.

The hyperparameters have been fixed in a data-driven way according to Green and Richardson (2001) and are set as follows: if $R$ is the data range, we take $\eta = R/2$ (mid-range), $\kappa = 1/R^2$, $\gamma = 2$ and $\beta = 0.02R^2$.

The blocked Gibbs sampler of Ishwaran and James (2001) ("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 $M = 2\alpha\log(n)$. Remember that in our sampler, this truncation only affects $H_{k_n}$ (see the right-hand side of Eq. (5)). This ensures that all the occupied components are represented and that the number of unoccupied components is large enough. Therefore, we do not worry about the missing components as their mass is small.

Algorithm 8 of Neal (2000) was tested with $m = 2$ auxiliary components. We followed Sokal's instructions (Sokal 1997), which recommends running samplers for a sufficient number of iterations. For each data set, we ran $2 \times 10^6$ iterations for each algorithm, discarding the first $2 \times 10^5$ for the burn-in period. It is worth noting that we chose a very high number of iterations, as recommended by Sokal, to ensure that we obtain reliable IAT estimates for all competing algorithms and to provide a very fair comparison. Samplers do not require as many iterations to reach the stationary state.

## 3.4 Results and comments

The following tables report the performance in terms of IAT and mean time per iteration (in milliseconds) achieved by

- the two variants of the proposed algorithm ("Slice exch. thres." and "Trunc. exch." )
- the truncated blocked Gibbs sampler of Ishwaran and James (2001) ("Truncated").
- the efficient slice sampler of Kalli et al. (2011) ("Slice efficient").

IAT is calculated using the formula (13), where $L$ is denoted as $L_D$ for the IAT on the deviance and $L_C$ for the IAT on the number of clusters. The estimated standard deviation (value in brackets in the tables) is calculated using (14). We recall that the smaller the IAT, the better the algorithm. The best performance is shown in bold and the second best in underlined (the ranking is based on conditional algorithms). We also report the mean number of clusters and the deviance estimated by each algorithm. This ensures that the algorithms perform the estimation correctly, and that they can be assessed through their mixing performance and computation time. As mentioned above, "Algo 8" is a marginal method and is not included in the comparison. Its performance is given for information.

DPM case We report here the results for the DPM case with $\alpha = 1$ (results with $\alpha = 5$ are provided in Appendix B).

We first look at the results on real data (Galaxy and S&P 500). The results are presented in Tables 4, 5. These tables highlight that the "Slice efficient" sampler gives the worst performance in terms of mixing, although it offers reduced computation times. The "Truncated" sampler offers better mixing than the "Slice efficient", but higher computation times. The two variants of the proposed sampling method

**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 ($n = 1000$) | 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 ($n = 1000$) | 26.5 |
| S&P 500 ($n = 2023$) | 30 |
| Lepto/bimod ($n = 10,000$) | 38 |

**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 ($n = 1000$) | 25.5 |
| S&P 500 ($n = 2023$) | 36.4 |
| Lepto/bimod ($n = 10,000$) | 58.9 |

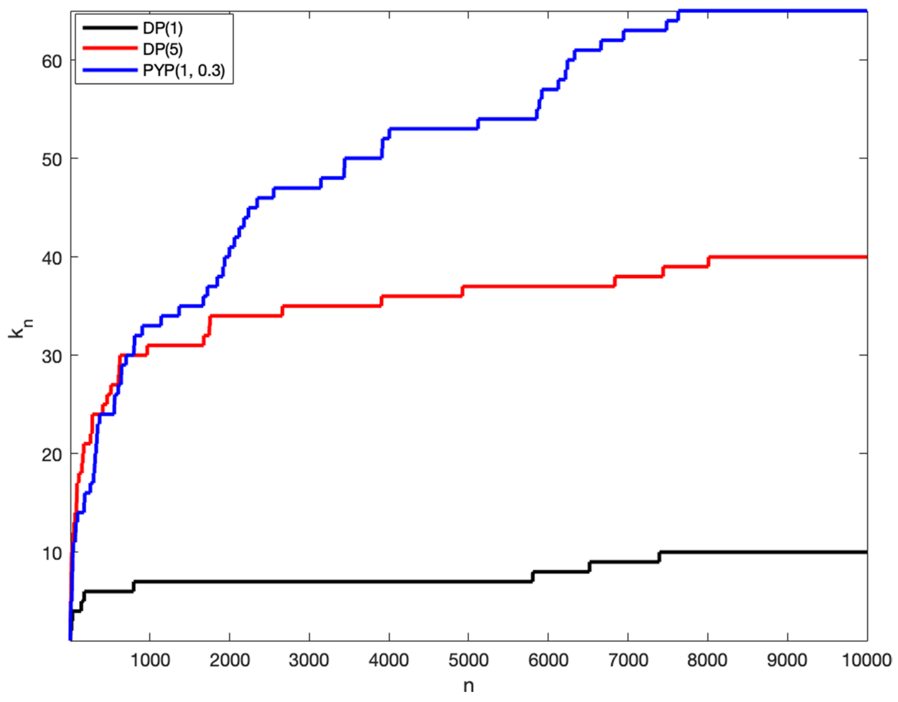

**Fig. 2** Number of clusters as a function of the sample size for the three processes: DP($\alpha = 1$), DP($\alpha = 5$), and PYP($\alpha = 1, d = 0.3$)

give the best results in terms of mixing, while being competitive in terms of computation time.

Secondly, we examine the results on simulated data (lepto and bimod with $n = 1000$ and $n = 10,000$). The results are given in Tables 6, 7, 8, 9. Simulation results for $n = 100$ are provided in Appendix A. This appendix also contains autocorrelation curves on the Galaxy data, as well as the data histogram and estimated posterior density for each competing algorithm.

These results, together with those presented in Appendix A (Tables 15, 16), confirm the observations made on the Galaxy and S&P 500 datasets: "Slice efficient" gives low computation times but too high IATs, and the opposite is true for "Truncated". The two versions we propose offer the best compromise between IAT and computation time. These conclusions remain valid for the DPM($\alpha = 5$) case (results presented in Appendix B).

PYM case We now examine the results obtained in the PYM case with $d = 0.3$ and $\alpha = 1$, only for Galaxy and lepto data with $n = 1000$. The results are given in Tables 10, 11. The rest of the results are detailed in Appendix C.

These results and those in Appendix C show that both proposed versions give the best IATs, with the " Trunc. exch." version standing out by having by far the lowest computation time. In contrast to the DPM case, here it is "Truncated" (ranked second in terms of time) that delivers lower computation times than "Slice efficient", while remaining third in terms of IAT. These results can be explained by the fact that in slice sampling based algorithms, the number of components at each iteration is random and unbounded. In the case of PYM, this number can be very large. Conversely, in truncated samplers, the number of components at each iteration is bounded.

In order to deepen the analysis of the proposed samplers, we now examine 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 (see Eqs. 8–9). For this reason, we have implemented our slice sampler using the exchangeable model but without the proposed threshold ("SE without thres."), and the "Slice efficient" of Kalli et al. (2011), which uses a non-exchangeable model with the introduction of our threshold ("Slice eff. thres."). In this case, the full conditional for allocation variables is given by (12).

Tables 12 and 13 show the results on the real data (Galaxy and S&P 500) in the DPM case with $\alpha = 1$, and Table 14 displays the results on Galaxy in the PYM ($\alpha = 1, d = 0.3$). 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.

We now analyse the results for DPM($\alpha = 1$) and PYM($\alpha = 1, d = 0.3$) separately.

- ***DPM*** First, as far as the threshold is concerned, its inclusion in "Slice efficient" significantly reduces the IAT, as can be seen by comparing the original "Slice efficient" and its thresholded version "Slice eff. thres.". This also increases the computation time per iteration, but this cost is small enough to be tolerable. The performance of "Slice eff. thres." is close to that of "Truncated" in terms of mixing, but at a lower time cost. We believe that the poor mixing due to the non-exchangeability in the stick-breaking posterior representation is exacerbated by the absence of weights in "Slice efficient". This could often hinder the Gibbs sampler in the allocation step, since the assignment of an observation to a component depends only on the likelihood. In contrast, the "Truncated" algorithm takes into account the weights of the mixture components when updating the classification variables. The introduction of the proposed threshold in "Slice efficient" encourages movement into less populated components and allows the mixing performance of this algorithm to be improved. On the other hand, the removal of the threshold in our sampler increases the IAT. This is noticeable in the differences between "Slice exch. thres" and "SE. without thres.". Overall, we have observed that for all datasets, the introduction of our threshold leads to a faster decrease of the autocorrelation curves in the early lags, and thus to a lower IAT, as well as to a slight increase of the computation time per iteration. We now look at the benefits we get from the exchangeability property of the model. This is reflected, for example, in the differences between "Slice exch. thres."

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms) ↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | 14.48(0.37) | **2.88**(0.05) | 3.986(0.93) | 1561.14(21.61) | 2.74 |
| *Trunc. exch. (ours)* | **14.42**(0.37) | 2.94(0.05) | 3.989(0.93) | 1561.16(21.69) | **1.57** |
| *Truncated* | 38.65(1.00) | 3.63(0.07) | 3.996(0.94) | 1561.15(21.66) | 4.33 |
| *Slice efficient* | 60.65(1.57) | 5.28(0.10) | 3.991(0.93) | 1561.15(21.62) | 2.27 |
| *Algo 8* ($m = 2$) | 8.25(0.21) | 2.57(0.05) | 3.987(0.93) | 1561.16(21.62) | 1.58 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms) ↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | <u>22.58</u>(0.75) | **145.07**(3.06) | 4.977(0.82) | 14990.47(57.56) | <u>7.62</u> |
| *Trunc. exch. (ours)* | **21.59**(0.72) | <u>148.66</u>(3.14) | 4.978(0.82) | 14990.50(59.09) | 11.9 |
| *Truncated* | 32.75(1.09) | 148.69(3.14) | 4.975(0.81) | 14990.94(59.44) | 19.5 |
| *Slice efficient* | 105.92(3.53) | 204.63(4.32) | 4.965(0.81) | 14991.21(61.14) | **5.84** |
| *Algo 8* ($m = 2$) | 13.55(0.45) | 106.28(2.24) | 4.980(0.82) | 14990.38(59.09) | 9.07 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms) ↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | 93.28(3.93) | **3.65**(0.07) | 3.806(1.73) | 2735.14(8.66) | 4.08 |
| *Trunc. exch. (ours)* | **91.20**(3.85) | 3.69(0.07) | 3.795(1.72) | 2735.14(8.66) | 3.08 |
| *Truncated* | 156.27(6.60) | 3.71(0.07) | 3.777(1.71) | 2735.13(8.62) | 8.25 |
| *Slice efficient* | 257.25(10.85) | 5.13(0.09) | 3.766(1.68) | 2735.12(8.61) | **2.70** |
| *Algo 8* ($m = 2$) | 47.25(1.99) | 3.06(0.06) | 3.798(1.72) | 2735.14(8.65) | 2.84 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms) ↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **235.49**(9.93) | 13.17(0.24) | 4.006(2.05) | 2400.51(18.68) | 4.15 |
| *Trunc. exch. (ours)* | 237.70(10.02) | 13.66(0.25) | 4.022(2.08) | 2400.48(18.72) | 3.17 |
| *Truncated* | 294.24(12.41) | **12.67**(0.23) | 3.958(2.01) | 2400.47(18.49) | 8.22 |
| *Slice efficient* | 472.95(19.95) | 16.91(0.31) | 3.864(1.92) | 2400.45(18.26) | **2.8** |
| *Algo* 8 ($m = 2$) | 148.81(6.28) | 11.55(0.21) | 4.018(2.07) | 2400.48(18.69) | 2.90 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms) ↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **244.64**(11.53) | 5.79(0.12) | 3.77(1.74) | 27235.46(9.09) | <u>9.10</u> |
| *Trunc. exch. (ours)* | <u>247.02</u>(11.65) | <u>5.57</u>(0.12) | 3.77(1.73) | 27235.45(9.04) | 21.50 |
| *Truncated* | 286.67(13.52) | **5.46**(0.11) | 3.73(1.74) | 27235.44(8.98) | 27.80 |
| *Slice efficient* | 456.95(21.55) | 12.20(0.26) | 3.76(1.76) | 27235.44(9.00) | **6.99** |
| *Algo 8* $(m = 2)$ | 180.58(8.51) | 4.76(0.10) | 3.78(1.76) | 27235.46(9.05) | 15.6 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | 212.65(10.03) | 11.74(0.25) | 3.74(1.77) | 23517.95(12.52) | 8.51 |
| *Trunc. exch. (ours)* | **179.46**(8.46) | **10.96**(0.23) | 3.74(1.74) | 23517.94(12.43) | 21.7 |
| *Truncated* | 186.83(8.81) | 17.80(0.38) | 4.73(1.75) | 23518.57(13.92) | 25.6 |
| *Slice efficient* | 444.243(20.95) | 17.60(0.37) | 3.68(1.70) | 23517.90(12.35) | **6.39** |
| *Algo* 8 ($m = 2$) | 142.67(6.73) | 10.47(0.22) | 3.74(1.77) | 23517.95(12.47) | 15.20 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | <u>10.56</u>(0.27) | <u>2.84</u>(0.05) | 4.867(2.13) | 1561.67(21.84) | 9.11 |
| *Trunc. exch. (ours)* | **9.81**(0.25) | **2.79**(0.05) | 4.716(1.77) | 1561.61(21.93) | **1.38** |
| *Truncated* | 29.20(0.75) | 3.65(0.07) | 4.932(1.97) | 1561.73(21.94) | <u>2.99</u> |
| *Slice efficient* | 44.65(1.15) | 5.43(0.10) | 4.872(2.13) | 1561.66(21.82) | 15.8 |
| *Algo 8* ($m = 2$) | 5.79(0.15) | 2.37(0.04) | 4.869(2.13) | 1561.66(21.89) | 1.40 |

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

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **156.21**(7.37) | <u>13.56</u>(0.29) | 4.255(3.22) | 2371.35(16.11) | 16 |
| *Trunc. exch. (ours)* | <u>167.13</u>(7.88) | **13.08**(0.28) | 4.247(3.13) | 2371.34(16.03) | **3.20** |
| *Truncated* | 270.51(12.75) | 17.04(0.36) | 4.387(3.62) | 2371.52(16.49) | <u>8.07</u> |
| *Slice efficient* | 422.09(19.90) | 20.47(0.43) | 4.291(3.30) | 2371.50(16.53) | 14.7 |
| *Algo 8* ($m = 2$) | 96.90(4.57) | 10.90(0.23) | 4.242(3.22) | 2371.34(15.98) | 2.92 |

**Table 12** Galaxy data (DPM)

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres* | 14.48(0.37) | 2.88(0.05) | 3.986(0.93) | 1561.14(21.61) | 2.74 |
| *Trunc. exch.* | 14.42(0.37) | 2.94(0.05) | 3.989(0.93) | 1561.16(21.69) | 1.57 |
| *SE without thres.* | 35.52(0.92) | 4.77(0.09) | 3.989(0.93) | 1561.15(21.61) | 1.94 |
| *Truncated* | 38.65(1.00) | 3.63(0.07) | 3.996(0.94) | 1561.15(21.66) | 4.33 |
| *Slice efficient* | 60.65(1.57) | 5.28(0.10) | 3.991(0.93) | 1561.15(21.62) | 2.27 |
| *Slice eff. thres.* | 37.82(0.98) | 3.61(0.07) | 3.986(0.93) | 1561.08(22.17) | 3.03 |
| *Algo 8* ($m = 2$) | 8.25(0.21) | 2.57(0.05) | 3.987(0.93) | 1561.16(21.62) | 1.58 |

**Table 13** S&P 500 (DPM)

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres* | 22.58(0.75) | 145.07(3.06) | 4.977(0.82) | 14990.47(57.56) | 7.62 |
| *Trunc. exch.* | 21.59(0.72) | 148.69(3.14) | 4.978(0.82) | 14990.50(59.09) | 11.9 |
| *SE without thres.* | 93.93(3.13) | 194.26(4.10) | 4.976(0.82) | 14990.35(57.80) | 5.38 |
| *Truncated* | 32.75(1.09) | 148.66(3.14) | 4.975(0.81) | 14990.94(59.44) | 19.5 |
| *Slice efficient* | 105.92(3.53) | 204.63(4.32) | 4.965(0.81) | 14991.21(61.14) | 5.84 |
| *Slice eff. thres.* | 34.87(1.16) | 145.46(3.07) | 4.969(0.82) | 14990.56(60.53) | 8.37 |
| *Algo 8* ($m = 2$) | 13.55(0.45) | 106.28(2.24) | 4.980(0.82) | 14990.38(59.09) | 9.07 |

and "Slice eff. thres.", and between "Trunc. exch." and "Truncated". We observed that the autocorrelation curves obtained with exchangeable samplers decrease and tend to zero faster than those of algorithms using non-exchangeable models ("Truncated" vs. "Trunc. exch."., "Slice eff. thres." vs. "Slice exch. thres.", "Slice efficient" vs. "SE without thres"). This behaviour was observed for all datasets. In summary, the exchangeability in the proposed samplers allows to drastically reduce the IAT, while keeping the computation time similar to "Slice efficient" and lower than "Truncated".

- ***PYM*** In the case of the Pitman-Yor model, the inclusion of our threshold in "Slice efficient" significantly reduces the IAT in "Slice eff. thres.", but also increases the computation time per iteration. As for exchangeability, the comparison between "Slice exch. thres." and "Slice eff. thres." leads to the conclusion that it significantly reduces IAT and computation time. However, the non-slice algorithms ("Trunc. exch." and "Truncated") perform much better in terms of time. As already explained, this is due to the fact that Slice-based samplers have to deal with an unbounded number of components at each iteration. In this case,

**Table 14** Galaxy data (PYM)

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres* | 10.56 (0.27) | 2.84 (0.05) | 4.867 (2.13) | 1561.67 (21.83) | 9.11 |
| *Trunc. exch.* | 9.81 (0.25) | 2.79 (0.05) | 4.716 (1.77) | 1561.61(21.93) | 1.38 |
| *SE without thres.* | 27.2 (0.70) | 4.57 (0.08) | 4.868 (2.13) | 1561.66 (21.83) | 5.36 |
| *Truncated* | 29.2 (0.75) | 3.65 (0.07) | 4.932 (1.97) | 1561.73 (21.94) | 2.99 |
| *Slice efficient* | 44.65 (1.15) | 5.43 (0.10) | 4.872 (2.13) | 1561.66 (21.82) | 15.8 |
| *Slice eff. thres.* | 24.95 (0.64) | 3.72 (0.07) | 4.858 (2.11) | 1561.79 (23.21) | 33.3 |
| *Algo 8* ($m = 2$) | 5.79 (0.15) | 2.37 (0.04) | 4.869 (2.13) | 1561.66 (21.89) | 1.40 |

our proposed “Trunc. exch.” remains the best choice, offering low IATs and very low computation times.

Finally, it is important to point out that the two proposed variants and “Algo 8” (Neal 2000) 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 (“Truncated” of Ishwaran and James 2001 and “Slice efficient” of Kalli et al. 2011) did not always give the same results. We also observed erratic convergence behaviour of the Gibbs sampler in these two algorithms, especially for large datasets (e.g., lepto with $n = 10,000$).

## 4 Conclusion and discussion

As models become more 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 that satisfies the constraint of efficient parallelisation capability, in particular the allocation step since there is no marginalisation of the weights, while maintaining good mixing properties. This property is of great importance when dealing with large data sets. We refer the interested reader to Fall et al. (2024), for an application of the proposed sampler to an inverse problem in image reconstruction, which deals with data of the order of tens of millions and therefore requires a very efficient and parallelisable sampling algorithm.

Our comparative study on real and simulated data sets confirms 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. One 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. The proposed exchangeable thresholded slice version ("Slice exch. thres.") achieves adaptive truncation at each iteration, and maintains a nice trade-off between IAT and time cost in DPM models. For PYM models, "Trunc. exch." offers the best compromise.

Our samplers have been developed for Pitman-Yor and Dirichlet process mixture models. Since the "Slice efficient" of Kalli et al. (2011) was 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 may also be to introduce mixing moves over cluster labels, as suggested by Porteous et al. (2006) and Papaspiliopoulos and Roberts (2007). As mentioned above, the order of cluster labels matters in the stick-breaking representation. A label permutation step could lead to 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 $n = 100$ (Tables 15, 16). We also present some figures relating to the autocorrelation curves and densities estimated by competing algorithms.

**Table 15** Bimod data $n = 100$, $L_D = 150$, $L_C = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | 28.76(0.74) | **5.85**(0.11) | 3.801(1.66) | 287.59(8.46) | 2.40 |
| *Trunc. exch. (ours)* | **28.51**(0.74) | 6.00(0.11) | 3.808(1.67) | 287.58(8.48) | **1.53** |
| *Truncated* | 54.38(1.40) | 5.89(0.11) | 3.789(1.66) | 287.58(8.42) | 4.47 |
| *Slice efficient* | 99.92(2.58) | 8.76(0.16) | 3.784(1.65) | 287.58(8.42) | 2.18 |
| *Algo 8* ($m = 2$) | 15.59(0.40) | 5.20(0.09) | 3.794(1.66) | 287.59(8.52) | 1.56 |

**Table 16** Lepto data $n = 100$, $L_D = 200$, $L_C = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | <u>25.61</u>(0.85) | <u>13.53</u>(0.29) | 3.991(1.64) | 239.74(11.38) | 2.67 |
| *Trunc. exch. (ours)* | **24.78**(0.83) | **13.46**(0.28) | 3.983(1.63) | 239.75(11.31) | **1.21** |
| *Truncated* | 41.22(1.37) | 17.03(0.36) | 4.001(1.64) | 239.72(11.31) | 4.07 |
| *Slice efficient* | 120.71(4.03) | 46.28(0.98) | 3.979(1.64) | 239.77(11.29) | <u>2.27</u> |
| *Algo 8* ($m = 2$) | 14.79(0.49) | 9.83(0.28) | 3.994(1.63) | 239.72(11.36) | 1.26 |

Figures 3 and 4 show, for the Galaxy data, the autocorrelation curves used to estimate the IAT on the number of clusters and on the deviance, respectively. Figure 5 displays the data histogram and the estimated posterior density for each algorithm.

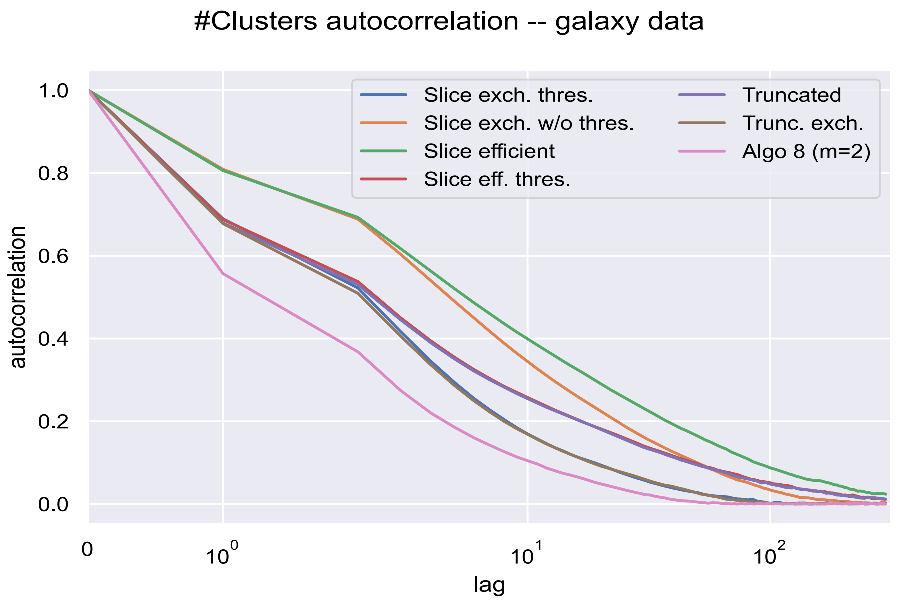

**Fig. 3** Autocorrelation curves used to estimate the IAT for the number of clusters (Galaxy data), using a logarithmic scale on the horizontal axis

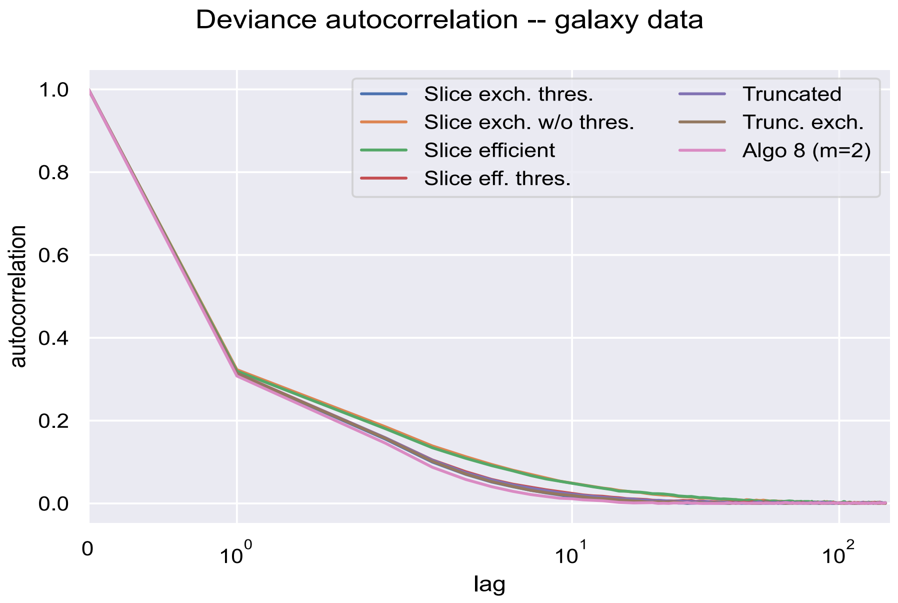

**Fig. 4** Autocorrelation curves used to estimate the IAT for the deviance (Galaxy data), using a logarithmic scale on the horizontal axis

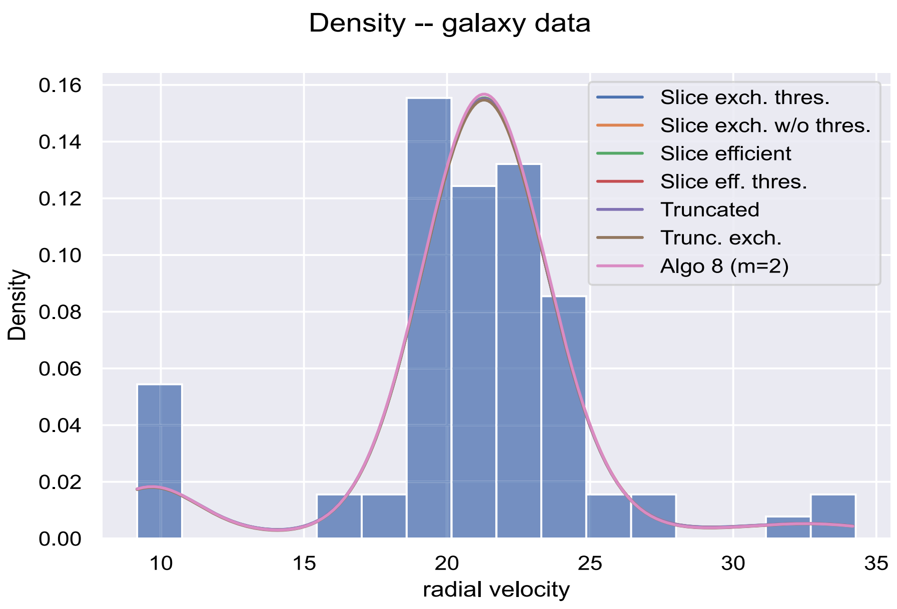

**Fig. 5** Histogram of data and estimated densities (Galaxy data)

## 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 (Tables 17, 18, 19, 20, 21, 22).

**Table 17** Galaxy data $n = 82$, $L_D = 150$, $L_C = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | 10.73(0.28) | **2.80**(0.05) | 7.082(3.32) | 1563.10(23.55) | 8.20 |
| *Trunc. exch. (ours)* | **10.11**(0.26) | 2.81(0.05) | 7.084(3.31) | 1563.10(23.54) | **1.83** |
| *Truncated* | 19.51(0.50) | 3.45(0.06) | 7.079(3.32) | 1563.10(23.57) | 13.2 |
| *Slice efficient* | 38.75(1.00) | 4.96(0.09) | 7.085(3.31) | 1563.11(23.59) | 6.29 |
| *Algo 8* ($m = 2$) | 6.16(0.16) | 2.35(0.04) | 7.084(3.31) | 1563.10(23.56) | 1.70 |

**Table 18** Bimod data $n = 100$, $L_D = 150$, $L_C = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | 14.24(0.37) | 2.35(0.04) | 8.886(5.41) | 283.18(8.98) | 7.70 |
| *Trunc. exch. (ours)* | **13.74**(0.35) | **2.33**(0.04) | 8.880(5.38) | 283.17(8.97) | **2.67** |
| *Truncated* | 23.22(0.60) | 2.67(0.05) | 8.883(5.41) | 283.18(9.00) | 17.0 |
| *Slice efficient* | 45.67(1.18) | 3.58(0.06) | 8.891(5.38) | 283.18(8.97) | 7.85 |
| *Algo 8* ($m = 2$) | 8.56(0.22) | 2.01(0.04) | 8.888(5.42) | 283.18(8.98) | 2.59 |

**Table 19** Bimod data $n = 1000$, $L_D = 150$, $L_C = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | $\underline{81.36}$(3.43) | $\underline{6.90}$(0.13) | 9.889(7.26) | 2741.07(12.05) | 11.7 |
| *Trunc. exch. (ours)* | **81.07**(3.42) | **6.81**(0.12) | 9.956(7.35) | 2741.08(12.05) | **9.51** |
| *Truncated* | 135.98(5.73) | 7.35(0.13) | 9.958(7.34) | 2741.08(12.09) | 37.9 |
| *Slice efficient* | 256.71(10.83) | 12.85(0.23) | 9.962(7.40) | 2741.09(12.06) | $\underline{9.63}$ |
| *Algo 8* ($m = 2$) | 42.85(1.81) | 5.35(0.10) | 9.928(7.36) | 2741.08(12.03) | 5.03 |

**Table 20** Lepto data $n = 100$, $L_D = 200$, $L_C = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **11.12**(0.37) | **14.26**(0.30) | 9.004(4.74) | 257.17(21.70) | <u>6.56</u> |
| *Trunc. exch. (ours)* | <u>11.84</u>(0.39) | <u>14.34</u>(0.30) | 8.988(4.75) | 257.19(21.89) | **3.42** |
| *Truncated* | 17.79(0.59) | 15.96(0.34) | 9.011(4.75) | 257.16(21.69) | 20.9 |
| *Slice efficient* | 37.12(1.24) | 33.49(0.71) | 9.009(4.74) | 257.18(21.67) | 6.92 |
| *Algo 8* ($m = 2$) | 6.95(0.23) | 11.43(0.24) | 8.999(4.73) | 257.18(21.66) | 3.16 |

**Table 21** Lepto data $n = 1000$, $L_D = 150$, $L_C = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | $\underline{90.94}$(3.84) | $\underline{15.81}$(0.29) | 11.121(7.69) | 2354.96(19.51) | 14.3 |
| *Trunc. exch. (ours)* | **90.68**(3.82) | **15.72**(0.29) | 11.127(7.64) | 2354.95(19.59) | **9.85** |
| *Truncated* | 145.95(6.16) | 17.40(0.32) | 11.080(7.70) | 2354.98(19.60) | 38.2 |
| *Slice efficient* | 254.45(10.73) | 27.28(0.50) | 11.196(7.65) | 2354.90(19.60) | $\underline{11.5}$ |
| *Algo 8*$(m = 2)$ | 50.75(2.14) | 13.13(0.24) | 11.098(7.69) | 2354.94(19.48) | 5.81 |

**Table 22** S&P 500 data $n = 2023$, $L_D = 300$, $L_C = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **17.09**(0.57) | 151.34(3.91) | 7.476(2.65) | 14989.81(53.22) | 21.7 |
| *Trunc. exch. (ours)* | 18.02(0.60) | **143.45**(3.71) | 7.484(2.65) | 14989.68(51.86) | 38.8 |
| *Truncated* | 21.73(0.72) | 150.94(3.90) | 7.454(2.64) | 14990.39(54.49) | 85.2 |
| *Slice efficient* | 66.67(2.22) | 225.38(5.82) | 7.481(2.65) | 14990.43(54.78) | **14.7** |
| *Algo 8* ($m = 2$) | 11.33(0.38) | 97.28(2.51) | 7.478(2.65) | 14989.76(52.48) | 9.76 |

## Appendix C

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 (Tables 23, 24, 25).

**Table 23** Bimod data $n = 100$, $L_D = 150$, $L_C = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **25.77**(0.67) | **7.75**(0.14) | 4.726(3.42) | 267.96(9.97) | 6.88 |
| *Trunc. exch. (ours)* | 26.47(0.68) | 7.97(0.15) | 4.650(3.06) | 267.93(9.99) | **1.77** |
| *Truncated* | 71.53(1.85) | 9.88(0.18) | 5.067(3.93) | 267.87(10.00) | 4.39 |
| *Slice efficient* | 97.86(2.53) | 16.35(0.30) | 4.743(3.42) | 267.95(10.05) | 14.0 |
| *Algo 8* ($m = 2$) | 14.27(0.37) | 5.79(0.11) | 4.720(3.40) | 267.95(9.99) | 1.82 |

**Table 24** Bimod data $n = 1000$, $L_D = 150$, $L_C = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **42.74**(1.80) | $\underline{2.60}$(0.05) | 4.427(3.21) | 2646.88(9.02) | 19.3 |
| *Trunc. exch. (ours)* | $\underline{46.05}$(1.94) | **2.54**(0.05) | 4.401(3.05) | 2646.88(9.01) | **3.30** |
| *Truncated* | 89.45(3.77) | 2.82(0.05) | 4.525(3.38) | 2646.89(9.05) | $\underline{8.27}$ |
| *Slice efficient* | 200.64(8.46) | 6.23(0.11) | 4.446(3.21) | 2646.88(9.03) | 19.5 |
| *Algo 8* ($m = 2$) | 22.80(0.96) | 2.15(0.04) | 4.425(3.18) | 2646.88(8.99) | 3.15 |

**Table 25** Lepto data $n = 100$, $L_D = 200$, $L_C = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

| | IAT on # of clusters ↓ | IAT on deviance ↓ | Estimated # clusters | Estimated deviance | Time (ms)↓ |
|---|---|---|---|---|---|
| *Slice exch. thres (ours)* | **47.85**(1.60) | 27.00(0.57) | 3.719(3.70) | 223.92(8.55) | 6.00 |
| *Trunc. exch. (ours)* | 50.04(1.67) | **26.91**(0.57) | 3.674(3.39) | 223.86(8.61) | **0.99** |
| *Truncated* | 93.37(3.11) | 35.96(0.76) | 3.955(4.18) | 223.78(8.75) | 2.93 |
| *Slice efficient* | 224.68(7.49) | 74.51(1.57) | 3.696(3.68) | 223.94(8.49) | 11.2 |
| *Algo 8* ($m = 2$) | 27.28(0.91) | 17.83(0.38) | 3.720(3.71) | 223.92(8.55) | 1.02 |

**Acknowledgements** The authors would like to thank the two anonymous reviewers for their valuable comments and suggestions, which have contributed to the improvement of the quality of the manuscript.

**Funding** Open access funding provided by Université d' Orleans.

**Data availibility** All data and scripts are available at this link.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

## References

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

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

Bush CA, MacEachern SN (1996) A semiparametric Bayesian model for randomised block designs. Biometrika 83(2):275–285

Canale A, Corradin R, Nipoti B (2022) Importance conditional sampling for Pitman-Yor mixtures. Stat Comput 32(3):40

Caron F, Teh YW, Murphy TB (2014) Bayesian nonparametric Plackett-Luce models for the analysis of preferences for college. Annals of Applied Statistics 8(2):1145–1181

Donnet S, Rivoirard V, Rousseau J, et al (2018) Posterior concentration rates for empirical Bayes procedures with applications to Dirichlet process mixtures. Bernoulli 24(1):231–256. https://hal.science/hal-01570308, arXiv:1406.4406v1

Dubey KA, Zhang M, Xing E, et al (2020) Distributed, partially collapsed MCMC for Bayesian nonparametrics. International Conference on Artificial Intelligence and Statistics pp 3685–3695

Escobar M (1994) Estimating normal means with a Dirichlet process prior. J Am Stat Assoc 89:268–277

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

Fall MD, Barat É, Dautremer T (2024) Hierarchical nonparametric Bayesian modeling for clinical 3D PET data reconstruction. Submitted

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

Favaro S, Walker SG (2013) Slice sampling $\sigma$-stable Poisson-Kingman mixture models. J Comput Graph Stat 22:830–847

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

Ferguson T, Klass M (1972) A representation of independent increment processes without Gaussian components. Ann Math Stat 43(5):1634–1643

Gelfand AE, Kottas A (2002) A computational approach for full nonparametric Bayesian inference under Dirichlet process mixture models. J Comput Graph Stat 11(2):289–305

Ghosal S, van der Vaart A (2017) Fundamentals of Nonparametric Bayesian inference. Cambridge Series in Statistical and Probabilistic Mathematics 44. https://books.google.fr/books?id=cs8oDwAAQBAJ

Green PJ, Richardson S (2001) Modelling heterogeneity with and without the Dirichlet process. Scand J Stat 28:355–375

Griffin JE, Walker SG (2011) Posterior simulation of Normalized Random Measure mixtures. J Comput Graph Stat 20:241–259

Ishwaran H, James LF (2001) Gibbs sampling methods for stick-breaking priors. J Am Stat Assoc 96:161–173

Jacquier E, Polson NG, Rossi PE (2004) Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. Journal of Econometrics 122(1):185–212

Kalli M, Griffin JE, Walker SG (2011) Slice sampling mixture models. Stat Comput 21(1):93–105

Lo AY (1984) On a class of Bayesian Nonparametric estimates: I. density estimates. Ann Stat 12:351–357

Lomeli M, Favaro S, Teh YW (2015) A hybrid sampler for Poisson-Kingman mixture models. Advances in Neural Information Processing Systems 28

MacEachern SN, Müller P (1998) Estimating mixture of Dirichlet process models. J Comput Graph Stat 7:223–238

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

Müller P, Mitra R (2013) Bayesian Nonparametric Inference-Why and How. Bayesian Anal 8(2):269–302

Müller P, Quintana F (2004) Nonparametric Bayesian data analysis. Stat Sci 19(1):95–110

Müller P, Quintana F, Jara A, et al (2015) Bayesian nonparametric data analysis. Springer Series in Statistics 1. https://books.google.fr/books?id=eH5ErgEACAAJ

Neal RM (1991) Bayesian mixture modeling. Maximum entropy and Bayesian Methods: Proceedings of the 11th International Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis, Seattle pp 197–211

Neal RM (2000) Markov chain sampling methods for Dirichlet process mixture models. J Comput Graph Stat 9(2):249–265

Nieto-Barajas LE, Prünster I (2009) A sensitivity analysis for Bayesian nonparametric density estimators. Stat Sin 19:685–705

Nieto-Barajas LE, Prünster I, Walker SG (2004) Normalized random measures driven by increasing additive processes. Ann Stat 32(6):2343–2360. https://doi.org/10.1214/009053604000000625

Papaspiliopoulos O (2008) A note on posterior sampling from Dirichlet mixture models. Technical report

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

Pitman J (1996) Random discrete distributions invariant under size-biased permutation. Adv Appl Probab 28:525–539

Pitman J (1996) Some developments of the Blackwell-MacQueen urn scheme. Lecture Notes-Monograph Series 30:245–267

Pitman J (2003) Poisson-Kingman partitions. Statistics and Science: A Festschrift for Terry Speed, IMS Lectures Notes Monograph 40:D. R. Goldstein

Porteous IR, Ihler A, Smyth P, et al (2006) Gibbs sampling for (coupled) infinite mixture models in the stick breaking representation. Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence pp 385–392

Roeder K (1990) Density estimation with confidence sets exemplified by superclusters and voids in the galaxies. J Am Stat Assoc 85(411):617–624

Sokal AD (1997) Monte Carlo methods in Statistical Mechanics: Foundations and new algorithms. NATO Adv Sci Inst Ser B Phys 361:131–192

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

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

West M, Müller P, Escobar MD (1994) Hierarchical priors and mixture models, with application in regression and density estimation. Aspects of Uncertainty pp 363–386

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.