Sampling and inference for discrete random probability measures in probabilistic programs

Benjamin Bloem-Reddy*, **Emile Mathieu***,
Adam Foster, Tom Rainforth, Yee Whye Teh
Department of Statistics
University of Oxford
Oxford, UK
{benjamin.bloem-reddy,emile.mathieu,
adam.foster,rainforth,
y.w.teh}@stats.ox.ac.uk

**Hong Ge, María Lomelí,
Zoubin Ghahramani**
Department of Engineering
University of Cambridge
Cambridge, UK
hg344@cam.ac.uk,
{maria.lomeli,zoubin}@eng.cam.ac.uk

Abstract

We consider the problem of sampling a sequence from a discrete random probability measure (RPM) with countable support, under (probabilistic) constraints of finite memory and computation. A canonical example is sampling from the Dirichlet Process, which can be accomplished using its stick-breaking representation and lazy initialization of its atoms. We show that efficiently lazy initialization is possible if and only if a size-biased representation of the discrete RPM is used. For models constructed from such discrete RPMs, we consider the implications for generic particle-based inference methods in probabilistic programming systems. To demonstrate, we implement SMC for Normalized Inverse Gaussian Process mixture models in Turing.

Bayesian non-parametric (BNP) models are a powerful and flexible class of methods for carrying out Bayesian analysis [25]. By allowing an unbounded number of parameters, BNP models can adapt to the data, providing an increasingly complex representation as more data becomes available. However, a major drawback to BNP modeling is that the resultant inference problems are often challenging, meaning that many models require custom-built inference schemes that are challenging and time consuming to design, thereby hampering the development and implementation of new models and applications. *Probabilistic programming systems* (PPSs) [e.g., 11; 42; 14] have the potential to alleviate this problem by providing an expressive modeling framework, and automating the required inference, making powerful statistical methods accessible to non-experts. *Universal* probabilistic programming languages [11; 29] may be particularly useful in the context of BNP modeling [e.g., 5] because they allow for the number of parameters to vary stochastically and provide automated inference algorithms to suit [40; 42]. Currently, most systems only provide explicit support for Dirichlet Processes (DPs) [8; 35] or direct extensions thereof (see Section 1.2).

The contributions of this paper are threefold. Firstly, we introduce the concept of the *laziest initialization* of a discrete RPM, which provides a computationally and inferentially efficient representation of the RPM suitable for a PPS. We show that this can be carried out only when the atoms of the RPM have a size-biased representation. Secondly, we derive the probability distribution of the number of variables initialized by the commonly used recursive coin-flipping implementation of the DP and its heavy-tailed extension, the Pitman–Yor Process (PYP). We show that this number has finite expectation for only part of its parameter range, indicating that although the coin-flipping recursion halts with probability 1, and thus it is computable, it may be undesirable for practical purposes. Finally, we demonstrate posterior inference for Normalized Inverse Gaussian Process (NIGP) mixture models using Turing [9]. To our knowledge, this is the first BNP mixture model in a PPS with posterior inference that is not the DP or PYP.

*Equal contribution.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1 Size-biased representations and lazy initialization

A discrete random probability measure (RPM) $\mathbf{P}$ on a measurable space $(\mathcal{W}, \Sigma)$ is a countable collection of *probability weights* $(P_j)_{j\geq 1}$ such that $\sum_{j\geq 1} P_j = 1$ a.s., and *atoms* $(\Omega_j)_{j\geq 1} \in \mathcal{W}$ such that $\mathbf{P}(A) = \sum_{j\geq 1} P_j \delta_{\Omega_j}(A)$ a.s. for any $A \in \Sigma$. A *size-biased permutation* π of $\mathbf{P} = (P_j, \Omega_j)_{j\geq 1}$, denoted $\widetilde{\mathbf{P}} = (\widetilde{P}_j, \widetilde{\Omega}_j)_{j\geq 1}$, is a random permutation of the atoms of $\mathbf{P}$ such that [28]

$$(\widetilde{P}_1, \widetilde{\Omega}_1) = (P_{\pi(1)}, \Omega_{\pi(1)}) \quad \text{where} \quad \mathbb{P}(\pi(1) = j \mid P_1, P_2, \dots) = P_j \tag{1}$$
$$(\widetilde{P}_2, \widetilde{\Omega}_2) = (P_{\pi(2)}, \Omega_{\pi(2)}) \quad \text{where} \quad \mathbb{P}(\pi(2) = j \mid \pi(1), P_1, P_2, \dots) = \frac{P_j}{1 - \widetilde{P}_1} ,$$

and so on. If a discrete RPM $\widetilde{\mathbf{P}}$ is equal in distribution to first sampling a realization $\mathbf{P} \sim \mu$ and then applying (1), we say that $\widetilde{\mathbf{P}}$ is a *size-biased version* of $\mathbf{P}$. $\mathbf{P}$ is said to be *lazily initialized* by a computer program if each atom of $\mathbf{P}$ is not instantiated in memory until the first time it is needed by the program; denote by $\widehat{\mathbf{P}}_k$ the first k atoms generated by the program and say that $\widehat{\mathbf{P}}$ is a *lazy size-biased version* of $\mathbf{P}$ if $\widehat{\mathbf{P}}_k \overset{\text{d}}{=} \widetilde{\mathbf{P}}_k$ for all $k \in \mathbb{N}$. Let $\mathbf{X} := (X_1, X_2, \dots)$ be a sequence taking values in $\mathcal{W}$, $\mathbf{X}_n = (X_1, \dots, X_n)$ its size-n prefix, and K_n the number of unique values in $\mathbf{X}_n$. $\widetilde{\mathbf{P}}$ is defined to be *induced* by $\mathbf{X}$ if $\widetilde{\mathbf{P}}$ is realized by labeling the atoms of $\mathbf{P}$ in the order in which they appear in the sample $X_1, X_2, \dots \sim \mathbf{P}$. The following examples illustrate the concept.

Sampling the DP by recursive coin-flipping. The stick-breaking construction of the DP [34; 16] yields a simple way to generate atoms of $\mathbf{P}$ when $\mathbf{P}$ is drawn from a DP prior with concentration parameter $\theta > 0$ and base measure H_0 (assumed to be non-atomic). $\mathbf{X}_n$ can be sampled as follows:

Algorithm 1 Recursive coin-flipping for sampling from the DP

```
M = 0                                          ▷ For tracking the number of atoms initialized.
for i = 1: n do                                ▷ Iterate over observations.
    j = 0, coin = 0
    while coin==0 do                           ▷ Recursively (in j) flip V_j-coins until the first heads.
        j = j + 1
        if j > M then                          ▷ Instantiate V_j and Ω_j when necessary.
            V_j ~ Beta(1, θ), Ω_j ~ H_0
            M = M + 1
        end if
        coin ~ Bernoulli(V_j)                  ▷ Flip a V_j-coin.
    end while
    X_i = Ω_j                                  ▷ X_i takes the value of the atom corresponding to the first heads.
end for
return X_n
```

A random number M_n of atoms are generated as needed by the program (equal to M when $\mathbf{X}_n$ is returned). With positive probability M_n is larger than K_n, the number of unique values in $\mathbf{X}_n$.

Sampling the DP by induced size-biased representation. The stick-breaking construction of the DP is distributionally equivalent to the size-biased representation of the DP [26; 28]: $\widetilde{P}_j \overset{\text{d}}{=} V_j \prod_{i=1}^{j-1}(1 - V_i)$ jointly for each j. Hence, the predictive distribution of X_{n+1} given $\widetilde{\mathbf{P}}_{K_n}$ is

$$\mathbb{P}[X_{n+1} \in \bullet \mid \widetilde{\mathbf{P}}_{K_n}] = \textstyle\sum_{j=1}^{K_n} \widetilde{P}_j \delta_{\widetilde{\Omega}_j}(\bullet) + \big(1 - \sum_{j=1}^{K_n} \widetilde{P}_j\big) H_0(\bullet) \, . \tag{2}$$

$\mathbf{X}$ can be sampled from $\mathbf{P}$ by using (2): If X_{n+1} belongs to a new category (which happens with probability $(1 - \sum_{j=1}^{K_n} \widetilde{P}_j)$), then $X_{n+1} = \widetilde{\Omega}_{K_n+1} \sim H_0$, $V_{K_n+1} \sim \text{Beta}(1, \theta)$, and $\widetilde{P}_{K_n+1} = V_{K_n+1} \prod_{j=1}^{K_n}(1 - V_j)$. In this way, only the first K_n atoms of the size-biased representation are generated, corresponding to those chosen by the elements of $\mathbf{X}_n$. Therefore, $K_n \leq M_n$ with probability 1 and $\widetilde{\mathbf{P}}_{K_n}$ is induced by $\mathbf{X}_n$. If the atom weights $\widetilde{\mathbf{P}}_{K_n}$ in (2) are marginalized with respect to the distribution of $\widetilde{\mathbf{P}}_{K_n} \mid \mathbf{X}_n$, another prediction rule, or *urn scheme* can be used to sample X_n; in the case of the DP, this gives rise to the Chinese Restaurant Process (CRP) [28], which can be used in the same way to sample $\mathbf{X}$.

1.1 The laziest initialization

We define a lazy initialization scheme $\widehat{\mathbf{P}}$ for a discrete RPM $\mathbf{P}$ to be *minimal with respect to* $\mathbf{X}$ if, with probability 1 for each $n \in \mathbb{N}_+$, the number of initialized atoms is K_n and the mapping

$\mathbf{X}_n \mapsto (\widetilde{\Omega}_j)_{j=1}^{K_n}$ is surjective; that is, each atom $\widehat{\mathbf{P}}_{K_n}$ corresponds to at least one $X_i \in \mathbf{X}_n$. Using this definition, we give the following result.

Theorem 1. *Let $\mathbf{X}$ be a sequence with elements distributed according to a discrete RPM $\mathbf{P}$. A lazy initialization $\widehat{\mathbf{P}}$ of $\mathbf{P}$ is minimal with respect to $\mathbf{X}$ if and only if $\widehat{\mathbf{P}}$ is a lazy size-biased version of $\mathbf{P}$ induced by $\mathbf{X}$.*

The proof (see Appendix A) follows from a well-known connection between i.i.d. sampling from a discrete RPM and its size-biased representation. However, it seems to have been overlooked in the design and implementation of various PPSs. The distinction is not merely conceptual; a program's representation of a RPM can have substantial influence over computational efficiency, both in prior simulation and in posterior inference.

To illustrate the difference quantitatively, consider the PYP with parameters $\alpha \in (0, 1)$, $\theta > -\alpha$ and continuous base measure H_0. The PYP has heavy tails, which results in size-biased atom weights that decay much slower than those of the DP. The distribution of K_n is known [28]; we derive the distribution of M_n in Appendix B. The following result implies that if the recursive coin-flipping scheme is used to sample from the PYP, not only is it inefficient, but the number of atoms generated is likely to be so large as to prohibit computation. (The proof is given in Appendix B.)

Proposition 1. *Let M_n be the number of atoms instantiated by the recursive coin-flipping scheme to sample $\mathbf{X}_n$. Then $\mathbb{E}_{\alpha,\theta}[M_1] < \infty$ if and only if $\alpha < \frac{1}{2}$. Furthermore, for all $n \geq 1$, if M_n is finite then $\mathbb{E}_{\alpha,\theta}[M_{n+1} \mid M_n] < \infty$ if and only if $\alpha < \frac{1}{2}$.*

Figure 1 shows the expected number of atoms initialized during sampling using the recursive method and the laziest method, which uses the induced size-biased representation; the figure on the right shows that the number of atoms initialized by the recursive coin-flipping explodes for $\alpha \geq \frac{1}{2}$. To our knowledge, analysis of M_n has not appeared previously in the literature.

1.2 Related work

BNP priors. Although the DP and the PYP are the most common priors used in BNP mixture models (due in large part to their stick-breaking representations), a number of other priors have been used. Examples include Gibbs-type priors [10; 3], the class of normalized completely random measures (NCRMs) [31; 17; 6], and its model superclass, Poisson–Kingman models [27; 21]. Both the Gibbs-type and the Poisson–Kingman models are formulated in terms of the size-biased distribution of their atom weights, and as such they are natural candidates for minimally lazy implementation in PPSs.

BNP models in PPSs. Goodman et al. [11] and Roy et al. [33] drew links between BNP models and stochastic programming via a stochastic generalization of memoization. They used the recursive coin-flipping construction to lazily sample from the DP and its hierarchical extensions. Subsequently, several PPSs have provided support for a limited class of BNP models. Edward [37] also uses the recursive construction for simulating from the prior, though posterior inference needs to be truncated to a finite number of atoms. Implementations of the DP and the PYP using the recursive coin-flipping method have been proposed for WebPPL [12; 13]. The back-end implementation of the DP in Anglican [36] and Venture [22] utilize the CRP representation, which is minimally lazy.

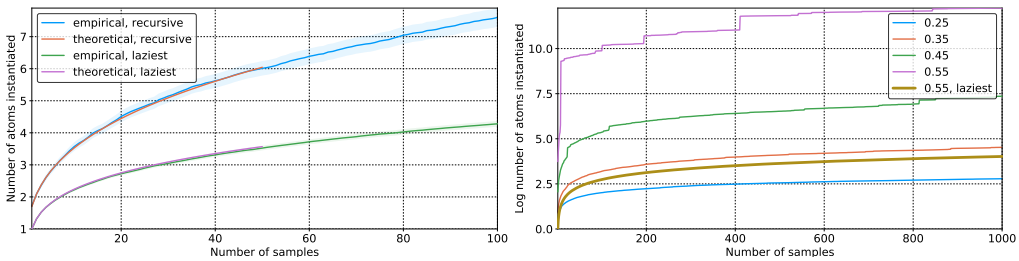

Figure 1: The expected number of atoms generated by recursive coin-flipping and by induced size-biased (laziest) schemes when sampling $\mathbf{X}_n$ from a PYP. Empirical means and standard errors were generated via simulation because theoretical values are numerically unstable for $n > 50$. Left: $\alpha = 0.25$, $\theta = 0.1$. Right: Empirical means (for 4,000 simulations) for $\theta = 0.1$ and various α for the recursive coin-flipping scheme. (Note the log scale on the vertical axis.)

Lazy initialization. The notion of lazy initialization is not new, though it seems to have escaped explicit consideration in the design of PPSs, with the recent exception of Birch [23], which uses a related but different notion called delayed sampling. Tripuraneni et al. [38] use lazy initialization in a particle Gibbs sampler for the HDP-HMM, and lazy initialization is central to the hybrid MCMC sampler of Lomelí et al. [21].

2 Generic particle-based inference methods for RPM mixture models

We consider RPM mixture models, i.e. models whose components $(X_i)_{i=1}^n$ are sampled from a RPM $\mathbf{P}$, itself sampled from a prior μ. Data $(Y_i)_{i=1}^n$ are then observed through an emission distribution $\mathcal{F}$:

$$Y_i \mid X_i \sim \mathcal{F}(\cdot \mid X_i) \quad \text{with} \quad X_i \mid \mathbf{P} \sim \mathbf{P} \quad \text{and} \quad \mathbf{P} \sim \mu \,.$$

A mixture model based on a discrete RPM with an induced size-biased representation can be written as a state space model, for which particle methods like Sequential Monte Carlo (SMC) [19; 4] are well suited. Furthermore, particle methods fit naturally in PPSs as generic inference engines [42; 12; 30]. Algorithm 2 in Appendix C is a general purpose SMC algorithm for these models: One only need to implement the RPM-specific size-biased distribution, and then apply Algorithm 2.

As a simple demonstration of this algorithm, we fit a nonparametric Gaussian mixture model with a common variance among all clusters, but different means, on the galaxy dataset [32], which consists of measurements of velocities of $n = 82$ galaxies from a survey of the Corona Borealis region. We implemented the laziest size-biased representations for certain RPMs, including the PYP, the normalized inverse Gaussian process (NIGP) [18], and the log-beta process [21], and used the generic SMC algorithm implemented in Turing [9], a universal probabilistic programming language written in Julia. Figure 2 shows the posterior predictive distribution of the NIGP mixture model, computed on five sweeps of SMC with 1000 particles.

Related work. SMC algorithms for various BNP mixture models have appeared in the literature: Fearnhead [7] evaluated particle filtering methods for DP mixture models; Wood and Black [41] used SMC to fit a CRP mixture model; particle methods are used for inference on CRP mixture models in the PPS Anglican [42]; Griffin [15] proposed a class of SMC algorithms for NCRMs; Lomelí [20] proposed several novel SMC schemes that use a conjugate prior and marginalize out parameters, and which are suitable for the DP, PYP, Gibbs-type models, and certain NCRMs.

Future directions. Algorithm 2 is a proof of concept and is consequently not competitive compared to inference schemes tailored for RPM mixture models [6; 21]. In our algorithm, cluster assignments, atoms locations, and weights are jointly sampled, which means we have to resample the other variables to update the assignments. Composing inference schemes should yield more efficient posterior samplers. One direction of research is to build a Gibbs sampler composed of Metropolis-Hastings (or Hamiltonian Monte Carlo [24]) updates targeting continuous variables (atom locations, weights, and hyperparameters), while a particle-based [2; 30] update targets the discrete assignments. Such a Gibbs sampler is reminiscent of the hybrid sampler for Poisson–Kingman mixture models [21], but without having to analytically derive conditionals and introduce auxiliary variables. Pseudo-marginal algorithms [1; 2] are also an interesting direction.

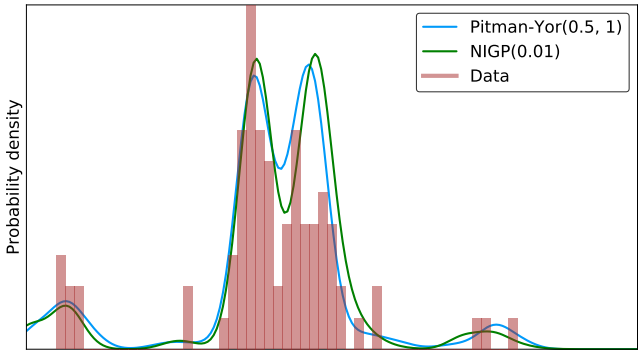

Figure 2: Visualizations of the estimated posterior predictive distribution of the PYP and NIGP mixture models fit to the galaxy data set.

Acknowledgments

BBR, EM, TR, YWT were supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 617071. EM was also supported by Microsoft Research through its PhD Scholarship Programme. AF was supported by an EPSRC Excellence Award. ML, HG and ZG acknowledge support from the Alan Turing Institute (EPSRC Grant EP/N510129/1) and EPSRC Grant EP/N014162/1, and donations from Google and Microsoft Research.

References

[1] Christophe Andrieu and Gareth O Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, pages 697–725, 2009.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2010.

[3] P. De Blasi, S. Favaro, A. Lijoi, R. H. Mena, I. Prünster, and M. Ruggiero. Are Gibbs-type priors the most natural generalization of the Dirichlet process? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):212–229, 02 2015.

[4] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[5] Neil Dhir, Matthijs Vákár, Matthew Wijers, Andrew Markham, Frank Wood, Paul Trethowan, Byron du Preez, Andrew Loveridge, and David Macdonald. Interpreting lion behaviour with nonparametric probabilistic programs. In *UAI*, 2017.

[6] S. Favaro and Y. W. Teh. MCMC for normalized random measure mixture models. *Statistical Science*, 28(3):335–359, 2013.

[7] Paul Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, Jan 2004.

[8] Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *Ann. Statist.*, 1(2): 209–230, 03 1973.

[9] Hong Ge, Kai Xu, Adam Scibior, Zoubin Ghahramani, et al. The Turing language for probabilistic programming. June 2016.

[10] Alexander Gnedin and Jim Pitman. Exchangeable Gibbs partitions and Stirling triangles. *Journal of Mathematical Sciences*, 138(3):5674–5685, 2006. ISSN 1573-8795.

[11] N Goodman, V Mansinghka, D M Roy, K Bonawitz, and J B Tenenbaum. Church: a language for generative models. In *UAI*, pages 220–229, 2008.

[12] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. `http://dippl.org`, 2014. Accessed: 2017-10-30.

[13] Noah D Goodman and Joshua B. Tenenbaum. Probabilistic Models of Cognition. `http://probmods.org/v2`, 2016. Accessed: 2017-10-30.

[14] Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*. ACM, 2014.

[15] J. E. Griffin. Sequential monte carlo methods for mixtures with normalized random measures with independent increments priors. *Statistics and Computing*, 27(1):131–145, Jan 2017.

[16] Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

[17] Lancelot F. James. Bayesian Poisson process partition calculus with an application to Bayesian Lévy moving averages. *Ann. Statist.*, 33(4):1771–1799, 08 2005.

[18] Antonio Lijoi, Ramsés H Mena, and Igor Prünster. Hierarchical mixture modeling with normalized inverse-gaussian priors. *Journal of the American Statistical Association*, 100(472): 1278–1291, 2005.

[19] Jun S Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

[20] María Lomelí. *General Bayesian inference schemes in infinite mixture models*. PhD thesis, University College London, 2017.

[21] María Lomelí, Stefano Favaro, and Yee Whye Teh. A hybrid sampler for Poisson-Kingman mixture models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2161–2169. Curran Associates, Inc., 2015.

[22] Vikash Mansinghka, Daniel Selsam, and Yura Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

[23] Lawrence M. Murray, Daniel Lundén, Jan Kudlicka, David Broman, and Thomas B. Schön. Delayed sampling and automatic Rao-Blackwellization of probabilistic programs. 08 2017. URL `https://arxiv.org/abs/1708.07787`.

[24] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

[25] P. Orbanz and Y. W. Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer, 2010.

[26] Jim Pitman. Random discrete distributions invariant under size-biased permutation. *Advances in Applied Probability*, 28(2):525–539, 1996.

[27] Jim Pitman. Poisson-Kingman partitions. In Darlene R. Goldstein, editor, *Statistics and science: a Festschrift for Terry Speed*, volume 40 of *Lecture Notes–Monograph Series*, pages 1–34. Institute of Mathematical Statistics, 2003.

[28] Jim Pitman. *Combinatorial Stochastic Processes*, volume 1875 of *Ecole d'Eté de Probabilités de Saint-Flour*. Springer-Verlag Berlin Heidelberg, 2006.

[29] Tom Rainforth. *Automating Inference, Learning, and Design using Probabilistic Programming*. PhD thesis, 2017.

[30] Tom Rainforth, Christian A Naesseth, Fredrik Lindsten, Brooks Paige, Jan-Willem van de Meent, Arnaud Doucet, and Frank Wood. Interacting particle Markov chain Monte Carlo. *ICML*, 48, 2016.

[31] Eugenio Regazzini, Antonio Lijoi, and Igor Prünster. Distributional results for means of normalized random measures with independent increments. *Ann. Statist.*, 31(2):560–585, 04 2003.

[32] K Roeder. Density estimation with confidence sets exemplified by super-clusters and voids in the galaxies. *Journal of the American Statistical Association*, 85:617–624, 1990.

[33] DM Roy, VK Mansinghka, ND Goodman, and JB Tenenbaum. A stochastic programming perspective on nonparametric Bayes. In *ICML Workshop on Bayesian Nonparametrics*, 2008.

[34] Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2): 639–650, 1994.

[35] Yee Whye Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer, 2011.

[36] David Tolpin, Jan-Willem van de Meent, and Frank Wood. Probabilistic programming in Anglican. Springer International Publishing, 2015.

[37] Dustin Tran, Matthew D. Hoffman, Rif A. Saurous, Eugene Brevdo, Kevin Murphy, and David M. Blei. Deep probabilistic programming. In *International Conference on Learning Representations*, 2017.

[38] Nilesh Tripuraneni, Shixiang Gu, Hong Ge, and Zoubin Ghahramani. Particle Gibbs for infinite hidden Markov Models. In *NIPS*, pages 2386–2394. Curran Associates, Inc., 2015.

[39] E. T. Whitaker and G. N. Watson. *A Course in Modern Analysis*. Cambridge Mathematical Library. Cambridge University Press, 4th edition, 1996.

[40] David Wingate, Andreas Stuhlmueller, and Noah D Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *AISTATS*, 2011.

[41] Frank Wood and Michael J. Black. A nonparametric Bayesian alternative to spike sorting. *Journal of Neuroscience Methods*, 173(1):1 –12, 2008.

[42] Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, pages 2–46, 2014.

A Proofs

Proof of Theorem 1. Suppose first that $\widehat{\mathbf{P}}$ is a lazy size-biased version of $\mathbf{P}$ induced by $\mathbf{X}$: $(\widehat{P}_1, \widehat{\Omega}_1)$ is equal in distribution to the atom of $\mathbf{P}$ chosen by X_1 according to (1); X_2 chooses the first atom with probability $\widehat{P}_1$, otherwise (with probability $1 - \widehat{P}_1$) a new atom equal in distribution to the second line of (1); and so on, so that $\widehat{\mathbf{P}}_k \stackrel{\mathrm{d}}{=} \widetilde{\mathbf{P}}_k$ for all $k \in \mathbb{N}_+$. Clearly, proceeding in this fashion will produce samples such that $\mathbf{X}_n$ has distribution $\mathbf{P}$ for each $n \in \mathbb{N}_+$, and only the atoms of $\mathbf{P}$ chosen by at least one element of $\mathbf{X}_n$ will be initialized; therefore $\widehat{\mathbf{P}}$ is minimal with respect to $\mathbf{X}$.

Conversely, assume a minimally (with respect to $\mathbf{X}$) lazy initialization $\widehat{\mathbf{P}}'$ is generated by some other process, which implies that the atoms of $\mathbf{P}$ appear in some order. Hence, $(\widehat{P}'_j, \widehat{\Omega}'_j)_{j\geq 1} \stackrel{\mathrm{d}}{=} (P_{\hat{\pi}(j)}, \Omega_{\hat{\pi}(j)})_{j\geq 1}$ for some (possibly random) permutation $\hat{\pi}$. Then under $\widehat{\mathbf{P}}'$,

$$\mathbb{P}[X_1 \in \bullet \mid \mathbf{P}] = \mathbb{P}[\widehat{\Omega}_1 \in \bullet \mid \mathbf{P}] = \mathbb{P}[\Omega_{\hat{\pi}(1)} \in \bullet \mid \mathbf{P}] \,,$$

where the first equality is due to the assumption that $\widehat{\mathbf{P}}'$ is minimal. Denote by $\lambda_{1,j}$ the probability that $\hat{\pi}(1) = j$, given $\mathbf{P}$. Then

$$\mathbb{P}[X_1 \in \bullet \mid \mathbf{P}] = \sum_{j\geq 1} \lambda_{1,j} \delta_{\Omega_j}(\bullet) \,,$$

and X_1 has the correct distribution only if $\lambda_{1,j} = P_j$ for all $j \geq 1$, in which case $\widehat{P}_1 \stackrel{\mathrm{d}}{=} \widetilde{P}_1$. Proceeding this way at all sampling steps that require a new atom to be initialized, we see that only when $\widehat{\mathbf{P}}'$ is a lazy size-biased version of $\mathbf{P}$ will $\mathbf{X}$ have the correct distribution. But this contradicts our assumption that $\widehat{\mathbf{P}}'$ was generated by some other process, and the proof is complete. □

B Analysis of the number of atoms initialized by recursive coin-flipping

We present in this section an analysis of the number of atoms initialized by the recursive coin-flipping scheme using the stick-breaking construction of the DP and PYP. For a sample of size n, denote by M_n the number of atoms instantiated during sampling, i.e., the maximum number of coins flipped during the generation of any X_i.

Proposition 2. *Let M_n be the number of atoms instantiated by the recursive coin-flipping scheme for the PYP with parameters $\alpha \in (0,1)$ and $\theta > -\alpha$. Then*

$$\mathbb{P}_{\alpha,\theta}[M_n \leq m] = \sum_{k=0}^{n} (-1)^k \binom{n}{k} \frac{\Gamma(\theta + m\alpha + k)\Gamma(\theta+1)\Gamma(\frac{\theta}{\alpha} + m)\Gamma(\frac{\theta+k}{\alpha}+1)}{\Gamma(\theta+m\alpha)\Gamma(\theta+1+k)\Gamma(\frac{\theta}{\alpha}+1)\Gamma(\frac{\theta+k}{\alpha}+m)} \quad m = 1, 2, \ldots \,. \tag{3}$$

For the DP ($\alpha = 0$), this simplifies to

$$\mathbb{P}_{0,\theta}[M_n \le m] = \sum_{k=0}^{n} (-1)^k \binom{n}{k} \left(\frac{\theta}{\theta+k}\right)^m \quad \textit{with} \quad \mathbb{E}_{0,\theta}[M_n] = 1 + \theta H_n \ ,$$

where H_n is the nth harmonic number.

Proof. Letting N_i denote the number of coins flipped to generate X_i, observe that

$$\begin{aligned}\mathbb{P}_{\alpha,\theta}[M_n \le m] &= \mathbb{E}[\mathbb{P}[M_n \le m \mid (V_j)_{j\ge 1}]] \\ &= \mathbb{E}[\mathbb{P}[N_1 \le m, \ldots, N_n \le m \mid (V_j)_{j\ge 1}]] = \mathbb{E}\left[\left(1 - \prod_{j=1}^{m}(1-V_j)\right)^n\right] .\end{aligned}$$

Applying the binomial theorem and using the fact that the $V_j \sim \text{Beta}(1-\alpha, \theta+j\alpha)$ are independent,

$$\mathbb{P}_{\alpha,\theta}[M_n \le m] = \sum_{k=0}^{n} (-1)^k \binom{n}{k} \prod_{j=1}^{m} \mathbb{E}[(1-V_j)^k] \tag{4}$$

$$= \sum_{k=0}^{n} (-1)^k \binom{n}{k} \prod_{j=1}^{m} \frac{\Gamma(\theta+j\alpha+k)\Gamma(\theta+1+(j-1)\alpha)}{\Gamma(\theta+j\alpha)\Gamma(\theta+1+(j-1)\alpha+k)} \tag{5}$$

$$= \sum_{k=0}^{n} (-1)^k \binom{n}{k} \frac{\Gamma(\theta+m\alpha+k)\Gamma(\theta+1)\Gamma(\frac{\theta}{\alpha}+m)\Gamma(\frac{\theta+k}{\alpha}+1)}{\Gamma(\theta+m\alpha)\Gamma(\theta+1+k)\Gamma(\frac{\theta}{\alpha}+1)\Gamma(\frac{\theta+k}{\alpha}+m)} , \tag{6}$$

where the last line follows from algebraic manipulations relying on the identity $\Gamma(a+1) = a\Gamma(a)$. When $\alpha = 0$, the second line above easily simplifies to give $\mathbb{P}_{0,\theta}[M_n \le m]$. Finally, using the identity $\mathbb{E}[M] = \sum_{m\ge 1} \mathbb{P}[M \ge m]$ for a non-negative random variable M, we have

$$\begin{aligned}\mathbb{E}_{0,\theta}[M_n] &= \sum_{m=1}^{\infty} \big(1 - \mathbb{P}_{0,\theta}[M_n \le m-1]\big) = \sum_{m=0}^{\infty} \big(1 - \mathbb{P}_{0,\theta}[M_n \le m]\big) \\ &= \sum_{m=0}^{\infty} \left(1 - \sum_{k=0}^{n} (-1)^k \binom{n}{k} \left(\frac{\theta}{\theta+k}\right)^m\right) = -\sum_{k=1}^{n} (-1)^k \binom{n}{k} \sum_{m=0}^{\infty} \left(\frac{\theta}{\theta+k}\right)^m \\ &= -\sum_{k=1}^{n} (-1)^k \binom{n}{k} \frac{\theta+k}{k} = 1 + \theta \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} \frac{1}{k} = 1 + \theta H_n \ ,\end{aligned}$$

where the last equality follows from Euler's integral representation of the harmonic numbers. □

The following result is useful for computing the probabilities in (3).

Proposition 3. *Denote the cumulative probabilities in* (3) *as $P^{\alpha,\theta}_{n,m}$. Then for $m \ge 1$ and $n \ge 2$, $P^{\alpha,\theta}_{n,m}$ satisfies the recursion*

$$P^{\alpha,\theta}_{n,m} = P^{\alpha,\theta}_{n-1,m} - P^{\alpha,\theta+1}_{n-1,m}(1 - P^{\alpha,\theta}_{1,m}) \quad \textit{with} \quad P^{\alpha,\theta}_{1,m} = 1 - \prod_{j=1}^{m} \frac{\theta+j\alpha}{\theta+1+(j-1)\alpha} \ .$$

Proof. From (5),

$$P^{\alpha,\theta}_{n,m} = \sum_{k=0}^{n} (-1)^k \binom{n}{k} \prod_{j=1}^{m} \frac{(\theta+j\alpha)_k}{(\theta+1+(j-1)\alpha)_k} := \sum_{k=0}^{n} (-1)^k \binom{n}{k} W^{\alpha,\theta}_{k,m} \ ,$$

where $(a)_n = a(a+1)\cdots(a+n-1)$ is the rising factorial and $W^{\alpha,\theta}_{k,m}$ is introduced for notational convenience. Noting that $(a)_n = a(a+1)_{n-1}$, which implies that $W^{\alpha,\theta}_{k,m} = W^{\alpha,\theta}_{1,m} W^{\alpha,\theta+1}_{k-1,m}$, and using

the identities $\binom{n}{k} = \frac{n}{n-k}\binom{n-1}{k}$ and $\binom{n-1}{k-1} = \frac{k}{n-k}\binom{n-1}{k}$,

$$\begin{aligned}
P_{n,m}^{\alpha,\theta} &= P_{n-1,m}^{\alpha,\theta} + \sum_{k=0}^{n}(-1)^k\binom{n}{k}W_{k,m}^{\alpha,\theta} - \sum_{k=0}^{n-1}(-1)^k\binom{n-1}{k}W_{k,m}^{\alpha,\theta} \\
&= P_{n-1,m}^{\alpha,\theta} + \sum_{k=1}^{n}(-1)^k\binom{n-1}{k}\left(\frac{n}{n-k}-1\right)W_{k,m}^{\alpha,\theta} \\
&= P_{n-1,m}^{\alpha,\theta} - \sum_{k=1}^{n}(-1)^{k-1}\binom{n-1}{k-1}W_{1,m}^{\alpha,\theta}W_{k-1,m}^{\alpha,\theta+1} \\
&= P_{n-1,m}^{\alpha,\theta} - W_{1,m}^{\alpha,\theta}\sum_{k=0}^{n-1}(-1)^k\binom{n-1}{k}W_{k,m}^{\alpha,\theta+1} \\
&= P_{n-1,m}^{\alpha,\theta} - W_{1,m}^{\alpha,\theta}P_{n-1,m}^{\alpha,\theta+1}\,.
\end{aligned}$$

Noting that $P_{1,m}^{\alpha,\theta} = 1 - W_{1,m}^{\alpha,\theta}$, the result follows. □

Proof of Proposition 1. Using (5) and again that $\mathbb{E}[M] = \sum_{m\geq 1}\mathbb{P}[M \geq m]$ for a non-negative random variable M,

$$\mathbb{E}_{\alpha,\theta}[M_1] = \sum_{m=0}^{\infty}\prod_{j=1}^{m}\frac{\theta+j\alpha}{\theta+1+(j-1)\alpha} = \sum_{m=0}^{\infty}\frac{(\frac{\theta}{\alpha}+1)_m}{(\frac{\theta+1}{\alpha})_m} = {}_2F_1(1,\tfrac{\theta}{\alpha}+1;\tfrac{\theta+1}{\alpha};1)\,,$$

where ${}_2F_1(a,b;c;z)$ is the ordinary hypergeometric function, which converges for $c-b-a>0$, corresponding to $\alpha < \frac{1}{2}$; it diverges if $\alpha \geq \frac{1}{2}$. If $\alpha < \frac{1}{2}$, then $\mathbb{E}_{\alpha,\theta}[M_1] = \frac{\theta+1+\alpha}{1-2\alpha}$, which follows from an identity [39, §14.11].

Similarly,

$$\begin{aligned}
\mathbb{E}_{\alpha,\theta}[M_{n+1} \mid M_n] &= M_n + \sum_{m=0}^{\infty}\prod_{j=1}^{M_n+m}\frac{\theta+j\alpha}{\theta+1+(j-1)\alpha} \\
&= M_n + \left[\prod_{j=1}^{M_n}\frac{\theta+j\alpha}{\theta+1+(j-1)\alpha}\right]\sum_{m=0}^{\infty}\prod_{j=1}^{m}\frac{\frac{\theta}{\alpha}+M_n+j}{\frac{\theta+1}{\alpha}+M_n-1+j} \\
&= M_n + \left[\prod_{j=1}^{M_n}\frac{\theta+j\alpha}{\theta+1+(j-1)\alpha}\right]\sum_{m=0}^{\infty}\frac{(\frac{\theta}{\alpha}+M_n+1)_m}{(\frac{\theta+1}{\alpha}+M_n)_m} \\
&= M_n + \left[\prod_{j=1}^{M_n}\frac{\theta+j\alpha}{\theta+1+(j-1)\alpha}\right]{}_2F_1(1,\tfrac{\theta}{\alpha}+M_n+1;\tfrac{\theta+1}{\alpha}+M_n;1)\,.
\end{aligned}$$

As before, this converges only when $\alpha < \frac{1}{2}$, in which case

$$\mathbb{E}_{\alpha,\theta}[M_{n+1} \mid M_n] = M_n + \left[\prod_{j=1}^{M_n}\frac{\theta+j\alpha}{\theta+1+(j-1)\alpha}\right]\frac{\theta+1+(M_n-1)\alpha}{1-2\alpha}\,.$$

□

C SMC for minimally lazy RPM mixture models.

Algorithm 2 Sequential Monte Carlo for RPM mixture models

Initialize $S_1^l = \{\},\ K^l = 0 \quad \forall l = 1, \ldots, L$
for $i = 1 \colon n$ **do** ▷ Iterate over observations.
  for $l = 1 \colon L$ **do** ▷ Iterate over particles.
    $z_i^l \sim \text{Cat}(\widetilde{P}_1^l, \ldots, \widetilde{P}_{K^l}^l, (1 - \sum_{j=1}^{K^l} \widetilde{P}_j^l))$ ▷ Sample atom assignment.
    if $z_i^l > K^l$ **then** ▷ Create a new atom.
      $K^l = K^l + 1$ ▷ Update number of atoms.
      $\widetilde{P}_{K^l} \sim \textbf{SBS}(\widetilde{P}_1^l, \ldots, \widetilde{P}_{K^l-1}^l)$ ▷ Lazy size-biased sampling (specific to the RPM).
      $\widetilde{\Omega}_{K^l} \sim H_0$ ▷ Sample new atom from the base measure.
      $S_i^l = \left(S_{i-1}^l, \widetilde{\Omega}_{K^l}, \widetilde{P}_{K^l}, z_i^l\right)$ ▷ Update state.
    else ▷ Use an existing atom.
      $S_i^l = \left(S_{i-1}^l, z_i^l\right)$ ▷ Update state.
    end if
    $w_i^l = f(y_i^l \mid \widetilde{\Omega}_{z_i^l})$ ▷ Compute weight – f is the density of $\mathcal{F}$.
  end for
  for $l = 1 \colon L$ **do** ▷ Resample particles
    $a_i^l \sim \mathcal{A}(\cdot \mid \mathbf{w}_i),\ S_i^l = S_i^{a_i^l}$ ▷ Sample new ancestor.
  end for
end for
return $\left(S_n^l\right)_{l=1}^L$ ▷ Return samples from the L particles.