BNPmix: an R Package for Bayesian Non-Parametric Modeling via Pitman-Yor Mixtures

Riccardo Corradin
University of Milano-Bicocca

Antonio Canale
University of Padova

Bernardo Nipoti
University of Milano-Bicocca

Abstract

BNPmix is an R package for Bayesian non-parametric multivariate density estimation, clustering, and regression, using Pitman-Yor mixture models, a flexible and robust generalization of the popular class of Dirichlet process mixture models. A variety of model specifications and state-of-the-art posterior samplers are implemented. In order to achieve computational efficiency, all sampling methods are written in C++ and seamless integrated into R by means of the **Rcpp** and **RcppArmadillo** packages. **BNPmix** exploits the **ggplot2** capabilities and implements a series of generic functions to plot and print summaries of posterior densities and induced clustering of the data.

Keywords: Bayesian non-parametric mixture, C++, multivariate density estimation, clustering, importance conditional sampler, slice sampler, marginal sampler.

1. Introduction

Bayesian non-parametric (BNP) methods provide flexible solutions to complex problems and data which are not easily described by parametric models (Hjort, Holmes, Müller, and Walker 2010; Müller, Quintana, Jara, and Hanson 2015). A remarkable example is represented by BNP mixtures, flexible models for density estimation and clustering, nowadays a well-established modeling option in the toolbox of statisticians and applied researchers. This line of research was initiated by the Dirichlet process (DP) (Ferguson 1973) mixture of Gaussian kernels by Lo (1984), a contribution which paved the way to the definition of a rich variety of non-parametric mixture models. More recently, increasing interest has been dedicated to mixture models based on non-parametric mixing random probability measures, other than the DP, which might provide increased modeling flexibility (e.g., Nieto-Barajas, Prünster, and Walker 2004; Lijoi, Mena, and Prünster 2005b,a, 2007; Argiento, Bianchini, and Guglielmi 2016).

A BNP mixture model for the observations $\boldsymbol{Y}^{(n)} = (\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n)$, with $\boldsymbol{Y}_i \in \mathbb{R}^p$, is defined by the random distribution

$$\tilde{f}(\boldsymbol{y}) = \int_{\Theta} \mathcal{K}(\boldsymbol{y}; \theta) \mathrm{d}\tilde{p}(\theta), \tag{1}$$

where Θ is the parameter space, $\mathcal{K}(\cdot;\cdot)$ is a suitable kernel defined on $\mathbb{R}^p \times \Theta$, and $\tilde{p}$ is a discrete random probability measure on Θ. Assuming that $\tilde{p}$ is distributed as a Pitman-Yor process (PY) (Perman, Pitman, and Yor 1992; Pitman 1995; Pitman and Yor 1997) leads to a model which stands out for being a good compromise between modeling flexibility, and

mathematical and computational tractability. Based on such assumption, model (1) can be alternatively written in hierarchical form as

$$\begin{aligned} \boldsymbol{Y}_i \mid \theta_i &\overset{\text{ind}}{\sim} \mathcal{K}(\boldsymbol{Y}_i; \theta_i), \qquad i = 1, \ldots, n \\ \theta_i \mid \tilde{p} &\overset{\text{iid}}{\sim} \tilde{p}, \\ \tilde{p} &\sim PY(\alpha, \vartheta; P_0), \end{aligned} \tag{2}$$

where $PY(\alpha, \vartheta; P_0)$ denotes a PY process with discount parameter $\alpha \in [0, 1)$, strength parameter $\vartheta > -\alpha$, and base probability measure P_0 defined on Θ. By exploiting the so-called stick-breaking representation (Sethuraman 1994; Pitman and Yor 1997) of the PY process, the random density $\tilde{f}$ can be equivalently written as an infinite sum, specifically

$$\tilde{f}(\boldsymbol{y}) = \sum_{j=1}^{\infty} \pi_j \mathcal{K}(\boldsymbol{y}; \tilde{\theta}_j), \tag{3}$$

with $\tilde{\theta}_j \overset{\text{iid}}{\sim} P_0$ and $\pi_j = V_j \prod_{l<j}(1 - V_l)$ with $V_j \overset{\text{ind}}{\sim} \text{Beta}(1 - \alpha, \vartheta + j\alpha)$. The PY process (with $\alpha > 0$) and the DP, special case recovered when $\alpha = 0$, are characterized by two radically different learning mechanisms (see De Blasi, Favaro, Lijoi, Mena, Prünster, and Ruggiero 2015). The discount parameter plays an important modeling role and has an impact on the induced prior distribution on the number of clusters in the data, the larger being α the flatter and less informative the prior. This, in the context of mixture models, makes the PY more robust in estimating the clustering structure underlying the data.

While open source software implementing Markov chain Monte Carlo (MCMC) methods for some classes of BNP mixture models is already available, the **BNPmix** package, described in this paper, aims at filling an existing gap by providing reliable routines which make posterior inference based on PY mixtures possible, straightforward and efficient. The **BNPmix** package implements different specifications of model (2) including both univariate and multivariate kernels. The inclusion of concomitant independent variables is also accounted for, by considering mixture models for partially exchangeable data with stratification induced by a categorical covariate (see Foti and Williamson 2015, and references therein), and mixture models for regression problems (in the spirit of De Iorio, Müller, Rosner, and MacEachern 2004). Moreover, while there is consensus that MCMC methods represent the gold standard for carrying out posterior inference for BNP mixtures, it is known that different MCMC simulation schemes feature different properties and thus might be convenient to serve different purposes. With this in mind, **BNPmix** implements three state-of-the-art MCMC methods for PY mixture models, henceforth referred to as marginal sampler, slice sampler and importance conditional sampler, providing the user with the option to choose one.

The list of R (**?**) packages implementing MCMC techniques for BNP models is rich. In order to clarify which features are specific to **BNPmix** and which are shared by other packages, we review state-of-the-art R packages for BNP inference via MCMC. To this end, a list of the models implemented in **BNPmix** is presented in Table 3 in Appendix A.1, along with the availability of the same models in other packages. Similarly, Table 4 in Appendix A.1 compares the main technical features of **BNPmix** with those of other packages. While these tables focus on the features of **BNPmix**, it is worth stressing that the packages considered for the comparison also implement other models, which we review concisely. The **DPpackage** by Jara, Hanson, Quintana, Müller, and Rosner (2011) is probably the most comprehensive of the packages we considered. It is mainly written in Fortran and consists of a rich

collection of functions implementing some of the most successful Bayesian non-parametric and semi-parametric models, including DP and dependent Dirichlet process (DDP) mixtures, hierarchical DP, Pólya trees, and random Bernstein polynomials. Regrettably, despite being widely used, the **DPpackage** was recently archived from the Comprehensive R Archive Network. More recent and, in some case, more specific R packages for BNP inference via mixture models are **PReMiuM** (Liverani, Hastie, Azizi, Papathomas, and Richardson 2015), **BNPdensity** (Barrios, Lijoi, Nieto-Barajas, Prünster, and Kon Kam King 2017), **dirichletprocess** (Ross and Markwick 2020), **BNPMIXcluster** (Carmona, Nieto-Barajas, and Canale 2017) and **msBP** (Canale 2017). The main focus of **PReMiuM** is the definition of non-parametric regression models linking an outcome (continuous, binary or discrete) to a set of covariates via dependent non-parametric mixtures. The package implements a rich set of functions to explore the output of the analysis, including the induced clustering structure and the role of each covariate in driving the final model specification. Models are implemented for the DP case and extended to the PY process only by means of approximation. Another interesting feature of **PReMiuM** is the implementation of a label switching move (Liverani *et al.* 2015) to improve mixing of MCMC samplers. **BNPdensity** implements a Ferguson and Klass algorithm (Ferguson and Klass 1972; Barrios, Lijoi, Nieto-Barajas, and Prünster 2013) for a large class of univariate mixture models, with the mixing random probability assumed distributed as a normalized random measure (Regazzini, Lijoi, and Prünster 2003). The package offers the choice of five different kernels (Gaussian, double exponential, gamma, lognormal and beta) and accounts for the presence of censored data. The **dirichletprocess** package provides a set of functions allowing users to implement their own DP mixture model: a new object class is introduced, which acts as building block for a variety of specific statistical models. **BNPMIXcluster** focuses exclusively on the case of multivariate mixed-scale data (Canale and Dunson 2011; Carmona, Nieto-Barajas, and Canale 2019), framework for which a marginal MCMC sampler for PY mixture models is implemented. The package **msBP** only implements the multiscale Bernstein polynomial mixture model of Canale and Dunson (2014). The described set of R packages, overall, offers the possibility of carrying out statistical inference by means of a very broad collection of BNP mixture models. At the same time, when the focus is on the use of the PY process, **BNPmix** plays a leading role. Finally, it is worth mentioning the increasing attention recently dedicated by the BNP literature to variational methods approximating the posterior distribution (Blei and Jordan 2006; Hughes, Kim, and Sudderth 2015; Campbell, Straub, Fisher III, and How 2015; Tank, Foti, and Fox 2015): the availability of R packages implementing such approach for BNP models is rather limited though, a notable exception being the package **MixDir** (Ahlmann-Eltze and Yau 2018) which implements a hierarchical DP mixture of multinomial kernels.

The rest of the paper is organized as follows. Section 2 describes the model specifications accounted for in **BNPmix**. Section 3 introduces the implemented state-of-the-art MCMC methods for posterior simulation. An overview of the design philosophy of the package is presented in Section 4. In Section 5 the main features and usage of the package are illustrated by means of illustrative analyses of both synthetic and real data. A further comparison with other R packages for BNP inference and technical details on the parametrization of the implemented models are provided in the Appendix.

2. Model specifications

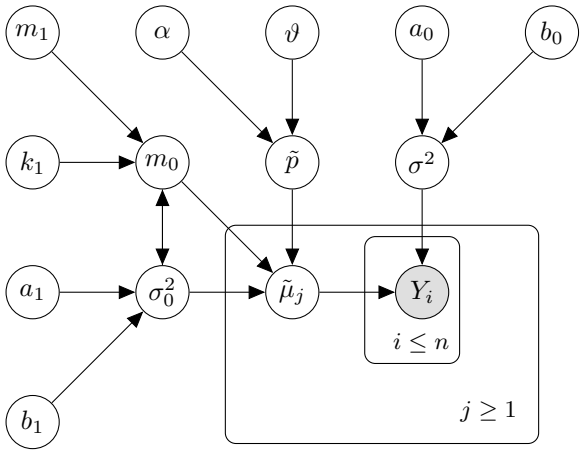

Figure 1: Hierarchical representation of the univariate location PY mixture model with Gaussian kernel.

In this section we briefly introduce the different PY mixture models implemented in the **BNPmix** package. The implemented models can be classified into two main classes, based on whether they account for individual covariates or not. The latter ones can be obtained by suitably specifying parametric space, kernel and base measure in (2). The first ones can be seen as generalizations of (2) to the context of regression problems, and to the analysis of correlated samples by means of DDP.

The focus of the **BNPmix** package is on models obtained by mixing univariate, multivariate Gaussian kernels, or univariate normal regression kernels, specifications which guarantee that the resulting mixture models are dense in the space of regular densities (Lo 1984). In addition, the parametric space and the base measure can be chosen so to impose constraints on the variance and, in the multivariate case, the covariance structure of the model. The resulting mixture models are discussed next in more detail. For the sake of clarity, the parametric forms of the distributions introduced along the paper are reported in Appendix A.2.

Univariate location PY mixture model

We consider a univariate Gaussian kernel and define a mixture on its location parameter (West 1991). That is, we set $\theta = \mu$, and thus $\Theta = \mathbb{R}$, and consider the kernel $\mathcal{K}(y;\theta) = \phi(y;\mu,\sigma^2)$, where $\phi(\cdot;\mu,\sigma^2)$ denotes the probability density function of a normal random variable with mean μ and variance σ^2. Following this specification, the random density in (3) becomes

$$\tilde{f}(y) = \sum_{j=1}^{\infty} \pi_j \phi(y;\tilde{\mu}_j,\sigma^2).$$

In order to achieve conjugacy between kernel and base measure, we assume P_0 is normal, namely $\tilde{\mu}_j \overset{\text{iid}}{\sim} N(m_0,\sigma_0^2)$, and we assign σ^2 an inverse gamma prior, that is $\sigma^2 \sim \text{IGa}(a_0,b_0)$. The model can be completed by specifying a normal-inverse gamma hyperprior on (m_0,σ_0^2), which is tantamount to assume $\sigma_0^2 \sim \text{IGa}(a_1,b_1)$ and $m_0 \mid \sigma_0^2 \sim \text{N}(m_1,\sigma_0^2/k_1)$. Figure 1 displays a graphical representation of the model specification for the univariate location PY mixture model.

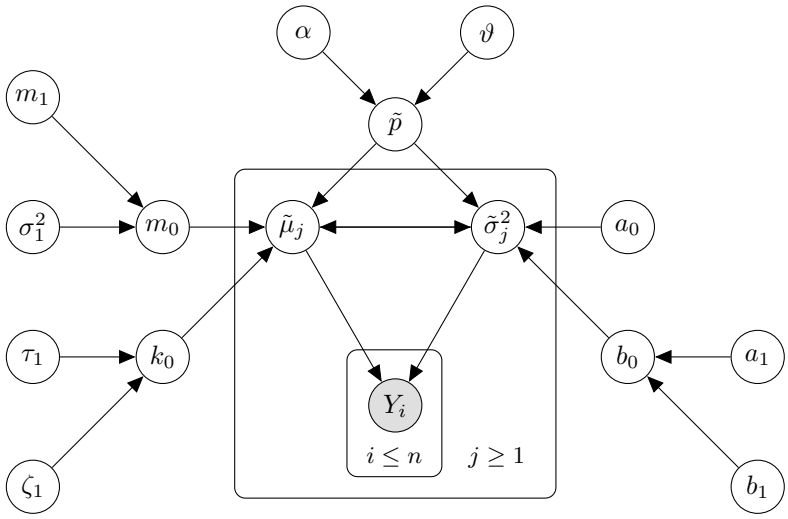

Figure 2: Hierarchical representation of the univariate PY process mixture model with location-scale base measure.

Univariate location-scale PY mixture model

As an extension of the previous model specification, we consider a univariate Gaussian kernel and define a mixture on the vector (μ, σ^2) composed by location and scale parameters of the kernel (Escobar and West 1995). That is, we set $\theta = (\mu, \sigma^2)$, and thus $\Theta = \mathbb{R} \times \mathbb{R}^+$, and consider the kernel $\mathcal{K}(y; \theta) = \phi(y; \mu, \sigma^2)$. Following this specification, the random density in (3) becomes

$$\tilde{f}(y) = \sum_{j=1}^{\infty} \pi_j \phi(y; \tilde{\mu}_j, \tilde{\sigma}_j^2).$$

In order to achieve conjugacy, we consider a normal-inverse gamma base measure P_0, namely $\tilde{\sigma}_j^2 \overset{\text{iid}}{\sim} \text{IGa}(a_0, b_0)$ and $\tilde{\mu}_j \mid \tilde{\sigma}_j^2 \overset{\text{ind}}{\sim} \text{N}(m_0, \tilde{\sigma}_j^2/k_0)$. The model can be completed by assigning independent hyperpriors to the main parameters of the base measure. Specifically, $m_0 \sim \text{N}(m_1, \sigma_1^2)$, $k_0 \sim \text{Ga}(\tau_1, \zeta_1)$, and $b_0 \sim \text{Ga}(a_1, b_1)$. Figure 2 displays a hierarchical representation of the univariate location-scale PY mixture model.

Multivariate location PY mixture model

We consider a p-dimensional multivariate Gaussian kernel and define a mixture on its mean vector $\boldsymbol{\mu}$ (Shen, Tokdar, and Ghosal 2013). That is we set $\theta = \boldsymbol{\mu}$, and thus $\Theta = \mathbb{R}^p$, and $\mathcal{K}(\boldsymbol{y}; \theta) = \phi_p(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\phi_p(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of a p-dimensional normal random vector with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, with $\boldsymbol{\Sigma}$ symmetric ad positive semi-definite. Following this specification, the random density in (3) becomes

$$\tilde{f}(\boldsymbol{y}) = \sum_{j=1}^{\infty} \pi_j \phi_p(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_j, \boldsymbol{\Sigma}).$$

In order to achieve conjugacy, we consider a multivariate normal base measure P_0, that is $\tilde{\boldsymbol{\mu}}_j \overset{\text{iid}}{\sim} \text{N}_p(\boldsymbol{m}_0, \boldsymbol{S}_0)$, and we endow $\boldsymbol{\Sigma}$ with an inverse Wishart prior, namely $\boldsymbol{\Sigma} \sim \text{IW}(\nu_0, \boldsymbol{\Sigma}_0)$. The model specification can be completed by assigning a normal-inverse Wishart hyperprior

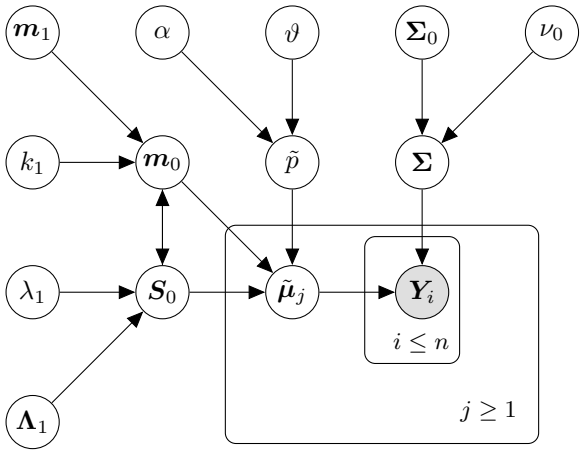

Figure 3: Hierarchical representation of the multivariate PY process mixture model with location base measure.

to $(\boldsymbol{m}_0, \boldsymbol{S}_0)$, that is by assuming $\boldsymbol{S}_0 \sim \text{IW}(\lambda_1, \boldsymbol{\Lambda}_1)$ and $\boldsymbol{m}_0 \mid \boldsymbol{S}_0 \sim \text{N}_p(\boldsymbol{m}_1, \boldsymbol{S}_0/k_1)$. Figure 3 displays a hierarchical representation of the multivariate location PY mixture model.

Multivariate location-scale PY mixture model: full covariance matrix

As an extension of the previous specification, we consider a p-dimensional multivariate Gaussian kernel and define a mixture jointly on the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ (Escobar and West 1995; Müller, Erkanli, and West 1996). That is, we set $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and thus $\mathbb{R}^p \times S_+^p$, where S_+^p denotes the space of $p \times p$ positive semi-definite matrices, and $\mathcal{K}(\boldsymbol{y}; \theta) = \phi_p(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Following this specification, the random density in (3) becomes

$$\tilde{f}(\boldsymbol{y}) = \sum_{j=1}^{\infty} \pi_j \phi_p(\boldsymbol{y}; \tilde{\boldsymbol{\mu}}_j, \tilde{\boldsymbol{\Sigma}}_j).$$

In order to achieve conjugacy, we specify a normal-inverse Wishart base measure P_0, namely $\tilde{\boldsymbol{\Sigma}}_j \overset{\text{iid}}{\sim} \text{IW}(\nu_0, \boldsymbol{\Sigma}_0)$ and $\tilde{\boldsymbol{\mu}}_j \mid \tilde{\boldsymbol{\Sigma}}_j \overset{\text{ind}}{\sim} \text{N}_p(\boldsymbol{m}_0, \tilde{\boldsymbol{\Sigma}}_j/k_0)$. The model can be completed by assuming independent hyperpriors on $\boldsymbol{m}_0$, k_0 and $\boldsymbol{\Sigma}_0$, namely $\boldsymbol{m}_0 \sim \text{N}_p(\boldsymbol{m}_1, \boldsymbol{S}_1)$, $k_0 \sim \text{Ga}(\tau_1, \zeta_1)$ and $\boldsymbol{\Sigma}_0 \sim \text{W}(\nu_1, \boldsymbol{\Sigma}_1)$. Figure 4 displays a hierarchical representation of the multivariate location-scale PY mixture model with full covariance matrix.

Multivariate location-scale PY mixture model: diagonal covariance matrix

A more parsimonious version of the multivariate location-scale PY mixture model presented in the previous section is obtained by constraining the random covariance matrices $\tilde{\boldsymbol{\Sigma}}_j$ to be diagonal (see, e.g., Bouveyron and Brunet-Saumard 2014). This model specification is particularly convenient when p is large. Based on this assumption, the random density in (3) becomes

$$\tilde{f}(\boldsymbol{y}) = \sum_{j=1}^{\infty} \pi_j \prod_{r=1}^{p} \phi(y_r; \tilde{\mu}_{jr}, \tilde{\sigma}_{jr}^2),$$

where y_r is the r-th component of $\boldsymbol{y}$, $\tilde{\boldsymbol{\mu}}_j = (\tilde{\mu}_{j1}, \ldots, \tilde{\mu}_{jp})$ and $\tilde{\boldsymbol{\Sigma}}_j$ is a diagonal matrix with diagonal $\tilde{\boldsymbol{\sigma}}_j^2 = (\tilde{\sigma}_{j1}^2, \ldots, \tilde{\sigma}_{jp}^2)$. Conditionally on $\tilde{p}$, the marginal model specification for each com-

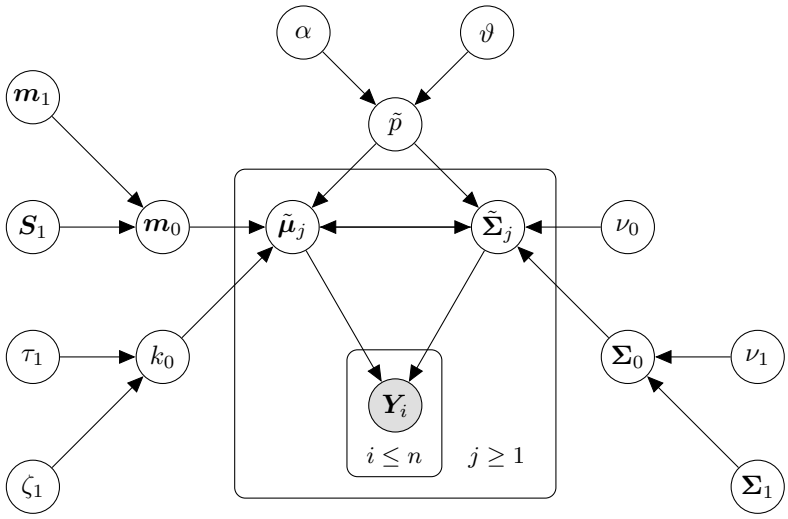

Figure 4: Hierarchical representation of the multivariate PY process mixture model with location-scale base measure and full covariance matrix.

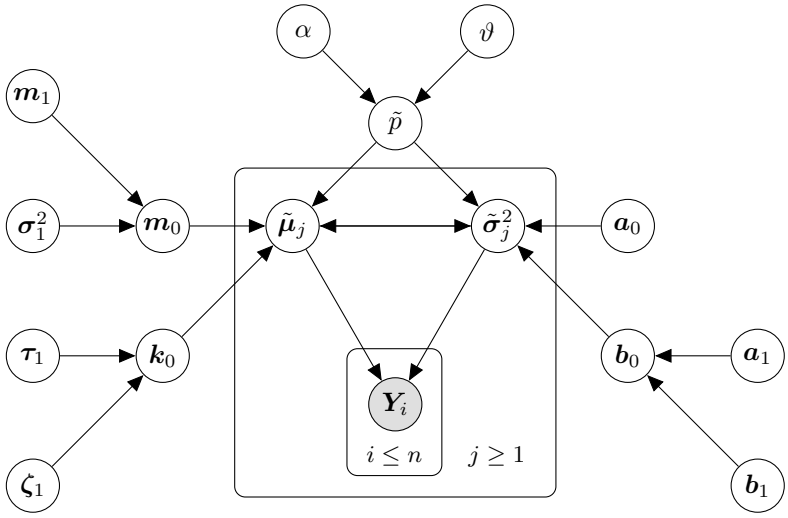

Figure 5: Hierarchical representation of the multivariate PY process mixture model with location-scale base measure and diagonal covariance matrix.

ponent is equivalent to the univariate location-scale case. Specifically, the base measure P_0 is the product of p independent normal-inverse gamma distributions, that is, $\tilde{\sigma}^2_{jr} \overset{\text{iid}}{\sim} \text{IGa}(a_{0r}, b_{0r})$ and $\tilde{\mu}_{jr} \mid \tilde{\sigma}^2_{jr} \overset{\text{ind}}{\sim} \text{N}(m_{0r}, \tilde{\sigma}^2_{jr}/k_{0r})$, for every $r = 1, \ldots, p$. The model specification can be completed by assigning independent hyperpriors to the components of $\boldsymbol{b}_0 = (b_{01}, \ldots, b_{0p})$, $\boldsymbol{m}_0 = (m_{01}, \ldots, m_{0p})$ and $\boldsymbol{k}_0 = (k_{01}, \ldots, k_{0p})$, namely $m_{0r} \sim \text{N}(m_{1r}, \sigma^2_{1r})$, $b_{0r} \sim \text{Ga}(a_{1r}, b_{1r})$ and $k_{0r} \sim \text{Ga}(\tau_{1r}, \tau_{2r})$, for every $r = 1, \ldots, p$. Figure 5 displays a hierarchical representation of the multivariate location-scale PY mixture model with diagonal covariance matrix.

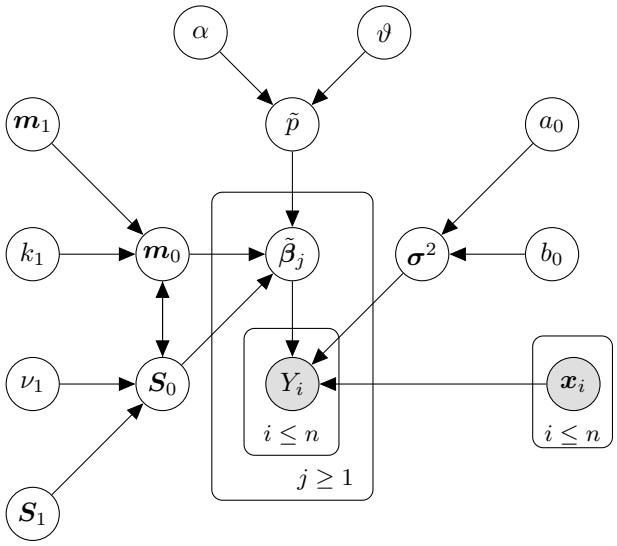

Figure 6: Hierarchical representation of the univariate regression PY mixture model.

Univariate regression PY mixture model

We consider an infinite mixture of regression model accounting for a d-dimensional vector of independent variables $\boldsymbol{x}$ (Dunson, Pillai, and Park 2007). This is obtained by considering a univariate Gaussian kernel with regressors acting linearly on the location parameter, and by defining a mixture on the regression coefficients $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_d)$, with the scale parameter of the kernel common across different groups. That is, we define $\boldsymbol{x}_0 = (1, \boldsymbol{x}^\top)^\top$, we set $\theta = \boldsymbol{\beta}$, and thus $\Theta = \mathbb{R}^{d+1}$, and consider the kernel $\mathcal{K}(y; \theta) = \phi(y; \boldsymbol{x}_0^\top \boldsymbol{\beta}, \sigma^2)$. Following this specification, the random density in (3) becomes

$$\tilde{f}(y \mid \boldsymbol{x}) = \sum_{j=1}^{\infty} \pi_j \phi(y; \boldsymbol{x}_0^\top \tilde{\boldsymbol{\beta}}_j, \sigma^2).$$

In order to achieve conjugacy, we assume that the base measure P_0 is a multivariate normal distribution, that is $\tilde{\boldsymbol{\beta}}_j \overset{\text{iid}}{\sim} \text{N}_{d+1}(\boldsymbol{m}_0, \boldsymbol{S}_0)$, with $j \geq 1$. We set an inverse gamma distribution for the common scale parameter, $\sigma^2 \sim \text{IGa}(a_0, b_0)$. The model can be completed by endowing $(\boldsymbol{m}_0, \boldsymbol{S}_0)$ with a normal-inverse Wishart hyperprior, that is $\boldsymbol{S}_0 \sim \text{IW}(\nu_1, \boldsymbol{S}_1)$ and $\boldsymbol{m}_0 \mid \boldsymbol{S}_0 \sim \text{N}_p(\boldsymbol{m}_1, \boldsymbol{S}_0/k_1)$. Figure 6 displays a hierarchical representation of the univariate regression-scale PY mixture model.

Univariate regression-scale PY mixture model

As an extension of the previous specification, we consider an infinite mixture of regression model accounting for a d-dimensional vector of independent variables $\boldsymbol{x}$, obtained by jointly defining a mixture on the regression coefficients and the scale parameter of the univariate Gaussian kernel (Dunson *et al.* 2007). That is, we set $\theta = (\boldsymbol{\beta}, \sigma^2)$, and thus $\Theta = \mathbb{R}^{d+1} \times \mathbb{R}^+$, and consider the kernel $\mathcal{K}(y; \theta) = \phi(y; \boldsymbol{x}_0^\top \boldsymbol{\beta}, \sigma^2)$. Following this specification, the random density in (3) becomes

$$\tilde{f}(y \mid \boldsymbol{x}) = \sum_{j=1}^{\infty} \pi_j \phi(y; \boldsymbol{x}_0^\top \tilde{\boldsymbol{\beta}}_j, \tilde{\sigma}_j^2).$$

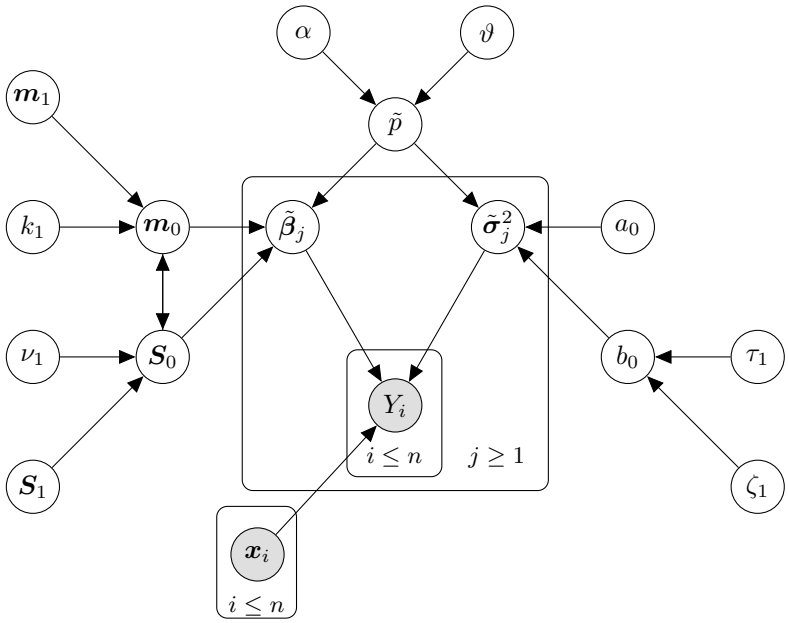

Figure 7: Hierarchical representation of the univariate regression-scale PY mixture model.

In order to achieve conjugacy, the base measure P_0 is set equal to a product of a multivariate normal distribution and an inverse gamma distribution, that is $\tilde{\boldsymbol{\beta}}_j \overset{\text{iid}}{\sim} \text{N}_{d+1}(\boldsymbol{m}_0, \boldsymbol{S}_0)$ and $\tilde{\sigma}_j^2 \overset{\text{iid}}{\sim} \text{IGa}(a_0, b_0)$. The model can be completed by endowing $(\boldsymbol{m}_0, \boldsymbol{S}_0)$ with a normal-inverse Wishart hyperprior, that is $\boldsymbol{S}_0 \sim \text{IW}(\nu_1, \boldsymbol{S}_1)$ and $\boldsymbol{m}_0 \mid \boldsymbol{S}_0 \sim \text{N}_d(\boldsymbol{m}_1, \boldsymbol{S}_0/k_1)$, and by assuming $b_0 \sim \text{Ga}(\tau_1, \zeta_1)$. Figure 7 displays a hierarchical representation of the univariate regression-scale PY mixture model.

Univariate location-scale DDP mixture model

We assume each observation y_i is endowed with a categorical covariate x_i taking values in $\{1, \ldots, L\}$, which allows observations to be gathered into L distinct groups $(y_{l1}, \ldots, y_{ln_l})$, with $l = 1, \ldots, L$. In order to account for heterogeneity across groups, while allowing borrowing of information, we consider the partially exchangeable mixture model proposed by Lijoi, Nipoti, and Prünster (2014). Namely, each group is modeled by means of a mixture with Gaussian kernel and group specific random probability measure $\tilde{p}_l$. The vector $\tilde{\boldsymbol{p}} = (\tilde{p}_1, \ldots, \tilde{p}_L)$ is distributed as a Griffiths-Milne dependent Dirichlet process (GM-DDP) with parameters $\vartheta > 0$ and $z \in (0, 1)$, and base measure P_0 on $\Theta = \mathbb{R} \times \mathbb{R}^+$, which implies that, marginally, groups are modeled with identically distributed location-scale DP mixtures, obtained by setting $\alpha = 0$ in (3).

The parameter z controls the dependence across the components of $\tilde{\boldsymbol{p}}$, with the two extremes of the range corresponding to full exchangeability (when $z \to 0$), that is $\tilde{p}_1 = \tilde{p}_2 = \ldots = \tilde{p}_L$, and independence across groups (when $z \to 1$), that is $\tilde{p}_l \overset{\text{iid}}{\sim} DP(\vartheta; P_0)$. Such characterization of the extreme cases helps in setting a value for z, which by default is fixed equal to 0.5. A formal elicitation of z might be obtained by specifying the value of the correlation between the random variables $\tilde{p}_{l_1}(A)$ and $\tilde{p}_{l_2}(A)$, with $l_1 \neq l_2$ and for a measurable A (see Equations 12 and 13 in Lijoi *et al.* 2014), quantity which is invariant with respect to the choice of A. In

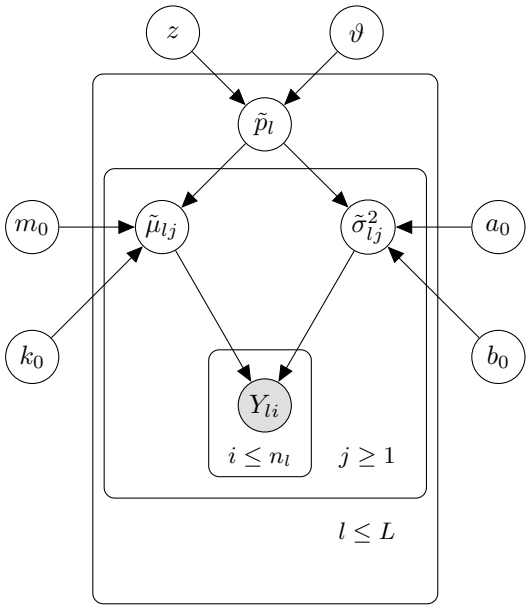

Figure 8: Hierarchical representation of the univariate location-scale DDP mixture model with Gaussian kernel.

order to achieve conjugacy, a normal-inverse gamma base measure is considered, that is $\tilde{\sigma}^2_{lj} \overset{\text{iid}}{\sim} \text{IGa}(a_0, b_0)$ and $\tilde{\mu}_{lj} \mid \tilde{\sigma}^2_{lj} \overset{\text{ind}}{\sim} \text{N}(m_0, \tilde{\sigma}^2_{lj}/k_0)$. Figure 8 displays a hierarchical representation of the univariate location-scale DDP mixture model.

3. Posterior simulation methods

The **BNPmix** package implements three types of MCMC algorithm for posterior simulation, namely the marginal sampler (Escobar and West 1995; Müller *et al.* 1996; Neal 2000), the slice sampler (Walker 2007; Kalli, Griffin, and Walker 2011), and the importance conditional sampler (ICS) (Canale, Corradin, and Nipoti 2020). The three sampling schemes are implemented for all the models described in Section 2, exception made for the univariate location-scale DDP mixture model, for which only the ICS is made available. Different simulation methods might be convenient in serving different purposes (see discussion in Canale *et al.* 2020): with this in mind, the method is offered as an option in the functions of the package, with the default one being the ICS, whose efficiency has been proved to be the most robust to model specifications (Canale *et al.* 2020).

Marginal sampler

Escobar and West (1995) and Müller *et al.* (1996) propose a marginal sampling scheme for DP mixtures of univariate and multivariate Gaussian kernels, respectively. The approach relies on the analytical marginalization of the mixing random probability measure $\tilde{p}$, while the parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ appearing in (2) are integrated out by means of a Gibbs sampler. The full conditional of each θ_i is reminiscent of the urn scheme of Blackwell and

MacQueen (1973), and, for the PY case with generic kernel $\mathcal{K}(\cdot;\cdot)$, is given by

$$\mathrm{P}[\theta_i \in \mathrm{d}t \mid \boldsymbol{\theta}^{(-i)}, \boldsymbol{y}^{(n)}] \propto \frac{\vartheta + k^{(-i)}\alpha}{\vartheta + n - 1}\mathcal{K}(\boldsymbol{y}_i; t)P_0(\mathrm{d}t) + \sum_{j=1}^{k^{(-i)}} \frac{n_j^{(-i)} - \alpha}{\vartheta + n - 1}\mathcal{K}(\boldsymbol{y}_i; \theta_j^*)\delta_{\theta_j^*}(\mathrm{d}t), \tag{4}$$

where $\boldsymbol{\theta}^{(-i)} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n)$ and $k^{(-i)}$ is the number of distinct values θ_j^* in $\boldsymbol{\theta}^{(-i)}$, with $n_j^{(-i)}$ denoting their frequencies. As a result, the marginal method generates realizations from the conditional posterior mean of the random density $\tilde{f}$, that is realizations of $\mathbb{E}[\tilde{f} \mid \boldsymbol{\theta}, \boldsymbol{y}^{(n)}]$, where the expectation is taken with respect to $\tilde{p}$. In case of nonconjugate specification of kernel and base measure, one can resort to Algorithm 8 of Neal (2000) which introduces a set of auxiliary random variables to evaluate the probability that θ_i takes a new value in (4).

Slice sampler

Introduced by Walker (2007) for DP mixtures, and improved on by Kalli *et al.* (2011), the slice sampler is a conditional method which works by introducing a suitable vector of augmenting random variables $\boldsymbol{U}^{(n)} = (U_1, \ldots, U_n)$. The components of $\boldsymbol{U}$ are independent uniform random variables and are such that the joint distribution of $(\boldsymbol{Y}_i, U_i)$ is given by

$$f(\boldsymbol{y}_i, u_i) = \sum_{j=1}^{\infty} \xi_j^{-1} \mathbb{1}_{\{u_i < \xi_j\}} \pi_j \mathcal{K}(\boldsymbol{y}_i; \tilde{\theta}_j), \tag{5}$$

where $\{\xi_j\}_{j=1}^{\infty}$ is any positive sequence. The random distribution of $\boldsymbol{Y}_i$, given in (3), is recovered by marginalizing (5) with respect to U_i. As a by-product and conveniently, the problem of dealing with the infinite-dimensional $\tilde{p}$ boils down to the problem of dealing with a finite sum, with a random number of terms. Approximate realizations of the posterior distributions of $\tilde{f}$ are obtained by implementing a Gibbs sampler which serves the purpose of marginalizing with respect to $\boldsymbol{U}$. Different choices for the sequence $\{\xi_j\}_{j=1}^{\infty}$ are considered. Following Kalli *et al.* (2011), **BNPmix** implements the dependent slice-efficient version of the algorithm by setting $\xi_j = \pi_j$, which leads to a convenient simplification of (5), and different versions of the independent slice-efficient sampler obtained by choosing a deterministic sequence $\{\xi_j\}_{j=1}^{\infty}$, the default choice being $\xi_j = \mathbb{E}(\pi_j)$. The efficiency of the dependent slice-efficient algorithm, when used for PY mixtures, is compromised if large values of α and ϑ are considered (see discussion in Canale *et al.* 2020).

Importance conditional sampler

The ICS is a conditional sampling strategy for PY mixture models, recently introduced by Canale *et al.* (2020). It combines a convenient characterization of the posterior distribution of a PY process (Pitman 1996) with a sampling importance resampling (SIR) step. The problem of dealing with the infinite-dimensional $\tilde{p}$ reduces to: 1) evaluating $\tilde{p}$ on a finite dimensional partition of the parameter space Θ, and 2) sampling, via SIR, from a distribution proportional to $\mathcal{K}(\boldsymbol{y}_i, t)\tilde{q}(\mathrm{d}t)$, where $\tilde{q}$ is PY process used as proposal distribution. Approximate realizations of the posterior distribution of $\tilde{f}$ are then obtained by marginalizing $\boldsymbol{\theta}$ out with a Gibbs sampler. The full conditional distribution of each θ_i, in a similar fashion as Algorithm 8 in Neal (2000), boils down to a discrete distribution with a finite support, given, up to a

proportionality constant, by

$$\mathrm{P}[\theta_i \in \mathrm{d}t \mid \boldsymbol{y}_i, \tilde{p}] \propto p_0 \sum_{l=1}^{k_m} \frac{m_l}{m} \mathcal{K}(\boldsymbol{y}_i; s_l^*) \delta_{s_l^*}(\mathrm{d}t) + \sum_{j=1}^{k_n} p_j \mathcal{K}(\boldsymbol{y}_i; t_j^*) \delta_{t_j^*}(\mathrm{d}t), \tag{6}$$

where $(t_1^*, \ldots, t_{k_n}^*)$ are the fixed jump points of $\tilde{p}$, $(p_0, p_1, \ldots, p_{k_n})$ is the random vector obtained by evaluating $\tilde{p}$ on the partition $(\Theta \setminus \{t_1^*, \ldots, t_{k_n}^*\}, t_1^*, \ldots, t_{k_n}^*)$, and $(s_1^*, \ldots, s_{k_n}^*)$ and $(m_1, \ldots, m_{k_m})$ are, respectively, the distinct values in the m-dimensional exchangeable sample generated from $\tilde{q}$ and their frequencies. The value of m can be chosen arbitrarily, with large values favoring a good mixing of the chain at the price of a longer run-time.

4. Package implementation

The **BNPmix** package consists of three main R functions, wrappers of C++ routines which implement the BNP models described in Section 2 and the MCMC simulation methods introduced in Section 3, along with some user-friendly functions which facilitate the elicitation of prior distributions and the post-processing of generated posterior samples. The three main functions are `PYdensity`, `PYregression`, and `DDPdensity`, and implement, respectively, Pitman-Yor mixture models for density estimation and clustering, a Pitman-Yor mixture model for density regression, and a DDP mixture model for density estimation of correlated samples. The spirit of the package is two-fold. On the one side, it provides a ready-to-use suite of functions to estimate a rich variety of BNP models: to this end, all functions provide a default specifications of their arguments, thus allowing for a non-informed estimation procedure. On the other side, each function offers a detailed list of arguments, concerning prior parameters, hyperpriors specification, kernel structure and sampling approach, which can be tuned by the more experienced user so to formalize and exploit available prior information.

The remainder of the section deals with both what lies under the hood (Section 4.1) and the design of the R wrappers (Section 4.2).

4.1. Low level implementation

All the main functions of the **BNPmix** package are written in C++ and exploit the seamless interface to R provided by **Rcpp** and **RcppArmadillo** (Eddelbuettel and François 2011; Eddelbuettel and Sanderson 2014).

All the C++ routines implementing the models listed in Section 2 share a common structure as they consist of a main function—where all the required objects are initialized—with a loop cycling over the number of MCMC iterations. Within this loop, all the method-specific functions are called to update the MCMC status.

Broadly speaking, any MCMC algorithm for posterior simulation under a mixture model setup includes two steps, which consist in *i*) allocating individual observations to clusters or mixture components, and *ii*) updating the parameters specific to each group or mixture component. In our C++ implementation, step *i*) is performed by specific functions, with names of the type `clust_update_method_model`, where `method` refers to one of the three MCMC methods discussed in Section 3 while `model` refers to one of the mixture specifications presented in Section 2. While step *i*) differs intrinsically from one MCMC method to another, step *ii*) stays fundamentally unchanged for different MCMC methods, as the update of component specific

parameters essentially depends on the structure of the kernel only. In our C++ implementation, step *ii*) is performed by specific routines, whose names —`accellerate_method_model`— follow the same logic as those of the functions implementing step *i*). While method-specific, these functions share most of the C++ code needed to perform common tasks. Such common structure allows, as a by-product, for a fair comparison between the performance of different MCMC simulation methods.

In order to limit the memory usage, all the `accellerate_method_model` functions are of type void. This conveniently allows us to update, at each iteration of the MCMC, all the quantities needed to produce posterior summaries — e.g., realizations of posterior densities evaluated at a grid of points, or partitions of the data into clusters — while passing on to the next iteration only those required for the evaluation of full conditional distributions. The number of elements to update can vary across the iterations of the MCMC (see Canale *et al.* 2020, for details), aspect which is addressed by means of method-specific functions, named `para_clean_method_model`, which suitably resize and reorder the arrays where the values taken by model parameters are stored. In line with the two-fold spirit of the package, however, the option to save all the quantities generated by the MCMC algorithm is offered for the user to independently compute any posterior summary that might be of interest.

Finally, functions of the type `hyper_accellerate_method_model` allow hyperprior distributions to be added on the parameters defining the base measure, thus avoiding the often daunting task of eliciting the same parameters.

4.2. Wrappers to the main functions

A BNP mixture model can be fitted with the **BNPmix** package by calling an R function which, based on its arguments, interfaces with one of the specific C++ subroutines described in Section 4.1. All the R functions require a data set `y` and a set of arguments referring to MCMC method, prior specification and form of the produced output. The MCMC parameters are passed through the named list `mcmc`, which allows the following arguments to be specified:

- `niter`, the number of MCMC iterations;
- `nburn`, the number of burn-in iterations to discard;
- `nupd`, argument controlling the number of iterations to be displayed on screen;
- `print_message`, control option: if equal to `TRUE` the status is printed to standard output every `nupd` iterations.

The prior specification is passed to the C++ routines through the named list `prior`. The latter includes the arguments `strength` (1 by default) and `discount` (0 by default) for the strength parameter ϑ and the discount parameter α of the Pitman-Yor process, and a set of model-specific arguments, as described in Section 2. If `hyper = TRUE`, as by default, the parameters defining the base measure are endowed with hyperprior distributions.

Finally, the arguments of the named list `output` describe the output which is to be returned. Such list shares a structure which is common to all the functions, and contains:

- `out_type`, summaries of the posterior distribution to be returned. If equal to `"FULL"`, the function returns the estimated partition and all the MCMC realizations of the

posterior densities. If equal to `"MEAN"`, the function returns the estimated partition and the mean of the sampled densities. If equal `"CLUST"`, the function only returns the estimated partition;

- `out_param`, option to return the values taken by cluster or component-specific parameters. If equal to `TRUE`, the function returns all the MCMC draws of such parameters;
- `grid`, a grid of points (or a data frame obtained via `expand.grid`) where to evaluate posterior densities.

While the general structure of these named lists is common across functions, there exist model-specific arguments which we describe, for each R function, in the remainder of the section.

PY mixture models

The function `PYdensity` performs univariate and multivariate density estimation and clustering, via Pitman-Yor mixtures with Gaussian kernels. Specific elements of the `mcmc` list are:

- `method`, the MCMC algorithm chosen to perform the estimation. Three options are available, namely ICS (`method = "ICS"`), marginal sampler (`method = "MAR"`) and slice sampler (`method = "SLI"`);
- `model`, the specific mixture model, among those described in Section 2, to be fitted. The default choice is location-scale mixture (`model = "LS"`) for both univariate and multivariate data. Other options are location mixture (`model = "L"`) for both univariate and multivariate data, and location-scale mixture with diagonal covariance matrix (`model = "DLS"`) for multivariate data only;
- `m_imp` (available if `method = "ICS"`), size (given by m in the notation of Section 3) of the exchangeable sample generated from the proposal distribution $\tilde{q}$, in the SIR step of the ICS method. Default is `m_imp = 10`;
- `slice_type` (available if `method = "SLI"`), the specification of the type of slice sampler. Options are `"DEP"` for dependent slice-efficient, and `"INDEP"` for independent slice-efficient. Default is `slice_type = "DEP"`.

Finally, the named list `prior` for `PYdensity` admits a set of model-specific arguments, for which an exhaustive description is reported in Table 1. The default values of the arguments in the `prior` list are set so that the expectation of the location component equals the sample mean, and both the variance of the location component and the expectation of the scale component coincide with the sample variance.

PY density regression

The function `PYregression` performs univariate density regression and clustering, via Pitman-Yor mixture of normal linear regressions. Beside `y`, the argument `x`, that is the values taken by the d regressors, is also required.

The specific arguments of the named list `mcmc` for `PYregression` are:

Table 1: Parameters for the `prior` list of the `PYdensity` function.

`hyper = FALSE`	Univariate	Multivariate
`model = "L"`	`m0`, mean of $\tilde{\mu}_j$ `s20`, variance of $\tilde{\mu}_j$ `a0`, shape of σ^2 `b0`, scale of σ^2	`m0`, mean of $\tilde{\boldsymbol{\mu}}_j$ `S20`, covariance of $\tilde{\boldsymbol{\mu}}_j$ `Sigma0`, scale matrix of $\boldsymbol{\Sigma}$ `n0`, df of $\boldsymbol{\Sigma}$
`model = "LS"`	`m0`, mean of $\tilde{\mu}_j$ `k0`, scale factor of $\tilde{\mu}_j$ `a0`, shape of $\tilde{\sigma}^2_j$ `b0`, scale of $\tilde{\sigma}^2_j$	`m0`, mean of $\tilde{\boldsymbol{\mu}}_j$ `k0`, scale factor of $\tilde{\boldsymbol{\mu}}_j$ `S0`, matrix of $\tilde{\boldsymbol{\Sigma}}_j$ `n0`, df of $\tilde{\boldsymbol{\Sigma}}_j$
`model = "DLS"`		`m0`, mean of $\tilde{\boldsymbol{\mu}}_j$ (vector) `k0`, scale factor of $\tilde{\boldsymbol{\mu}}_j$ (vector) `a0`, shape of $\tilde{\boldsymbol{\Sigma}}_j$ (vector) `b0`, scale of $\tilde{\boldsymbol{\Sigma}}_j$ (vector)
`hyper = TRUE`	Univariate	Multivariate

`model = "L"`	`m1`, mean of `m0`
	`k1`, scale factor of `m0`
	`a1`, shape of `s20`
	`b1`, scale of `s20`
		`m1`, mean of `m0`
		`k1`, scale factor of `m0`
		`lambda1`, df of `S20`
		`Lambda1`, matrix of `S20`

`model = "LS"`	`m1`, mean of `m0`
	`s21`, variance of `m0`
	`tau1`, shape of `k0`
	`zeta1`, rate of `k0`
	`a1`, shape of `b0`
	`b1`, rate of `b0`
		`m1`, mean of `m0`
		`S1`, covariance of `m0`
		`tau1`, shape of `k0`
		`zeta1`, rate of `k0`
		`n1`, df of `Sigma0`
		`Sigma1`, matrix of `Sigma0`

`model = "DLS"`		`m1`, mean of `m0` (vector) `s21`, variance of `m0` (vector) `tau1`, shape of `k0` (vector) `zeta1`, rate of `k0` (vector) `a1`, shape of `b0` (vector) `b1`, rate of `b0` (vector)

- `method`, the MCMC algorithm chosen to perform the estimation. Three options are available, namely ICS (`method = "ICS"`), marginal sampler (`method = "MAR"`) and slice sampler (`method = "SLI"`);

- `model`, the specific mixture model, between those described in Section 2, to be fitted. The default choice is location-scale mixture (`model = "LS"`), a second option is the location mixture (`model = "L"`);

- `m_imp` (available if `method = "ICS"`), size of the exchangeable sample generated from the proposal distribution $\tilde{q}$, in the SIR step of the ICS method. Default is `m_imp = 10`;

- `m_marginal` (available if `method = "MAR"`), number of auxiliary variables introduced for the Monte Carlo evaluation of the probability that θ_i takes a new value in (4) (see Algorithm 8 of Neal 2000);

Table 2: Parameters for the `prior` list of the `PYregression` function.

	`hyper = FALSE`	`hyper = TRUE`
`model = "L"`	`m0`, mean of $\tilde{\beta}_j$	`m1`, mean of `m0`
	`S0`, covariance of $\tilde{\beta}_j$	`k1`, scale factor of `m0`
	`a0`, shape parameter of σ^2	`n1`, degrees of freedom for `S0`
	`b0`, scale parameter of σ^2	`S1`, scale matrix of `S0`
		`a0`, shape parameter of σ^2
		`b0`, scale parameter of σ^2
`model = "LS"`	`m0`, mean of $\tilde{\beta}_j$	`m1`, mean of `m0`
	`S0`, covariance of $\tilde{\beta}_j$	`k1`, scale factor of `m0`
	`a0`, shape parameter of $\tilde{\sigma}_j^2$	`n1`, degrees of freedom for `S0`
	`b0`, scale parameter of $\tilde{\sigma}_j^2$	`S1`, scale matrix of `S0`
		`tau1`, shape of `b0`
		`zeta1`, rate of `b0`

- `slice_type` (available if `method = "SLI"`), the specification of the type of slice sampler. Options are `"DEP"` for dependent slice-efficient, and `"INDEP"` for independent slice-efficient. Default is `slice_type = "DEP"`.

Finally, the model-specific arguments admitted by the named list `prior` for `PYregression`, are described in Table 2. The default specification of the hyperparameters is such that the intercept has prior expectation equal to the average of `y` and variance 100, the regression coefficients have prior expectation and variance equal to 0 and 100, respectively, and the prior expectation for the scale component coincides to the sample variance of `y`.

DDP density estimation

The function `DDPdensity` performs univariate density estimation for correlated samples, via GM-DDP mixture model with Gaussian kernel. Beside `y`, an argument named `groups` is also required. The latter can be thought of as a label assigned to each observation in `y`, which thus identifies distinct subsamples in `y`. Unlike the previous two functions, the named list `mcmc` does not account for the MCMC algorithm choice since only the ICS method is implemented for this class of models. Beside the common arguments, the `mcmc` named list includes, as model-specific, `m_imp`, which is the size of the exchangeable sample generated from the proposal distribution $\tilde{q}$, in the SIR step of the ICS method.

The base measure of the `DDPdensity` function is a normal-inverse gamma distribution with parameters `m0`, `k0`, `a0` and `b0`, where the first two are mean and scale factor defining the normal base measure on the location parameter, and the latter two are shape and scale of the inverse gamma base measure on the scale parameter. These parameters can be specified as arguments of the named list `prior`. Their default specification is such that the expectation of the location component of the base measure is equal to the overall sample mean, obtained by pooling the groups together, and the expectation of the scale component coincides to the overall sample variance. In addition, the `prior` list admits the argument `wei` (z in the notation of Section 2), which is a parameter taking values in $(0, 1)$ (default is 0.5) and controlling the strength of the borrowing of information across groups. Finally, notice that, for the `DDPdensity` function, the argument `discount` is fixed equal to 0 while one can tune

the argument `strength` (1 by default), corresponding to the parameter ϑ in the specification of the univariate location-scale DDP mixture model in Section 2.

Other functions

The `PYdensity`, `PYregression`, and `DDPdensity` functions return an object of class `BNPdens`. The `plot` method, extended to the `BNPdens` class by means of the **ggplot2** package, produces a plot of the estimated posterior mean density function. Based on the type of model and the dimension of the data, `plot` can be called with additional arguments. Specifically:

- if the `BNPdens` object is produced by `PYdensity` and data are univariate, `plot` admits the argument `show_hist`, a logical argument which returns the histogram of the raw data along with the posterior density. The size of the bins can be set with `bin_size`. Observations can be displayed on the x-axis by specifying `show_points = TRUE`. Moreover, if also `show_clust = TRUE`, the displayed observations are colored based on the estimated clustering. The `col` argument controls the color of the estimated posterior density. Pointwise posterior credible bands of level `conf_level` around the posterior mean densities can be added by setting `band = TRUE`;

- if the `BNPdens` object is produced by `PYdensity` and data are multivariate, `plot` generates, for the pair of variables indexed by `dimension`, the bivariate contour plot of the corresponding bivariate marginal density function. Adding `show_points = TRUE` or `show_clust = TRUE`, allows to display the observations in the contour plot and to color them based on the estimated clustering;

- if the `BNPdens` object is produced by `PYregression` and the number of covariates does not exceed four, the `plot` function returns the scatterplot of the observations colored according to the estimated clustering;

- if the `BNPdens` object is produced by `DDPdensity` and thus data consist of two or more subgroups, the `plot` function returns a wrapped plot, with each subplot corresponding to a specific group. An additional argument specific to the output of `DDPdensity` is `wrap_dim`, which allows to choose the number of rows and columns in the plot.

Other methods have been extended to the `BNPdens` class. The `print` method returns details on the `BNPdens` object and the model which produced it. The `summary` method returns some basic information on the model fitting, such as number of iterations, number of burn-in iterations, computational time and average number of clusters. The `partition` method, when applied to `BNPdens` objects, returns a point estimate for the clustering of the data, based on the partitions visited during the MCMC algorithm. This can be achieved by adopting two distinct loss functions, namely the variation of information loss function (`dist = "VI"`) and Binder's loss function (`dist = "Binder"`). This method is an efficient C++ implementation of the method described by Wade and Ghahramani (2018). The **BNPmix** package also includes a method, named `BNPdens2coda`, to interface objects of class `BNPdens` with the **coda** package. For univariate PY mixture models, the `BNPdens2coda` method exports a matrix, whose first row reports the number of blocks of the partitions visited by the chain, while each one of the remaining rows is composed by the values taken by the density at each point of the grid (`grid`), at different iterations of the MCMC algorithm. For multivariate and regression PY

mixture models, the `BNPdens2coda` method exports only a vector with the number of blocks of the partitions visited by the chain. For DDP mixture models the `BNPdens2coda` method exports a matrix whose first row reports the number of clusters of the partition visited by the chain, while the other rows report the values taken by the weights of the group specific processes at each iteration. Finally, the `PYcalibrate` function allows to elicit the strength parameter of the Pitman-Yor process by specifying the sample size and by fixing the discount parameter and the expected number of clusters a priori.

4.3. Package scalability

Considering the variety of models implemented in the **BNPmix** package and the possibility to fit these models to data sets with different levels of complexity in terms of size (n), dimension (p), presence and number of covariates (d) and number of groups (L), we briefly comment on the scalability of the package with respect to these quantities. Based on our experience, when considering the default specifications and without taking into account the time needed to evaluate estimated densities on a given grid of values (i.e., `output$out_type = "CLUST"`), the average time per iteration for the functions `PYdensity`, `PYregression`, and `DDPdensity`, displays a linear grow as the sample size n becomes large. The run-time is also affected by the complexity of the data. Specifically, assuming n is kept fixed, a growth faster than linear is observed both when the dimension p in `PYdensity` and when the number of covariates d in `PYregression` increases. Finally, the function `DDPdensity` shows a roughly linear growth of its average time per iteration as the number of groups L becomes large. The described behavior of `PYdensity`, `PYregression` and `DDPdensity` is not surprising as it is well known that MCMC methods are severely affected by the dimension of the space that has to be explored by the chain, that is the support of the posterior distribution for the functions implemented in **BNPmix**, which is ultimately connected to the quantities discussed above. Evaluating the generated densities on a given grid of points at each iteration of an MCMC algorithm can severely affect computational efficiency, issue which is particularly relevant for large dimensions p in `PYdensity` or number of covariates d in `PYregression`. Note, however, that **BNPmix** conveniently allows users to specify the type of desired output (see Section 4.2), thus avoiding unnecessary computations if not needed.

5. Usage of the package

We next illustrate the use of the **BNPmix** package by walking the reader through the entire process of density estimation, clustering and regression, via BNP mixture models. We start by showing how to elicit prior distributions, how to run the sampler, and how to process the output of the functions, for inferential purposes. In order to do this, we analyze different subsets of the Collaborative Perinatal Project (CPP) data set (Klebanoff 2009), as well as synthetic data. The CPP data set is a rich collection of observations referring to a large prospective study conducted in the United States in the '60s. The goal of the study was to assess the cause of neurological disorders and other pathologies in children. The full data set counts more than 2 300 observations, each consisting of several measurements referring to pregnant women and their babies. The focus of our illustrative analysis is on three quantities: the gestational age (in weeks), the weight of the baby at birth (in g), and the concentration level in maternal serum of DDE (in µg/L), a persistent metabolite of the pesticide DDT,

known to have adverse impact on health (Longnecker, Klebanoff, Zhou, and Brock 2001). This subset of the original data set is available in the **BNPmix** package via

```
R> library("BNPmix")
R> data(CPP, package = "BNPmix")
```

5.1. Univariate density estimation

We illustrate here how to perform univariate density estimation and clustering, by focusing on the gestational age for a group of women belonging to one of the 12 university hospitals participating in the study.

```
R> y <- CPP[CPP$hosp == 11, ]
```

We consider a DP location-scale mixture model, thus setting the discount parameter $\alpha = 0$. Previous studies show that the distribution of the gestational age is left skewed and shows an irregular shape due to the presence of three subpopulations corresponding to early preterm, preterm, and normal births. For this reason we fix the prior expected number of clusters to three and choose the value of strength parameter ϑ accordingly. This can be done by using the `PYcalibrate` function.

```
R> DPprior <- PYcalibrate(Ek = 3, n = nrow(y), discount = 0)
```

We endow the parameters of the base measure with hyperprior distributions with default parameters. The complete prior specification for our BNP mixture model is specified as

```
R> prior <- list(strength = DPprior$strength, discount = 0)
```

Before running the MCMC algorithm (opting here and henceforth for the default ICS simulation method), we specify the number of MCMC iterations and the grid of points where to evaluate the density. This can be done with

```
R> mcmc <- list(niter = 5000, nburn = 4000)
R> output <- list(grid = seq(30, 46, length.out = 100), out_type =
```

Then MCMC algorithm can then be run by calling the `PYdensity` function and providing all the previously defined lists.

```
R> set.seed(42)
R> fit1 <- PYdensity(y = y$gest, mcmc = mcmc, prior = prior, outpu
R> print(fit1)

BNPdens object
Type: univariate density
```

The plot of the posterior mean density is produced by calling the method `plot` on the fitted model `fit1`. The posterior mean density plot can be endowed with pointwise quantile-based posterior credible bands, histogram of the raw data, observations colored according to the estimated clustering (obtained by calling the function `partition` within the method `plot`), or a combination of them. The two examples displayed in Figure 9 have been produced by the following code:

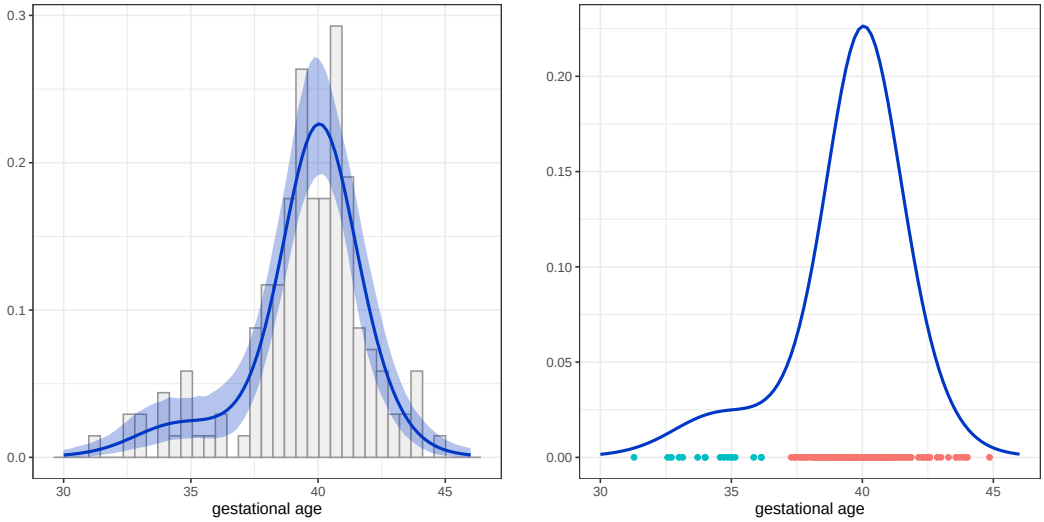

Figure 9: Posterior mean density of `gest` in the 11th hospital (left) and 95% pointwise posterior credible bands along with the histogram of the raw data and observations colored according to the estimated clustering (right).

```
R> plot(fit1, show_hist = TRUE, xlab = "gestational age")
R> plot(fit1, band = FALSE, show_clust = TRUE, xlab = "gestational
```

5.2. Multivariate density estimation

Next we illustrate how to perform density estimation and model-based clustering for multivariate data. The first illustration focuses on the three continuous variables composing the `CPP` data set. The goal of the analysis is to provide an estimate of the joint trivariate density function of the variables gestational age, DDE (after a logarithmic transformation), and weight at birth.

```
R> y <- cbind(CPP$gest, log(CPP$dde), CPP$weight)
```

Given the lack of precise prior information on the clustering structure of the data when these three variables are concerned, we consider a PY mixture model with discount parameter $\alpha = 0.1$ and strength parameter ϑ equal to 0.05. We also assume hyperprior distributions on the parameters of the base measure and adopt the default empirical specification for the corresponding hyperparameters.

```
R> prior <- list(strength = 0.05, discount = 0.1)
```

We specify the remaining arguments of `PYdensity` and run the MCMC algorithm with

```
R> grid <- expand.grid(seq(30, 46, length.out = 25
+    seq(0.7, 5.2, length.out = 25), seq(0, 130, l
R> output <- list(grid = grid, out_type = "FULL")
R> mcmc <- list(niter = 2000, nburn = 1000, m_imp
```

```
R> set.seed(42)
R> fit2 <- PYdensity(y = y, mcmc = mcmc, prior = p
```

Posterior mean densities, univariate marginal or bivariate marginal (collected and displayed in Figure 10), can be produced by calling the method `plot` as follows:

```
R> p12 <- plot(fit2, dim = c(1, 2), show_clust =
+    xlab = "gestational age", ylab = "log(DDE)")
R> p21 <- plot(fit2, dim = c(2, 1), show_clust = T
+    ylab = "gestational age", xlab = "log(DDE)")
R> p13 <- plot(fit2, dim = c(1, 3), show_clust = T
+    slab = "gestational age", ylab = "weight")
R> p23 <- plot(fit2, dim = c(2, 3), show_clust = T
+    xlab = "log(DDE)", ylab = "weight")
R> p32 <- plot(fit2, dim = c(3, 2), show_clust = T
+    ylab = "log(DDE)", xlab = "weight")
R> p31 <- plot(fit2, dim = c(3, 1), show_clust = T
+    ylab = "gestational age", xlab = "weight")
R> p1 <- plot(fit2, dim = c(1, 1), show_clust = TR
+    xlab = "gestational age", ylab = "density")
R> p2 <- plot(fit2, dim = c(2, 2), show_clust = TR
+    xlab = "log(DDE)", ylab = "density")
R> p3 <- plot(fit2, dim = c(3, 3), show_clust = TR
+    xlab = "weight", ylab = "density")
R> gridExtra::grid.arrange(p1, p12, p13, p21, p2,
+    layout_matrix = matrix(1:9, 3, 3))
```

A second illustration of the usage of `PYdensity` focuses on model-based clustering only, and considers observations of a larger dimension. To this end, we consider a synthetic data set and estimate the clustering structure induced by a BNP model assuming a more parsimonious location-scale mixture specification where each Gaussian component has a diagonal covariance matrix. This is done by specifying the option `model = "DLS"` in the `mcmc` list. We also exploit the marginal sampler implementation specifying `method = "MAR"` in the `mcmc` list.

```
R> output <- list(out_type = "CLUS
R> mcmc <- list(niter = 5000, nbur
```

A sample of 100 synthetic observations of dimension $p = 25$ is simulated from a mixture of two Gaussians and one scaled Student t distribution. Given this simulation framework, we might expect to observe two regions of the sample space dense of observations, and a moderate number of observations, generated from the heavy tailed mixture component, which are likely to constitute singleton clusters in our model-based clustering analysis. The code to simulate these data is

```
R> p <- 25
R> set.seed(42)
R> ysim <- rbind(
+    mnormt::rmnorm(50, mean = rep
```

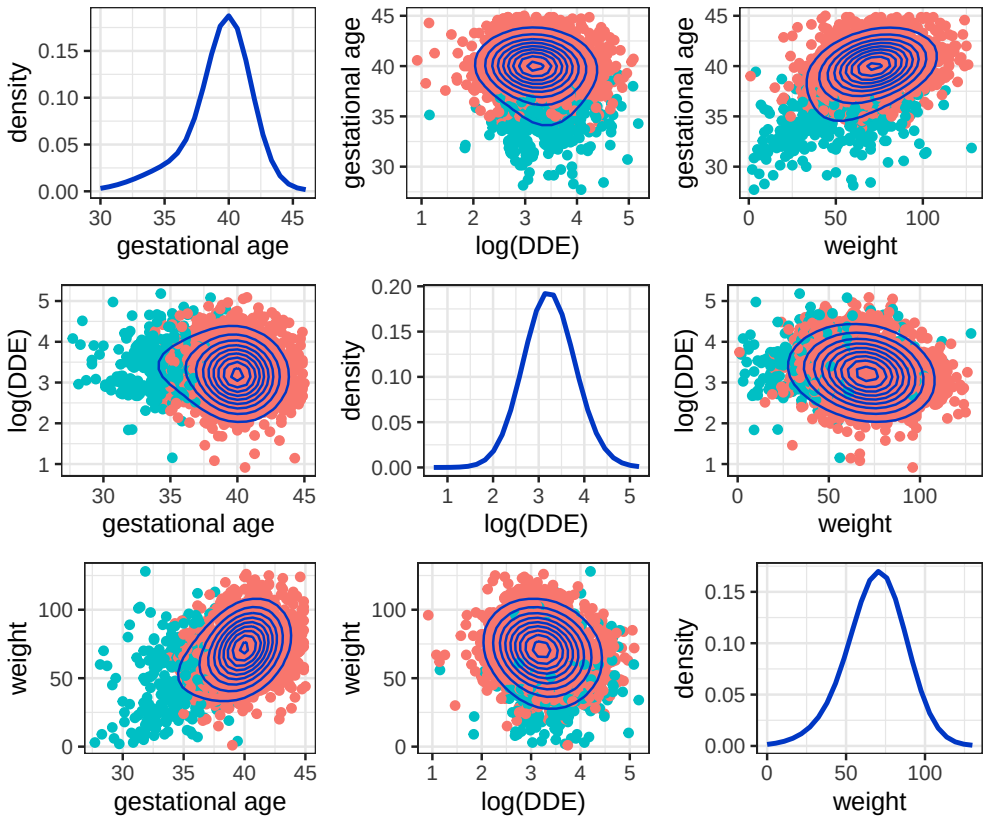

Figure 10: Posterior mean densities, univariate marginal and pairwise bivariate marginal, obtained by calling `plot` on the fitted model `fit2`. In addition to the posterior mean density value, each pairwise bivariate density shows the data points colored according to the estimated clustering.

```
+     mnormt::rmnorm(40, mean = sam
+     varcov = diag(1, p)),
+     matrix(rt(10*p, df = 2), 10,
R> ysim <- scale(ysim)
```

We run the MCMC with the following prior specification:

```
R> prior <- list(strength = 1, dis
R> fit.sim <- PYdensity(y = ysim,
+     output = output)
```

The clustering structure detected by the model fitted to the object `fit.sim` can be explored by using the function `partition`. As point estimate, we consider the partition which, among those visited during the MCMC, minimizes the posterior expected loss, where we work under the variation of information framework introduced and studied by Wade and Ghahramani (2018) in the context of clustering estimation problems.

```
R> fit.sim.part <- partition(fit.s
R> sort(ftable(fit.sim.part$partit
```

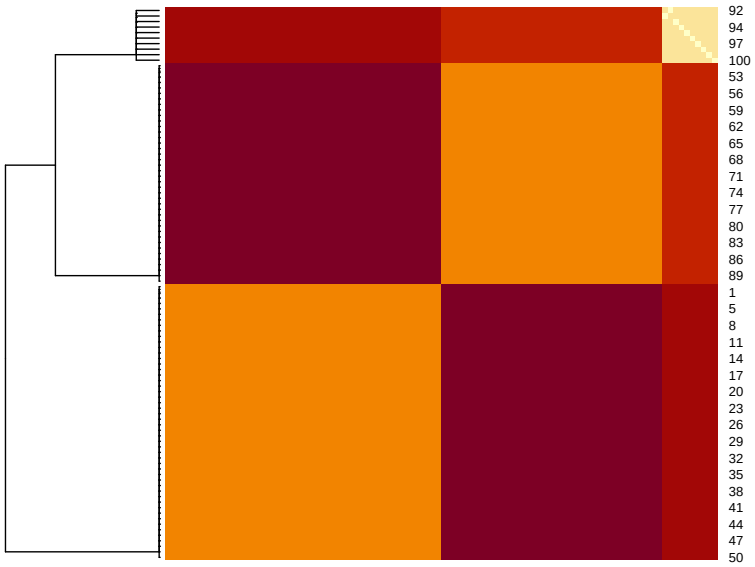

Figure 11: Dissimilarity matrix returned by `partition(fit.sim)`, obtained by using Wade and Ghahramani (2018)'s method with variation of information loss function (`dist = "VI"`).

```
50 40  1  1  1  1  1  1  1  1  1
```

The data appear grouped into twelve clusters. The frequencies are consistent with the presence of two large clusters and ten small ones, all being singletons, arguably representing the 10% of the data coming from the heavy-tailed Student t distribution. Further insight on the clustering structure of the data can be gained by visualizing the estimated dissimilarity matrix with

```
R> dissmat <- as.dist(1 - fit.sim.
R> clus <- hclust(dissmat)
R> heatmap(as.matrix(dissmat), Row
+    Colv = NA)
```

The dendrogram obtained with the complete linkage through the `hclust` function (displayed in Figure 11) clearly shows the presence of two main clusters. The remaining small clusters, while distinct in our analysis, can be interpreted as one group of outlying observations, in agreement with the simulation scenario we devised.

5.3. Density regression

In order to illustrate the usage of the `PYregression` function, we study how the distribution of gestational age changes with DDE. Following the same argument as in Section 5.1, we fix the prior expected number of clusters to 3 and, given a moderate discount parameter $\alpha = 0.25$, we elicit the strength parameter via `PYcalibrate`. We opt for the default specification of the other hyperparameters and summarize our prior assumptions with the code

```
R> PYpar <- PYcalibrate(Ek = 3, n
R> prior <- list(strength = PYpar$
```

As summary of the posterior distribution, we compute the estimated conditional density of `gest`, conditionally on 6 values of `dde` equal to the minimum observed value and the empirical quantiles of order 0.25, 0.5, 0.75, 0.95, and 0.99.

```
R> grid_y <- seq(26, 47, length.ou
R> grid_x <- round(quantile(CPP$dd
R> mcmc <- list(niter = 2000, nbur
R> output <- list(grid_x = grid_x,
+    out_type = "FULL", out_param
R> set.seed(42)
R> fit.reg <- PYregression(y = CPP
+    mcmc = mcmc, output = output)
```

We next plot the posterior mean conditional density of `gest`, given the values of `dde` in `grid_x`. In doing this, we exploit all the information contained in the object `fit.reg`, which allows us to compute also pointwise posterior credible bands for the estimated densities.

```
R> regplot <- data.frame(
+    dens = as.vector(apply(fit.re
+    qlow = as.vector(apply(fit.re
+    quantile, probs = 0.025)),
+    qupp = as.vector(apply(fit.re
+    quantile, probs = 0.975)),
+    grid = rep(grid_y, 6),
+    label = factor(rep(paste("DDE
+    level = rep(paste("DDE = ", g
R> library(ggplot2)
R> ggplot(regplot) + theme_bw() +
+    geom_line(data = regplot, map
+    geom_ribbon(data = regplot, m
+    ymax = qupp), fill = "blue",
+    facet_wrap(~label, ncol = 3,
+    labs(x = "gestational age", y
```

The results displayed in Figure 12 show, for increasing values of `dde`, a deflation of the mode around 40 weeks of pregnancy and an inflation of the left tail of the distribution (corresponding to preterm births). This is an expected behavior, possibly due to the negative effect of DDE on gestational age (Longnecker *et al.* 2001). For a related discussion involving a different BNP mixture approach see Canale, Durante, and Dunson (2018).

5.4. Density estimation for correlated samples

We conclude this section by illustrating the usage of the `DDPdensity` function. The CPP data set consists of observations coming from 12 hospitals: while assuming homogeneity within each hospital seems reasonable, we opt for a model which might account for heterogeneity across hospitals and thus consider the GM-DDP mixture model with Gaussian kernels, described in Section 2. We use an empirical approach and center the base measure parameters on sample summaries, as by default.

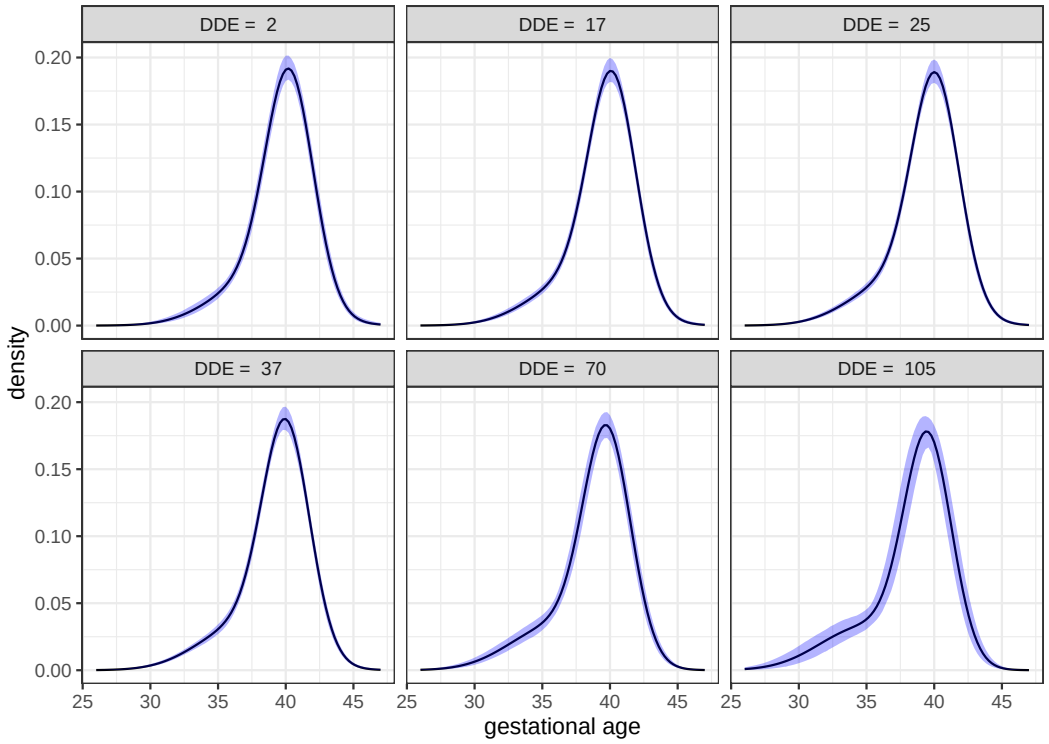

Figure 12: Posterior mean conditional univariate densities (and 95% pointwise credible bands) of `gest`, conditionally on the values of `dde` reported in the plots' headers.

```
R> mcmc <- list(niter = 5000, nbur
R> output <- list(grid = seq(30, 4
```

The lists defined above are supplied as arguments to the function `DDPdensity`, which can be run to estimate the posterior mean densities for the 12 hospitals. The latter can be visualized by running the method `plot` on the output of `DDPdensity`.

```
R> fit.ddp <- DDPdensity(y = CPP$g
+    output = output)
R> fit.ddp$group <- factor(CPP$hos
+    labels = paste("Hospital ", 1
R> plot(fit.ddp, wrap_dim = c(3, 4
+    xlab = "gestational age")
```

Figure 13 displays the estimated densities along with pointwise 95% posterior credible bands. The model successfully accounts for heterogeneity across hospitals: this can be noticed, for example, by observing that the estimated densities for hospitals 3 and 5 are more skewed than those of most of the other hospitals in the study. At the same time, the model allows for borrowing information across hospitals displaying similar distributions: the result of this is apparent when the posterior density of `gest` for hospital 11 is compared with the one, for the same hospital, displayed in Figure 9 and obtained by ignoring information from other hospitals.

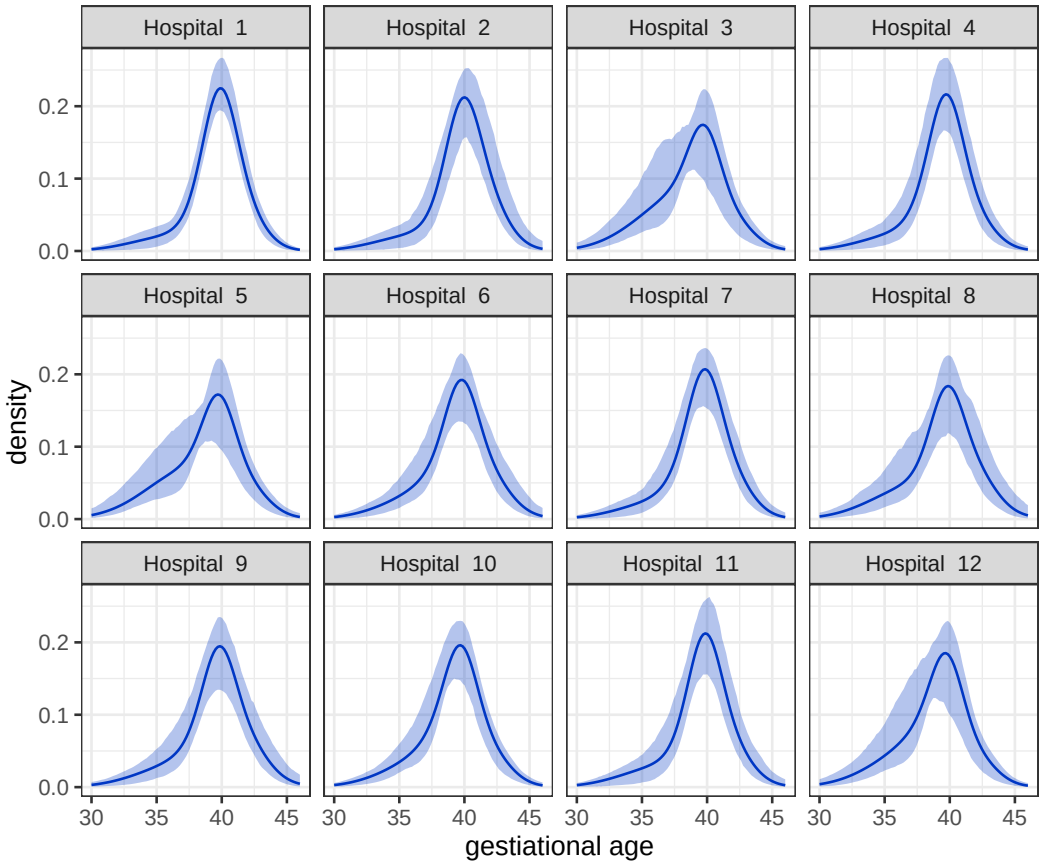

Figure 13: Posterior mean densities and 95% pointwise posterior credible bands of gestational age for the 12 hospitals.

Technical details

The results presented in this paper were obtained by using a macOS 10.15.7 machine. All the routines were executed on R 4.1.1 with the **BNPmix** 0.2.9 package, dependent of **Rcpp** 1.0.7 and **RcppArmadillo** 0.10.7.0.0 packages. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

Acknowledgments

The authors wish to thank two anonymous referees and the guest editors of the special issue on Bayesian Statistics. Riccardo Corradin and Bernardo Nipoti are grateful to the DEMS Data Science Lab for supporting this work by providing computational resources. Antonio Canale is supported by the University of Padova under the STARS Grant program (acronym of the project: BNP-CD).

References

Ahlmann-Eltze C, Yau C (2018). "**MixDir**: Scalable Bayesian Clustering for High-Dimensional Categorical Data." In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 526–539. doi:10.1109/DSAA.2018.00068.

Argiento R, Bianchini I, Guglielmi A (2016). "Posterior sampling from ε-approximation of normalized completely random measure mixtures." *Electron. J. Statist.*, **10**(2), 3516–3547. doi:10.1214/16-EJS1168.

Barrios E, Lijoi A, Nieto-Barajas LE, Prünster I (2013). "Modeling with Normalized Random Measure Mixture Models." *Statist. Sci.*, **28**(3), 313–334. doi:10.1214/13-STS416.

Barrios E, Lijoi A, Nieto-Barajas LE, Prünster I, Kon Kam King G (2017). ***BNPdensity**: Ferguson-Klass Type Algorithm for Posterior Normalized Rando Measures*. R package version 2020.3.4, URL https://CRAN.R-project.org/package=BNPdensity.

Blackwell D, MacQueen JB (1973). "Ferguson Distributions Via Polya Urn Schemes." *The Annals of Statistics*, **1**(2), 353–355. doi:10.1214/aos/1176342372.

Blei DM, Jordan MI (2006). "Variational inference for Dirichlet process mixtures." *Bayesian Anal.*, **1**(1), 121–143. doi:10.1214/06-BA104.

Bouveyron C, Brunet-Saumard C (2014). "Model-based clustering of high-dimensional data: A review." *Computational Statistics & Data Analysis*, **71**, 52 – 78. ISSN 0167-9473. doi:https://doi.org/10.1016/j.csda.2012.12.008.

Campbell T, Straub J, Fisher III JW, How JP (2015). "Streaming, distributed variational inference for Bayesian nonparametrics." In *Advances in Neural Information Processing Systems*, pp. 280–288.

Canale A (2017). "**msBP**: An R Package to Perform Bayesian Nonparametric Inference Using Multiscale Bernstein Polynomials Mixtures." *Journal of Statistical Software*, **78**(6), 1–19. ISSN 1548-7660. doi:10.18637/jss.v078.i06.

Canale A, Corradin R, Nipoti B (2020). "Importance Conditional Sampling for Pitman-Yor mixtures." *preprint arXiv:1906.08147*.

Canale A, Dunson D (2011). "Bayesian multivariate mixed-scale density estimation." *Statistics and Its Interface*, **8**. doi:10.4310/SII.2015.v8.n2.a7.

Canale A, Dunson D (2014). "Multiscale Bernstein polynomials for densities." *Statistica Sinica*, **26**. doi:10.5705/ss.202015.0163.

Canale A, Durante D, Dunson DB (2018). "Convex mixture regression for quantitative risk assessment." *Biometrics*, **74**(4), 1331–1340. doi:10.1111/biom.12917.

Carmona C, Nieto-Barajas L, Canale A (2017). ***BNPMIXcluster**: Bayesian Nonparametric Model for Clustering with Mixed Scale Variables*. R package version 1.2.4, URL https://CRAN.R-project.org/package=BNPMIXcluster.

Carmona C, Nieto-Barajas L, Canale A (2019). "Model-based approach for household clustering with mixed scale variables." *Advances in Data Analysis and Classification*, **13**(2), 559–583. doi:10.1007/s11634-018-0313-6.

De Blasi P, Favaro S, Lijoi A, Mena RH, Prünster I, Ruggiero M (2015). "Are Gibbs-Type Priors the Most Natural Generalization of the Dirichlet Process?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(2), 212–229. doi:10.1109/TPAMI.2013.217.

De Iorio M, Müller P, Rosner GL, MacEachern SN (2004). "An ANOVA Model for Dependent Random Measures." *Journal of the American Statistical Association*, **99**(465), 205–215. doi:10.1198/016214504000000205.

Dunson DB, Pillai N, Park JH (2007). "Bayesian density regression." *Journal of the Royal Statistical Society B*, **69**(2), 163–183. doi:10.1111/j.1467-9868.2007.00582.x.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with high-performance C++ linear algebra." *Computational Statistics and Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Escobar MD, West M (1995). "Bayesian Density Estimation and Inference Using Mixtures." *Journal of the American Statistical Association*, **90**(430), 577–588. doi:10.2307/2291069.

Ferguson T (1973). "A Bayesian Analysis of some Nonparametric Problems." *The Annals of Statistics*, **1**(2), 209–230. doi:10.1214/aos/1176342360.

Ferguson TS, Klass MJ (1972). "A Representation of Independent Increment Processes Without Gaussian Components." *The Annals of Mathematical Statistics*, **43**(5), 1634–1643. doi:10.1214/aoms/1177692395.

Foti N, Williamson S (2015). "A Survey of Non-Exchangeable Priors for Bayesian Nonparametric Models." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**. doi:10.1109/TPAMI.2013.224.

Hjort NL, Holmes C, Müller P, Walker SG (2010). *Bayesian Nonparametrics*, volume 28. Cambridge University Press.

Hughes M, Kim DI, Sudderth E (2015). "Reliable and scalable variational inference for the hierarchical dirichlet process." In *Artificial Intelligence and Statistics*, pp. 370–378.

Ishwaran H, Zarepour M (2000). "Markov Chain Monte Carlo in Approximate Dirichlet and Beta Two-Parameter Process Hierarchical Models." *Biometrika*, **87**(2), 371–390. doi:10.1093/biomet/87.2.371.

Jara A, Hanson T, Quintana F, Müller P, Rosner G (2011). "**DPpackage**: Bayesian Semi- and Nonparametric Modeling in R." *Journal of Statistical Software*, **40**(5), 1–30. doi:10.18637/jss.v040.i05.

Kalli M, Griffin J, Walker S (2011). "Slice sampling mixture models." *Statistics and Computing*, **21**, 93–105. doi:10.1007/s11222-009-9150-y.

Klebanoff M (2009). "The Collaborative Perinatal Project: A 50-year retrospective." *Paediatric and perinatal epidemiology*, **23**, 2–8. doi:10.1111/j.1365-3016.2008.00984.x.

Lijoi A, Mena RH, Prünster I (2005a). "Bayesian Nonparametric Analysis for a Generalized Dirichlet Process Prior." *Statistical Inference for Stochastic Processes*, **8**, 283–309. doi:10.1007/s11203-005-6071-z.

Lijoi A, Mena RH, Prünster I (2005b). "Hierarchical Mixture Modeling With Normalized Inverse-Gaussian Priors." *Journal of the American Statistical Association*, **100**(472), 1278–1291. doi:10.1198/016214505000000132.

Lijoi A, Mena RH, Prünster I (2007). "Controlling the Reinforcement in Bayesian Non-Parametric Mixture Models." *Journal of the Royal Statistical Society B*, **69**(4), 715–740. doi:10.1111/j.1467-9868.2007.00609.x.

Lijoi A, Nipoti B, Prünster I (2014). "Bayesian Inference with Dependent Normalized Completely Random Measures." *Bernoulli*, **20**(3), 1260–1291. doi:10.3150/13-BEJ521.

Liverani S, Hastie DI, Azizi L, Papathomas M, Richardson S (2015). "**PReMiuM**: An R Package for Profile Regression Mixture Models Using Dirichlet Processes." *Journal of statistical software*, **64**(7), 1–30. doi:10.18637/jss.v064.i07.

Lo AY (1984). "On a Class of Bayesian Nonparametric Estimates: I. Density Estimates." *The Annals of Statistics*, **12**(1), 351–357. doi:10.1214/aos/1176346412.

Longnecker MP, Klebanoff MA, Zhou H, Brock JW (2001). "Association Between Maternal Serum Concentration of the DDT Metabolite DDE and Preterm and Small-for-Gestational-Age Babies at Birth." *The Lancet*, **358**(9276), 110–114. doi:10.1016/S0140-6736(01)05329-6.

Müller P, Erkanli A, West M (1996). "Bayesian Curve Fitting Using Multivariate Normal Mixtures." *Biometrika*, **83**(1), 67–79. doi:10.1093/biomet/83.1.67.

Müller P, Quintana FA, Jara A, Hanson T (2015). *Bayesian Nonparametric Data Analysis*, volume 18. Springer-Verlag.

Neal RM (2000). "Markov Chain Sampling Methods for Dirichlet Process Mixture Models." *Journal of Computational and Graphical Statistics*, **9**(2), 249–265. doi:10.1080/10618600.2000.10474879.

Nieto-Barajas LE, Prünster I, Walker SG (2004). "Normalized Random Measures Driven by Increasing Additive Processes." *The Annals of Statistics*, **32**(6), 2343–2360. doi:10.1214/009053604000000625.

Perman M, Pitman J, Yor M (1992). "Size-Biased Sampling of Poisson Point Processes and Excursions." *Probability Theory and Related Fields*, **92**(1), 21–39. doi:10.1007/BF01205234.

Pitman J (1995). "Exchangeable and Partially Exchangeable Random Partitions." *Probability Theory and Related Fields*, **102**, 145–158. doi:10.1007/BF01213386.

Pitman J (1996). "Some Developments of the Blackwell-Macqueen Urn Scheme." *Lecture Notes-Monograph Series*, **30**, 245–267. doi:10.1214/lnms/1215453576.

Pitman J, Yor M (1997). "The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator." *The Annals of Probability*, **25**(2), 855–900. doi:10.1214/aop/1024404422.

Regazzini E, Lijoi A, Prünster I (2003). "Distributional results for means of normalized random measures with independent increments." *The Annals of Statistics*, **31**(2), 560–585. doi:10.1214/aos/1051027881.

Ross GJ, Markwick D (2020). *dirichletprocess: Build Dirichlet Process Objects for Bayesian Modelling*. R package version 0.4.0, URL https://CRAN.R-project.org/package=dirichletprocess.

Sethuraman J (1994). "A Constructive Definition of Dirichlet Priors." *Statistica Sinica*, **4**(2), 639–650. URL http://www.jstor.org/stable/24305538.

Shen W, Tokdar ST, Ghosal S (2013). "Adaptive Bayesian Multivariate Density Estimation with Dirichlet Mixtures." *Biometrika*, **100**(3), 623–640. doi:10.1093/biomet/ast015.

Tank A, Foti N, Fox E (2015). "Streaming variational inference for Bayesian nonparametric mixture models." In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pp. 968–976.

Wade S, Ghahramani Z (2018). "Bayesian Cluster Analysis: Point Estimation and Credible Balls." *Bayesian Anal.*, **13**(2), 559–626. doi:10.1214/17-BA1073.

Walker SG (2007). "Sampling the Dirichlet Mixture Model with Slices." *Communications in Statistics - Simulation and Computation*, **36**(1), 45–54. doi:10.1080/03610910601096262.

West M (1991). *Modelling with Mixtures*. Institute of Statistics and Decision Sciences, Duke University.

Table 3: Mixture models implemented in the package **BNPmix** and availability of the same in other R packages for BNP inference via mixtures: ✓ indicates the model is implemented, ✗ the model is not. The structure of the Gaussian kernel can be location (L), location-scale (LS) and, only for the multivariate case, location-scale with diagonal covariance matrix (DLS) as described in Section 2.

package	mixing process		Gaussian kernel structure					dependent mixture	
			univariate		multivariate				
	DP	PY	L	LS	L	LS	DLS	GM-DDP	regression
BNPmix	✓	✓	✓	✓	✓	✓	✓	✓	✓
DPpackage	✓	✗	✓	✓	✓	✓	✗	✗	✓
PreMiuM	✓	✓[1]	✗	✓	✗	✓	✗	✗	✓
BNPdensity	✓	✗	✓	✓	✗	✗	✗	✗	✗
dirichletprocess	✓	✗	✗	✓	✗	✓	✗	✗	✗
BNPMIXcluster	✓	✓	✗	✗	✗	✗	✗	✗	✗
msBP	✗	✗	✗	✗	✗	✗	✗	✗	✗

[1] PY mixtures are implemented only by approximating the PY process via deterministic truncation.

A. Appendix

A.1. Packages comparison

We compare state-of-the-art R packages for BNP inference. To this end, a list of the models implemented in **BNPmix** is presented in Table 3 along with the availability of the same models in other packages. Similarly, Table 4 compares the main technical features of **BNPmix** with those of other packages.

A.2. Base measures and hyperdistributions

Table 5 summarizes the base measures and the hyperdistributions on the base measures parameters, for the specifications of univariate and multivariate PY mixture models with Gaussian kernels, described in Section 2. The focus of the table is on the models that can be fitted by using `PYdensity`. The top part of the table refers to the models without hyperpriors on the parameters of the base measure (`hyper = FALSE`), the bottom part (`hyper = TRUE`) instead describes the specification of the hyperpriors.

For the sake of clarity and in order to avoid any ambiguity, Table 6 reports the parametrization of the relevant probability distributions, adopted throughout the paper.

Affiliation:

Riccardo Corradin
Department of Economics, Management and Statistics
University of Milano-Bicocca, Milan, Italy
E-mail: riccardo.corradin@unimib.it

Table 4: Main technical features of the **BNPmix** package and comparison with other R packages for BNP models using MCMC. The sampling methods considered in the table are marginal (Escobar and West 1995), dependent slice efficient (Kalli *et al.* 2011), independent slice efficient (Kalli *et al.* 2011), ICS (Canale *et al.* 2020), marginal algorithm 8 (Neal 2000), truncated sampler (Ishwaran and Zarepour 2000), label switching move (Liverani *et al.* 2015) for the truncated sampler, Ferguson and Klass algorithm (Ferguson and Klass 1972; Barrios *et al.* 2013), and the slice sampler for multi-scale Bernstein polynomials (Canale and Dunson 2014).

package	sampling method	programming language		S3
		core functions	MCMC loop	system
BNPmix	Marginal Slice dependent Slice independent ICS	C++	C++	✓
DPpackage	Marginal Marginal algorithm 8	Fortran	Fortran	✗
PreMiuM	Truncated sampler Slice dependent (not for PY) Slice independent (not for PY) Label switching move	C++	C++	✗
BNPdensity	Ferguson and Klass	R	R	✓
dirichletprocess	Marginal Marginal algorithm 8	R	R	✓
BNPMIXcluster	Marginal algorithm 8	C++	R	✗
msBP	Slice sampler	C++	R	✗

Antonio Canale
Department of Statistics
University of Padova, Padova, Italy
E-mail: canale@stat.unipd.it

Bernardo Nipoti
Department of Economics, Management and Statistics
University of Milano-Bicocca, Milan, Italy
E-mail: bernardo.nipoti@unimib.it

Table 5: Base measures and optional hyperprior distributions on the parameters of the base measures, for location (L), location-scale (LS) and, only for the multivariate case, location-scale with diagonal covariance matrix (DLS) PY mixture models. As for the multivariate location-scale PY mixture model with diagonal covariance matrix, assume that $r = 1, \ldots, p$.

`hyper = FALSE`	Univariate	Multivariate
L	$\tilde{\mu}_j \sim \mathrm{N}(m_0, \sigma_0^2)$ $\sigma^2 \sim \mathrm{IGa}(a_0, b_0)$	$\tilde{\boldsymbol{\mu}}_j \sim \mathrm{N}(\boldsymbol{m}_0, \boldsymbol{S}_0)$ $\boldsymbol{\Sigma} \sim \mathrm{IW}(\nu_0, \boldsymbol{\Sigma}_0)$
LS	$\tilde{\mu}_j \mid \tilde{\sigma}_j^2 \sim \mathrm{N}(m_0, \tilde{\sigma}_j^2/k_0)$ $\tilde{\sigma}_j^2 \sim \mathrm{IGa}(a_0, b_0)$	$\tilde{\boldsymbol{\mu}}_j \mid \tilde{\boldsymbol{\Sigma}}_j \sim \mathrm{N}(\boldsymbol{m}_0, \tilde{\boldsymbol{\Sigma}}_j/k_0)$ $\tilde{\boldsymbol{\Sigma}}_j \sim \mathrm{IW}(\nu_0, \boldsymbol{\Sigma}_0)$
DLS		$\tilde{\mu}_{jr} \mid \tilde{\sigma}_{jr}^2 \sim \mathrm{N}(m_{0r}, \tilde{\sigma}_{jr}^2/k_{0r})$ $\tilde{\sigma}_{jr}^2 \sim \mathrm{IGa}(a_{0r}, b_{0r})$
`hyper = TRUE`	Univariate	Multivariate
L	$m_0 \mid \sigma_0^2 \sim \mathrm{N}(m1, \sigma_0^2/k1)$ $\sigma_0^2 \sim \mathrm{IGa}(a_1, b_1)$	$\boldsymbol{m}_0 \mid \boldsymbol{S}_0 \sim \mathrm{N}(m1, \boldsymbol{S}_0/k1)$ $\boldsymbol{S}_0 \sim \mathrm{IW}(\lambda_1, \boldsymbol{\Lambda}_1)$
LS	$m_0 \sim \mathrm{N}(m_1, \sigma_1^2)$ $k_0 \sim \mathrm{Ga}(\tau_1, \zeta_1)$ $b_0 \sim \mathrm{Ga}(a_1, b_1)$	$\boldsymbol{m}_0 \sim \mathrm{N}(\boldsymbol{m}_1, \boldsymbol{S}_1)$ $k_0 \sim \mathrm{Ga}(\tau_1, \zeta_1)$ $\boldsymbol{\Sigma}_0 \sim \mathrm{W}(\nu_1, \boldsymbol{\Sigma}_1)$
DLS		$m_{0r} \sim \mathrm{N}(m_{1r}, \sigma_{1r}^2)$ $k_{0r} \sim \mathrm{Ga}(\tau_{1r}, \zeta_{1r})$ $b_{0r} \sim \mathrm{Ga}(a_{1r}, b_{1r})$

Table 6: Parametrizations of the probability distributions used throughout the paper. Multivariate distributions are assumed of dimension p. As for the Wishart and the inverse Wishart distribution, only the (j, l)-th element of the covariance matrix is reported.

Distribution	Notation	Parameters	Expectation	Variance/Covariance
Univ. normal	$\mathrm{N}(\mu, \sigma^2)$	μ (location), σ^2 (scale)	μ	σ^2
Mult. normal	$\mathrm{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\boldsymbol{\mu}$ (location), $\boldsymbol{\Sigma}$ (scale)	$\boldsymbol{\mu}$	$\boldsymbol{\Sigma}$
Gamma	$\mathrm{Ga}(a, b)$	a (shape), b (rate)	a/b	a/b^2
Inverse gamma	$\mathrm{IGa}(a, b)$	a (shape), b (scale)	$\frac{b}{a-1}$	$\frac{(b-1)^2}{(a-1)^2(b-1)}$
Wishart	$\mathrm{W}(\nu, \boldsymbol{S})$	ν (d.o.f.), $\boldsymbol{S}$ (scale)	$\nu\boldsymbol{S}$	$\nu(\boldsymbol{S}_{jl}^2 + \boldsymbol{S}_{jj}\boldsymbol{S}_{ll})$
Inverse Wishart	$\mathrm{IW}(\nu, \boldsymbol{S})$	ν (d.o.f.), $\boldsymbol{S}$ (scale)	$\frac{\boldsymbol{S}}{\nu-p-1}$	$\frac{(\nu-p+1)\boldsymbol{S}_{jl}^2+(\nu-p-1)\boldsymbol{S}_{jj}\boldsymbol{S}_{ll}}{(\nu-p)(\nu-p-1)^2(\nu-p-3)}$