Gibbs sampling methods for Pitman-Yor mixture models

Mame Diarra Fall, Éric Barat

Noname manuscript No.
(will be inserted by the editor)

Gibbs sampling methods for Pitman-Yor mixture models

Mame Diarra Fall · Éric Barat

Received: date / Accepted: date

Abstract We introduce a new sampling strategy for the two-parameter Poisson-Dirichlet process mixture model, also known as Pitman-Yor process mixture model (PYM). Our sampler is therefore applicable to the well-known Dirichlet process mixture model (DPM). Inference in DPM and PYM is usually performed via Markov Chain Monte Carlo (MCMC) methods, specifically Gibbs sampler. These sampling methods are usually divided in two classes: marginal and conditional algorithms. Each method has its merits and limitations. The aim of this paper is to propose a new sampler which combines the main advantages of each class. Our method relies on a result of [Pit96b] for updating Pitman-Yor processes. The infinite part of the unconditional process is sampled in two ways, leading to two variants of the proposed sampler. We also propose a threshold to improve mixing in the first variant of our algorithm. The two variants of our sampler are compared with a marginal method (algorithm 8 of [Nea00]) and two state of the art conditional algorithms which are formulated in the space of cluster labels namely the efficient slice sampler of [KGW11] and the truncated blocked Gibbs sampler of [IJ01]. We also investigate effects of removing the proposed threshold in the first variant of our algorithm and introducing the threshold in the efficient slice sampler of [KGW11]. Results on real and simulated data sets illustrate that our algorithms outperform the other conditionals in terms of mixing properties.

M.D. Fall
Laboratoire MAP5, Université Paris Descartes and CNRS, Sorbonne Paris Cité.
E-mail: diarra.fall@parisdescartes.fr

É. Barat
DRT/LIST/DCSI/LM2S, CEA Saclay.
E-mail: eric.barat@cea.fr

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

1 Introduction

Bayesian nonparametrics have shown their utility in a great number of applications in statistics and machine learning (density estimation, clustering, image segmentation and reconstruction, language modeling etc.) and their interest is increasing. In this context, it becomes necessary to investigate mixing properties of MCMC samplers, particularly in high dimensional spaces with large data sets.

After the first MCMC method proposed by [Esc94] for inferring DPM, many authors contributed to its improvement ([EW95], [MM98]) and to handle non-conjugate cases [Nea00]. These aforementioned techniques marginalize the random distribution using Dirichlet distribution properties and sample draws from it. Efficient versions of marginal algorithms usually achieve good mixing performances [Nea00]. An alternative to marginal methods is given by the the so-called conditional algorithms which explicitly represent the measure generated by the process using its stick-breaking representation ([IJ01], [Wal07]). The challenge in this approach is to deal with the countably infinite representation of the distribution. In [IJ01], authors resort to an approximation and truncate the measure at a chosen value N. An alternative which avoids truncations was introduced by [PR07] and [Wal07]. This later algorithm was improved by [Pap08] and [KGW11]. The advantage of conditional methods is that they take into account the measure generated by the process making its inference possible. However, as it was pointed out by

[PISW06], in conditional methods using stick-breaking representation, the sampling occurs in the space of non-exchangeable clusters labels. As a consequence, clusters prior labeling contributes to the posterior sampling and additional moves in the MCMC algorithm are necessary to improve mixing over clusters labels.

In this context, we introduce a new conditional sampling scheme which is formulated in the space of equivalence classes over clusters labels where clusters identities are irrelevant. This is the adequate space because nothing observable depends on these labels.

The remainder of this paper is structured as follows. In section 2, we recall some preliminaries on definitions and results about Pitman-Yor processes. In section 3, we briefly recall the basis of the algorithms that are used to compare our samplers and discuss the advantages and limitations of marginal and conditional methods. Afterwards, in section 4, we derive explicitly a truncated Gibbs sampler working in the equivalence class of random exchangeable partitions and a new exchangeable truncated slice Gibbs sampler which is free of approximation. We evaluate performance of the algorithms through application to one real and two simulated data sets in section 5. Finally, we conclude the paper in section 6 with discussions and extensions for further work.

2 Preliminaries on Pitman-Yor processes

2.1 Some basic definitions and results ([Pit96b], [PY97], [Pit02])

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

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

Definition 2 *(Two-parameter Poisson-Dirichlet distribution)*
Let $\mathbf{w} \sim GEM(d, \alpha)$. The ranked size sequence $\widetilde{\mathbf{w}} = r(\mathbf{w})$ where $\widetilde{w}_1 \geq \widetilde{w}_2 \geq \ldots$ and $r(\cdot)$ the sequence ranking function, has a two-parameter Poisson-Dirichlet distribution with parameters d and α and is denoted by

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

[1] The acronym stands for Griffiths, Engen and McCloskey.

Definition 3 *(Pitman-Yor process)*
Let $d \in [0,1]$, $\alpha > -d$ and $\widetilde{\mathbf{w}} \sim PD(d, \alpha)$[2]. Let G_0 be a diffuse (non-atomic) probability measure on a measurable space $(\Theta, \mathcal{B})$ (i.e $G_0(\boldsymbol{\theta}) = 0$ for all $\boldsymbol{\theta} \in \Theta$). Consider $Z_1, Z_2, \ldots$ to be iid G_0, taking values on Θ and independent of $\widetilde{\mathbf{w}}$. Let $\delta_Z(\cdot)$ denote the Dirac measure giving mass 1 at Z. Then, the random probability measure

$$H(\cdot) = \sum_{k=1}^{\infty} \widetilde{w}_k \delta_{Z_k}(\cdot),$$

is a two-parameter Poisson-Dirichlet process (a.k.a Pitman-Yor process) on $(\Theta, \mathcal{B})$, with parameters d and α and base measure G_0. It is denoted by:

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

The parameters α and d tune the variability of generated measures around G_0. Setting $d = 0$, the Pitman-Yor process reduces to the Dirichlet process with parameter α and base measure G_0, denoted $\mathrm{DP}(\alpha, G_0)$.

Like DP, the PYP has several properties, for instance a stick-breaking representation and a characterization in terms of generalized Blackwell-MacQueen urn scheme. Since the algorithms we will describe later on are based upon the Pólya urn characterization or the stick-breaking construction of the process, we recall in the following the basis of these representations.

2.2 Stick-breaking characterization of PYP

2.2.1 Stick-breaking representation

Ishwaran and James ([IJ01], [IJ03]) extended the stick-breaking representation of the Dirichlet process due to Sethuraman [Set94] to more general random measures that encompass both DP and PYP. The stick-breaking representation is a constructive definition based on a residual allocation model and is given as follows. Let $d \in [0,1]$, $\alpha > -d$, $\mathbf{w} \sim GEM(d, \alpha)$ and G_0 a diffuse probability measure on Θ. Let $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*, \ldots)$ be *iid* G_0, independently of $\mathbf{w}$. Let us construct

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

Under these assumptions, we have

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

[2] There is another parametrization of the PYP, with $d = -\kappa < 0$ et $\alpha = m\kappa$ for some $\kappa > 0$ and $m = 2, 3, \ldots$

2.2.2 Posterior distribution under stick-breaking

Let H be a random measure generated via a stick-breaking process. Consider $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ a sample from H. Let us define the membership variables $\mathbf{c} = (c_1, \ldots, c_n)$ such that $c_i = k$ iff $\boldsymbol{\theta}_i = \boldsymbol{\theta}^*_k$. Let $\mathbf{c}^* = (c^*_1, \ldots, c^*_{K_n})$ the set of unique values of $\mathbf{c}$ and $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}^*_1, \ldots, \boldsymbol{\theta}^*_{K_n})$ the set of unique values of $\boldsymbol{\Theta}$. Then,

$$H(\cdot)|\boldsymbol{\theta} = \sum_{k \in \mathbf{c}^*} w^*_k \delta_{\boldsymbol{\theta}^*_k}(\cdot) + \sum_{k \notin \mathbf{c}^*} w_k \delta_{Z_k}(\cdot), \tag{2}$$

where for all $k \notin \mathbf{c}^*$, $Z_k \overset{iid}{\sim} G_0$, the weights w^*_k follow a GEM distribution with updated parameters: $w^*_1 = v^*_1, w^*_2 = v^*_2(1 - v^*_1), \ldots, w^*_n = v^*_n \prod_{i=1}^{n-1}(1 - v^*_i)$ where $v^*_l \sim \text{Beta}(1 - d + n_l, \alpha + ld + \sum_{m=l+1}^{\infty} n_m)$ with $n_m = \#\{c_k : c_k = m\}$.

2.3 Generalized Blackwell-MacQueen representation of PYP

The PYP can also be described by a generalization of the Blackwell-MacQueen urn scheme ([Pit95], [Pit96a]). This result stems from characterization of PYP via species sampling model (SSM).

2.3.1 Prediction rule characterization of PYP

Let $d \in [0, 1]$, $\alpha > -d$ and G_0 a diffuse probability measure on Θ. Consider a sequence of $(\boldsymbol{\theta}_i)$ generated via the following predictive distributions:

$$\boldsymbol{\theta}_1 \sim G_0 \tag{3}$$

$$\boldsymbol{\theta}_{n+1}|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \sim \frac{\alpha + dK_n}{\alpha + n} G_0 + \sum_{j=1}^{K_n} \frac{n_j - d}{\alpha + n} \delta_{\boldsymbol{\theta}^*_j}, \tag{4}$$

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

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

where $\mathbf{w} \sim \text{GEM}(d, \alpha)$, $\boldsymbol{\theta}^*_1, \ldots, \boldsymbol{\theta}^*_n \sim G_0$ and are independent of $\mathbf{w}$.

The predictive distributions (3)-(4) are the key components of marginal methods described in section 3.

Remark 1
Note that in the representation (5), the weights are given by the empirical frequencies: $w_k = \lim_{n \to \infty} \frac{n_k}{n}$, where the frequencies are defined in the order of appearance of species.

2.3.2 Posterior distribution of PYP under species sampling process

We present here an important result from ([Pit96b], Corollary 20). A detailed proof of this corollary can be found in [Car99]. This result is the basis of the new sampling method we propose to infer the mixture of Pitman-Yor processes. This corollary characterizes the posterior distribution of PYP under SSM.

Let $H \sim \text{PY}(d, \alpha, G_0)$ where G_0 is a diffuse probability measure s.t. $\mathbb{E}(H) = G_0$. Let $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n|H \sim H$ and K_n the number of distinct values of $\boldsymbol{\theta}_i$, let $\{\boldsymbol{\theta}^*_j\}_{j=1}^{K_n}$ the set of unique values of $\{\boldsymbol{\theta}_i\}_{i=1}^n$ and finally let n_j the number of occurrences of $\boldsymbol{\theta}^*_j$ in the sample. Then, we have

$$H|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \overset{d}{=} \sum_{j=1}^{K_n} w_j \delta_{\boldsymbol{\theta}^*_j} + r_{K_n} H_{K_n}, \tag{6}$$

where

$$(w_1, \ldots, w_{K_n}, r_{K_n}) \sim \text{Dir}(n_1 - d, \ldots, n_{K_n} - d, \alpha + dK_n)$$
$$H_{K_n} \sim \text{PY}(d, \alpha + dK_n, G_0),$$

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

We will detail more this result and its consequences in section 4 dedicated to our new sampler.

3 Sampling from DPM and PYM

3.1 Pitman-Yor process mixture model (PYM)

The well-known Dirichlet process mixture model (DPM) due to [Fer83] and [Lo84] can be extended to define Pitman-Yor process mixture models in the following way

$$\mathbf{x} \sim \int p(\cdot|\boldsymbol{\theta}) \mathrm{d}H(\boldsymbol{\theta}) \quad \text{with} \quad H \sim \text{PY}(d, \alpha, G_0), \tag{7}$$

where $p(\cdot|\boldsymbol{\theta})$ is a kernel, for instance a Gaussian with parameters $\boldsymbol{\theta} = (\mu, \sigma^2)$ with μ being the mean and σ^2 the variance of the normal component.

The model (7) can be expressed in a hierarchical Bayesian model as follows

$$\begin{aligned} \mathbf{x}_i|\boldsymbol{\theta}_i &\sim p(\mathbf{x}_i|\boldsymbol{\theta}_i) \\ \boldsymbol{\theta}_i|H &\sim H \\ H|d, \alpha, G_0 &\sim \text{PY}(d, \alpha, G_0). \end{aligned} \tag{8}$$

The parameters $\boldsymbol{\theta}_i$ are conditionally independent given H and each observation $\mathbf{x}_i$ is conditionally independent on the others given $\boldsymbol{\theta}_i$.

Since H is discrete, some $\boldsymbol{\theta}_i$ will be identical. One can consider observations having the same parameter $\boldsymbol{\theta}_i$ as belonging to the same component. Then, latent indicator variables can be introduced that assign observations to mixture components. Let c_i such that $c_i = k$ iff $\boldsymbol{\theta}_i = \boldsymbol{\theta}^*_k$, where $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \dots)$ denote the unique values among $(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n)$. The parameter $\boldsymbol{\theta}_i$ takes the value $\boldsymbol{\theta}^*_k$ with probability w_k. The model (8) can be expressed hierarchically as follows using stick-breaking representation:

$$\begin{aligned} \mathbf{x}_i|c_i, \boldsymbol{\Theta}^* &\sim p(\mathbf{x}_i|\boldsymbol{\theta}^*_{c_i}) \\ c_i|\mathbf{w} &\sim \sum_{k=1}^{\infty} w_k \delta_k(\cdot) \\ \mathbf{w}|d, \alpha &\sim \mathrm{GEM}(d, \alpha) \\ \boldsymbol{\theta}^*_k|G_0 &\sim G_0. \end{aligned} \tag{9}$$

3.2 MCMC algorithms for PYM

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

3.2.1 Marginal methods

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

Algorithm 8 of [Nea00]: The idea behind "algorithm 8" of [Nea00] is to add auxiliary components (representing potential future components) in order to avoid evaluating the intractable integral when updating classification variables. Since data are exchangeable and labels of components completely arbitrary, each datum $\mathbf{x}_i$ can be treated as the last. It is then assigned to an already represented component or to an auxiliary component. If we denote by K^-_n the number of active components disregarding observation i, the prior probability to allocate $\mathbf{x}_i$ to an active component is $\frac{n_{-i,k}-d}{\alpha+n-1}$ and the probability to create a new component is $\frac{\alpha+dK^-_n}{\alpha+n-1}$, which will be equally distributed among the m auxiliary components. The choice of m is left to the user. It is governed by a balance between computational considerations and mixing properties. When updating classification variables, the following rules are used: if $c_i = c_j$ for some $j \neq i$ (i.e $\mathbf{x}_i$ belongs to a non-singleton component), auxiliary parameters are drawn indepedenlty from the prior G_0. Otherwise, if $c_i \neq c_j$ for all $j \neq i$, the observation will be allocated to one of the m auxiliary components. To do this, one could randomly choose one of the m auxiliary components but thanks to the exchangeability, this does not matter and $\mathbf{x}_i$ is allocated to the first auxiliary component (i.e $c_i = K^-_n + 1$), with component parameter $\boldsymbol{\theta}^*_{c_i}$. Parameters of the $m-1$ other auxiliary components (if $m > 1$) are drawn independently from G_0. The update of c_i is done via the conditional probabilities:

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

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

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

3.2.2 Conditional methods

Marginal approaches integrate out the random measure H and draw values from it. In contrast, conditional methods retain the random distribution and explicitly represent it using its stick-breaking construction. Among them, there are methods that approximate the random measure by truncating the number of its components, and those that are not based on approximations.

The truncated blocked Gibbs sampler [IJ01]
An issue with non-marginal approaches is to treat the infinite number of components. To sidestep this, Ishwaran and James ([IJ01], [IJ03]), using the fact that weights in the GEM distribution decrease exponentially fast in law, showed that the number of components can be truncated at a chosen integer value N. It is necessary to set $v_N = 1$ in the stick-breaking construction of H to ensure that the truncated measure H_N is a probability measure, with distribution:

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

Under this truncated framework, the hierarchical PYM (9) can be rewritten as follows

$$\begin{aligned} \mathbf{x}_i|c_i, \boldsymbol{\Theta}^* &\sim p(\mathbf{x}_i|\boldsymbol{\theta}^*_{c_i}) \\ c_i|\mathbf{w} &\sim \sum_{k=1}^{N} w_k \delta_k(\cdot) \\ \mathbf{w}|d, \alpha &\sim \mathrm{GEM}(d, \alpha) \\ \boldsymbol{\theta}_k^*|G_0 &\sim G_0, \end{aligned} \tag{10}$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ represent the data, $\mathbf{c} = (c_1, \ldots, c_n)$ the classification variables, $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}_1^*, \ldots, \boldsymbol{\theta}_N^*)$ the parameters of the mixture components and $\mathbf{w} = (w_1, \ldots, w_N)$ the components weights. Rewriting the model in this form makes direct posterior inference possible since one has to treat a finite number of components. The algorithm works by drawing values from the following conditional distributions of the *blocked* variables

Allocation variables:	$\mathbf{c}|\mathbf{w}, \boldsymbol{\Theta}^*, \mathbf{X}$.
PYP weights:	$\mathbf{w}|\mathbf{c}$.
Components parameters:	$\boldsymbol{\Theta}^*|\mathbf{c}, \mathbf{X}$.

If we denote by K_n the number of currently non-empty components, the full conditionals involved in the Gibbs sampler are given by the following:

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

   $$w_1 = v_1^* \text{ et } w_k = v_k^* \prod_{l=1}^{k-1}(1 - v_l^*) \quad \text{for } k = 2, \ldots, N-1,$$

   with

   $$v_k^* \sim \mathrm{Beta}(1-d+n_k,\ \alpha+kd+\sum_{l=k+1}^{N} n_l) \text{ for } k = 1, \ldots, N-1,$$

   where $n_k = \#\{i : c_i = k\}$.
3. Conditional for $\boldsymbol{\Theta}^*$:
   - $\boldsymbol{\theta}_k^* \sim G_0$ for $k = K_n + 1, \cdots, N$.
   - $\boldsymbol{\theta}_k^*|\mathbf{c}, \mathbf{X}$ has density proportional to

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

This sampler is easy to implement since the truncation allows it to be similar to standard Gibbs samplers in finite dimensional models. However, even if methods for controlling the truncation accuracy have been proposed ([IJ01], [IJ03]), it would be better to keep the exact infiniteness nature of the distributions and avoid any hard approximation. To this purpose, algorithms which sample from the exact posterior distributions while requiring a finite number of components at each iteration have been proposed by Papaspiliopoulos and Roberts [PR07] and Walker [Wal07]. This later uses an elegant strategy called *slice sampling* and is based on auxiliary variables. Walker's slice sampling was improved by Papaspiliopoulos [Pap08] and Kalli *et al.* [KGW11]. This later is named after "slice efficient".

The slice sampler ([Wal07], [KGW11])
The idea under the slice sampler proposed by [Wal07] for inference in DPM is to introduce auxiliary variables which make the mixture model conditionally finite. To make it precise, let us consider the model (8) with kernel p where H is constructed using the stick-breaking representation. The density of a single observation $\mathbf{x}_i$, given $\mathbf{w}$ and $\boldsymbol{\Theta}^*$, is

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

Let us introduce $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ uniform auxiliary variables such that the joint density of any $(\mathbf{x}_i, u_i)$ is

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

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

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

Given u_i, the number of components in the mixture is finite. One can complete the model by introducing an assignment variable c_i and considering

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

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

1. Conditional for u_i:
$$u_i \sim \mathcal{U}(u_i|0,w_{c_i}).$$
2. Conditional for $\boldsymbol{\theta}_k^*$:
   - for non-empty components, the density of $\boldsymbol{\theta}_k^*$ is proportional to
   $$G_0(\mathrm{d}\boldsymbol{\theta}_j^*) \prod_{\{i:c_i=j\}} p(\mathbf{x}_i|\boldsymbol{\theta}_j^*),$$
   - for empty components, $\boldsymbol{\theta}_k^* \sim G_0$.
3. Conditional for w_k :
   This conditional is given in the truncated blocked Gibbs sampler of [IJ01] previously described.
4. Conditional for c_i:
   $\mathbb{P}(c_i = k) \propto \mathbf{1}(k : w_k > u_i)\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*)$.

To sample from this probability mass function, one needs to know the exact number of components that are required at each iteration of the sampler. It is given by the smallest K such that

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

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

In this section, we have outlined a marginal and two conditional methods for inference in Pitman-Yor mixture models. In the following paragraph, the merits and limitations of each class of algorithms are highlighted.

3.2.3 Advantages and disadvantages of each class

The key advantage of conditional methods using stick-breaking construction is in updating the random measure generated by the process as well as parameters drawn from it. This makes direct inference possible for this measure. In addition, by construction, they do not suffer from the issue of conjugacy encountered in marginal methods. Furthermore, components weights are explicitly represented. As a consequence, updating indicator variables in the allocation step is done without conditioning on the other indicators. This property makes these algorithms able to update blocks of parameters and easy to implement in a parallel computer, which is well suited particularly for large data sets.

On the flip side, by integrating mixture components out of the model, marginal techniques make the allocation step very sequential since they need to condition on all previously allocated data. These incremental updates are prejudicial when working with huge data sets where computational efficiency is essential. Another drawback of marginalizing over the prior is that computing posterior conditionals require additional sampling steps (see [IJ01]). However, dealing with exchangeable prediction rules, marginal methods exhibit most of the time better mixing properties, and our experimental comparison in section 5 corroborates this assessment. Another important point is that the random weights are collapsed by marginalization and this results in a crucial reduction of the parameters space dimension.

Another aspect pointed out by [PISW06] is that in conditional methods based on the stick-breaking process, the sampler operates in the space of non-exchangeable cluster labels. Indeed, in this representation, weights are explicitly defined by the prior and components are represented with a size-biased ordering over their labels. This means that components with lower labels have higher prior probabilities than components with higher labels. As a consequence, components are not interchangeable and cluster priors labeling contributes to the posterior sampling. In this situation, the sampler needs to mix efficiently over clusters labels to avoid any clustering bias. [PISW06] recommend systematic use of two additional Metropolis-Hastings moves ("label-swap" and "label-permute") in order to improve mixing over clusters. When working with non-exchangeable clusters labels, this additional step seems to be the only way to improve the mixing over clusters (see also

[Pap08]). In contrast, in marginal methods using Pólya urn representation, the sampling occurs in the space of equivalence classes over exchangeable clusters labels ($i \sim j$ iff $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$) where clusters identities are arbitrary and insignificant. This is the adequate space to live for the sampler because cluster labels are irrelevant.

In this context, we propose a conditional algorithm which is rather different from the others discussed so far. As opposed to the other conditional methods using stick-breaking representation, our sampler lives in the space of equivalence classes over clusters labels. These labels are then exchangeable and no mix over them is needed. This property has important consequences on the algorithm mixing.

4 A new sampling method

Our sampler is an attempt to combine the main advantages of marginal and conditional algorithms. The underlying idea is to integrate out the explicit order of clusters labels like in marginal methods hence collapsing the model to a lower dimensional space while keeping components weights as done in conditional approaches. Then, we aim to the following characteristics:

1. Follow a conditional approach by retaining the random distribution function generated by the process.
2. Sample in the space of equivalence classes of exchangeable clusters to obtain a better mixing chain.
3. Target easy parallel implementation of the allocation step in order to deal with large data sets.

Our method relies on a result of ([Pit96b], Corollary 20) which expresses the posterior distribution of limit relative frequencies of atoms in a species sampling model (SSM) based on a two-parameter Pólya urn (see section 2.3.2). This model ensures exchangeability of the underlying random partition, characterized by the symmetry of the exchangeable partition probability function (EPPF). The symetric EPPF determines the distribution of the exchangeable random partition $\{1, 2, \dots\}$ whose classes are the equivalence classes for the random equivalence relation defined by $i \sim j$ iff $c_i = c_j$. In particular, $c_1 = 1$ implies that $\mathcal{A}_1$ contains all m s.t. $c_m = c_1$. This implies that the label j of any class given by the order of appearance of the j^{th} species is an arbitrary tag and does not preexist to the sampling process.

We have seen, in section (2.3.2), that if

$$H \sim \mathrm{PY}(d, \alpha, G_0),$$

where G_0 is a non-atomic probability measure and if we consider a sample $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n$ from H, then the posterior of H can be expressed as follows

$$H|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n \overset{d}{=} \sum_{j=1}^{K_n} w_j \delta_{\boldsymbol{\theta}_j^*} + r_{K_n} H_{K_n}, \tag{11}$$

where

$$(w_1, \dots, w_{K_n}, r_{K_n}) \sim \mathrm{Dir}(n_1 - d, \dots, n_{K_n} - d, \alpha + dK_n),$$
$$H_{K_n} \sim \mathrm{PY}(d, \alpha + dK_n, G_0),$$

and H_{K_n} is independent of $(w_1, \dots, w_{K_n}, r_{K_n})$.

Pitman showed in [Pit96a] equivalence between exchangeability of the random partition induced by the model and symmetry in the law characterizing the limiting frequencies of occupied components given the data. We can easily check that exchangeability is ensured in equation (11) since it sums to a Dirichlet distribution (symmetric) and a rescaled ISBP Pitman-Yor process (independent of observed data). It is worth mentioning that exchangeability is lost when using the usual conditional updating in the general Sethuraman stick-breaking representation of Pitman-Yor processes (equation (2)). Furthermore, in the SSM representation, atoms correspond to equivalence classes and their labels are defined *in the order of appearance* of new species. On the contrary, labels are explicit in general stick-breaking approaches and predefined by the prior before any sampling sequence. This property is not necessary and has the impact of bothering the Gibbs sampler.

4.1 Proposed variants

For sampling the independent infinite process H_{K_n} in equation (11), we propose two variants. The first makes use of a thresholded version of the "slice efficient dependent" of [KGW11]. The second is based on a truncation of the posterior Pitman-Yor mixture, originally suggested in [IJ01].

4.1.1 Exchangeable Thresholded Slice Sampler

We first propose a slice sampler inspired from [Wal07] and [KGW11] for sampling the posterior of a Pitman-Yor mixture process model. The main steps are now summarized. Let us introduce $\mathbf{u} = (u_1, u_2, \dots, u_n)$ uniform auxiliary variables such that the joint density for any $(\mathbf{x}_i, u_i)$, given $\mathbf{w}$ and $\boldsymbol{\Theta}^*$, is

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

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

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

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

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

where $\mathbb{E}_{\alpha,d}(K_n) = \sum_{i=1}^{n} \frac{(\alpha+d)_{i-1\uparrow}}{(\alpha+1)_{i-1\uparrow}}$ (with $(x)_{a\uparrow} = \Gamma(x+a)/\Gamma(x)$) which can be fairly approximated for sufficiently large n by $\mathbb{E}_{\alpha,d}(K_n) \approx \frac{\Gamma(\alpha+1)}{d\,\Gamma(\alpha+d)} n^d$ (see [Pit02]).

Using equation (13), we can rewrite equation (12) as follows:

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

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

The Gibbs sampler allows to generate variables from the joint posterior $(\boldsymbol{\Theta}^*, \mathbf{c}, \mathbf{w}, \mathbf{u}|\mathbf{X})$, by sampling iteratively from each full conditional. As in [KGW11], we jointly sample $\mathbf{w}, \mathbf{u}|\mathbf{c}$. The full conditional distributions involved in the steps of the Gibbs sampler are then:

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

We now detail each conditional.

1. Conditional for $(\mathbf{w}, \mathbf{u})$:
   We denote by K_n the number of distinct clusters after n observations and n_k the number of observations belonging to cluster k, for $k \le K_n$. We jointly sample $\mathbf{w}, \mathbf{u}|\mathbf{c}$ by first sampling $w_1, w_2, \ldots, w_{K_n}|\mathbf{c}$, then sampling $\mathbf{u}|w_1, w_2, \ldots, w_{K_n}, \mathbf{c}$, and finally sampling $w_{K_n+1}, w_{K_n+2}, \ldots|\mathbf{u}$. The mains steps are now given.
   – Sample w_k for $k \le K_n$,

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

   – 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 $r_{k-1} > u^*$,

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

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

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

   Note that, at each iteration, non-empty clusters are re-labeled according to their *order of appearance* in the sampling. We operate in the space of *equivalence classes over non-empty clusters labels* which are thus *exchangeable*. The stick-breaking prior only concerns empty clusters for the given iteration of the Gibbs sampler. As pointed out, this encourages good mixing over clusters in comparison to the stick-breaking sampling of [IJ01]. In the same time, the sampler keeps the random distribution functions.

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

   $$c_i|\mathbf{w}, \mathbf{u}, \boldsymbol{\Theta}^*, \mathbf{X} \overset{\text{ind}}{\sim} \sum_{k=1}^{K^*} w_{k,i}\, \delta_k(\cdot), \tag{14}$$

   where

   $$w_{k,i} \propto \mathbf{1}(w_k > u_i) \max(w_k, \zeta)\, f_{\mathcal{N}}(\mathbf{x}_i|\boldsymbol{\theta}_k^*),$$

   and $\sum_{j=1}^{K^*} w_{k,i} = 1$.
   Note also that, in order to speed up computations, it is convenient to sort weights w_k, $k > K_n$ in decreasing order. By this, we can avoid tests for all $k > \kappa$ as soon as $w_\kappa < u_i$.

3. Conditional for $\boldsymbol{\Theta}^*$:
   – Updating parameters for non-empty components from the density proportional to:

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

   – Sampling parameters for unallocated components from their priors:

   $$\boldsymbol{\theta}_k^* \overset{\text{i.i.d.}}{\sim} G_0, \text{ for } K_n < k \le K^*.$$

The blocked Gibbs sampler structure allows easy implementation of the algorithm on a parallel computer since, at one iteration, the random distributions are retained for the whole data set.

4.1.2 Exchangeable Truncated Gibbs Sampler

The second variant of the algorithm we propose is an alternative of the first one. It is still based on the posterior under a Pitman-Yor distribution given in equation (11). But instead of using the slice sampling strategy to sample the infinite part of the unconditional process, one can resort to an approximation by taking a fixed level L. This truncation eliminates the need of auxiliary variables. This leads to a truncated Pitman-Yor process for the independent part of the process. This scheme was suggested in [IJ01]. We approximate equation (11) by

$$\sum_{j=1}^{K_n} w_j \delta_{\boldsymbol{\theta}_j^*} + r_{K_n} H^*_{K_n},$$

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

- Sample classification variables:

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

  where

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

- Sample w_k for $k \leq K_n$:

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

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

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

  Set $w_{K^*} = r_{K^*-1}$ such that $v_{K^*} = 1$.
- Sample components parameters using
  - the density proportional to

$$G_0(\mathrm{d}\boldsymbol{\theta}_k^*) \prod_{i:c_i=k} p(\mathbf{x}_i|\boldsymbol{\theta}_k) \tag{15}$$

    for non-empty components ($k \leq K_n$),
  - the priors for unallocated components:

$$\boldsymbol{\theta}_k^* \overset{\text{i.i.d.}}{\sim} G_0, \text{ for } K_n < k \leq K^*. \tag{16}$$

5 Comparisons of algorithms

In this section, we evaluate on several data sets the performance of the samplers described in the previous sections and our new sampling method. We thus compare these following algorithms:

- algorithm 8 of [Nea00] ("Algo. 8"),
- the slice efficient of [KGW11] ("Slice efficient"),
- the truncated blocked Gibbs sampler of [IJ01] ("Trunc."),
- the two variants of our sampling scheme using the exchangeable model ("Slice exch. thres." and "Trunc. exch.").

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

Data specification:

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

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

1. The simulated data were generated from the following mixtures of Gaussians.
   - A bimodal mixture (bimod):

$$0.5\,\mathcal{N}(-1, 0.5^2) + 0.5\,\mathcal{N}(1, 0.5^2).$$

   - An unimodal lepto-kurtic mixture (lepto):

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

   These simulated densities are shown in Fig. 1.
   In order to gauge algorithms performance for small and large data sets, we generated $n = 100$, $n = 1,000$ and $n = 10,000$ draws from each of these two mixtures. [3]
2. The real data are Galaxy data, which are the velocities (in 10^3 km/s) of 82 distant galaxies diverging from our own. It is a popular data set in density

[3] Results for $n = 10,000$ are not shown here.

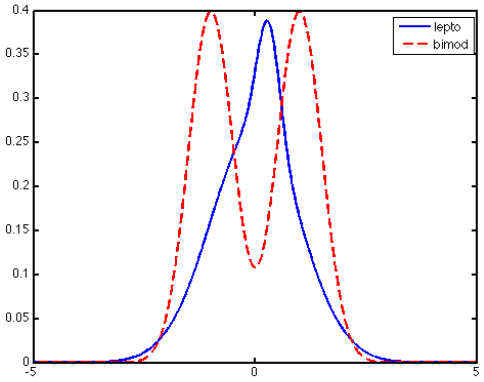

Fig. 1 Bimodal (bimod) and unimodal lepto-kurtic (lepto) mixtures

estimation problems and is also used by [EW95], [GR01] for instance.

Algorithms performance:

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

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

where n_j is the size of cluster j.

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

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

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

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

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

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

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

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

where M is the Monte-Carlo size.

Algorithms parametrization:

At first, we set the discount parameter d of the PYM to zero in order to reduce it to a DPM. The strength parameter of the PYM, that is now the precision parameter of the DPM, was respectively set to $\alpha = \{1, 0.2, 5\}$. Secondly, we investigate the behavior of the competing algorithms in a power-law case (Pitman-Yor). The values $d = 0.3$ and $\alpha = 1$ were chosen for the PYM.

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

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

Data	ϵ
Galaxy ($n = 82$)	7.4139e-04
Lepto/bimod ($n = 100$)	9.0413e-04
Lepto/bimod ($n = 1000$)	8.2446e-06

We also truncate the second variant we propose (for sampling the infinite part the unconditional Pitman-Yor process) at level $L = 2\alpha\log(n)$. The algorithm 8 of [Nea00] was tested with $m = 2$ auxiliary components.

We follow the instructions of [Sok97] who recommends running the samplers for a sufficient number of

iterations. For each of the data sets, we run $2,000,000$ iterations for each algorithm and discarded the first $200,000$ for the burn-in period. We believe that these numbers suffice to obtain reliable results.

Results and comments:

We report in Tables 1-5 the results of our comparisons in the DPM case with $\alpha = 1$. The other results are shown in the appendix.

Each table contains respectively, for each algorithm, the estimated IAT for the mean number of clusters and for the deviance, the estimated mean number of clusters and the estimated deviance. Estimated IAT are obtained by integrating autocorrelation values for each monitored quantity up to a fixed lag (L_D for deviance and L_C for the mean number of clusters). The estimates of standard errors are put inside parentheses.

By looking at the curves of the estimated densities, the values of the estimated deviances and the mean number of clusters, we made sure that all algorithms perform the estimation correctly and then they can be assessed through their mixing performance.

In the overall experiments, it turns out that:

- As expected, algorithm 8 performs better than all conditional algorithms since it works in an unidentifiable allocation structure. Furthermore, integrating out the mixture components speeds up the convergence because the dimensionality of the space is drastically reduced. One can refer to [PR07] and [PISW06] for more details about why conditional approaches are outperformed by marginals.
- On the other hand, the two variants of our method are superior to all other competitors in conditional algorithms, thanks to exchangeability in the model and the introduction of the threshold we propose. The "Slice efficient" gives the worst performance.

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

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

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

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

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

Table 1 Galaxy data $n = 82$, $L_D = 150$, $L_C = 300$.

	IAT for # clusters	*IAT for 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)
Slice exch. without thres.	35.52(0.92)	4.77(0.09)	3.989(0.93)	1561.15(21.61)
Trunc.	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 2 Bimod data $n = 100$, $L_D = 150$, $L_C = 300$.

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

Table 3 Bimod data $n = 1000$, $L_D = 150$, $L_C = 800$.

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

Table 4 Lepto data $n = 100$, $L_D = 200$, $L_C = 500$.

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

Table 5 Lepto data $n = 1000$, $L_D = 150$, $L_C = 800$.

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

6 Conclusion

When models become more and more complex, the poor mixing of a MCMC algorithm can be inhibiting. Therefore, it is important to develop models that allow to improve mixing while experimenting strategies to reduce the computational cost.

In order to satisfy the constraint of efficient parallelization ability while maintaining mixing properties closed to Pólya urn approach, we developed two variants of a new scheme for sampling the posterior of Pitman-Yor type mixture models. We attempted to combine blocking properties of conditional approaches which retain the random distribution in sampling, and exchangeability of the model which is maintained in Pólya urn based algorithms. The resulting algorithms provide interesting mixing behavior. A difference between the two proposed variants is that for the truncated version ("Trunc. exch."), the fixed length of approximation has to be decided before effective sampling. This is most of the time not a crux for Dirichlet processes, but for the two-parameter case the fixed approximation may give rise to biased estimates for moderate truncation lengths. For large lengths, the computational burden is emphasized especially for large data sets. The exchangeable thresholded slice version ("Slice exch. thres.") achieves adaptive truncation at each iteration and maintain nice trade-off between IAT and time cost. This variant gives then rise to convenient trade-off between IAT and computation time while avoiding any hard truncation.

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

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

Appendix

A-RESULTS FOR $d = 0$ AND $\alpha = 5$

Table 6 Galaxy data $n = 82$, $L_D = 150$, $L_C = 300$.

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

Table 7 Bimod data $n = 100$, $L_D = 150$, $L_C = 300$.

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

Table 8 Bimod data $n = 1000$, $L_D = 150$, $L_C = 800$.

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

Table 9 Lepto data $n = 100$, $L_D = 200$, $L_C = 500$.

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

Table 10 Lepto data $n = 1000$, $L_D = 150$, $L_C = 800$.

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

B- RESULTS FOR $d = 0.3$ AND $\alpha = 1$

Table 11 Galaxy data $n = 82$, $L_D = 150$, $L_C = 300$.

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

Table 12 Bimod data $n = 100$, $L_D = 150$, $L_C = 300$.

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

Table 13 Bimod data $n = 1000$, $L_D = 150$, $L_C = 800$.

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

Table 14 Lepto data $n = 100$, $L_D = 200$, $L_C = 500$.

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

Table 15 Lepto data $n = 1000$, $L_D = 200$, $L_C = 1000$.

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

References

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

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

[Car99] M. Carlton. *Applications of the Two-Parameter Poisson-Dirichlet Distribution*. PhD thesis, UCLA Department of Statistics, 1999.

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

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

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

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

[HP98] B. Hansen and J. Pitman. Prediction rules for exchangeable sequences related to species sampling. k *Technical Report 520,* Department of Statistics, University of California, Berkeley, 1998.

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

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

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

[LQMT08] J. Lee, F. A. Quintana, P. Müller, and L. Trippa. Defining predictive probability functions for species sampling models. *Technical report*, Working paper, 2008.

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

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

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

[Nea00] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *J. Comput. Graph. Stat.*, 9(2):249–265, 2000.

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

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

[Pit92] J. Pitman. The two-parameter generalization of Ewens' random partition structure. Technical Report 345, Dept. Statistics, U.C. Berkeley, 1992.

[Pit95] J. Pitman. Exchangeable and partially exchangeable random partitions. *Probab. Th. Rel. Fields*, 102:145–158, 1995.

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

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

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

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

[PPY92] M. Perman, J. Pitman, and M. Yor. Size-biased sampling of Poisson point processes and excursions. *Probability Theory and Related Fields*, 92(1):21–39, 1992.

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

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

[Set94] J. Sethuraman. A constructive definition of Dirichlet priors. *Stat. Sinica*, 4:639–650, 1994.

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

[TE97] S. Tavaré and W. J. Ewens. *Multivariate Ewens distribution*, chapter 41, pages 232–246. Wiley, 1997.

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

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

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