Bayesian Nonparametric Mixture Models Using
NIMBLE

Claudia Wehrhahn
Department of Statistics
University of California, Santa Cruz
Santa Cruz, CA, 95064
cwehrhah@ucsc.edu

Abel Rodríguez
Department of Statistics
University of California, Santa Cruz
Santa Cruz, CA, 95064
abel@soe.ucsc.edu

Christopher J. Paciorek
Department of Statistics
University of California, Berkeley
Berkeley, CA, 94720
paciorek@stat.berkeley.edu

Abstract

This paper describes and illustrates new features of the NIMBLE computing environment (de Valpine et al., 2017) that enable simulation-based inference for general nonparametric Bayesian mixture models.

1 Introduction

The introduction of Markov chain Monte Carlo (MCMC) algorithms in the statistical literature (Gelfand & Smith, 1990) revolutionized the discipline by enabling the application of Bayesian methods to increasingly complex models. Approaches such as Gibbs sampling, random walk Metropolis-Hastings or Hamiltonian Monte Carlo provide general templates to derive algorithms that are applicable to large classes of statistical models. However, because of their generality, the application of these templates to specific problems can be time-consuming, as they require tedious model-specific derivations to specialize them to particular problems. This issue is compounded in the case of algorithms for nonparametric Bayesian models, which require careful bookkeeping and the manipulation of infinite dimensional objects. This often means that only practitioners with specialized training in statistics and/or machine learning are able to successfully apply non-parametric Bayesian methods in their work.

The challenge of enabling a more general audience to use Bayesian models has been traditionally tackled in two distinct ways. Starting with WinBUGS (Sturtz et al., 2005), a number of software packages have been introduced to enable automated inference for general statistical models. Examples include OpenBUGS (Lunn et al., 2009), JAGS (Plummer et al., 2003), Stan (Carpenter et al., 2017), Edward (Tran et al., 2017) and Turing (Ge et al., 2018). In particular, both Edward and Turing provide support for nonparametric Bayesian models. This type of software combines a probabilistic programming language that allows users to specify general hierarchical models, with a system that assigns inferential algorithms from among a few options following simple rules. A key shortcoming of this approach is that it usually restricts the type of algorithms that can be used for any specific model, potentially leading to suboptimal choices. An alternative approach involves the design of specialized packages that focus on very specific models and their associated algorithms. Examples of packages for fitting nonparametric Bayesian models include DPpackage (Jara et al., 2011), BNPdensity (Barros et al., 2017), and msBP (Canale et al., 2017) for the R environment (R Core Team 2018). Bnpy (Hughes & Sudderth, 2014) for Python (Van Rossum et al., 2007), and BNP.jl

Preprint. Work in progress.
We have recently added support for BNP mixture modeling to NIMBLE (see Section 10 of the NIMBLE manual). CRP($v$) and DP mixture models can be specified in NIMBLE using a (truncated) stick-breaking representation. The simplest such model is

\[ y_i \mid \theta_i, \phi \sim \psi(y_i \mid \theta_i, \phi), \quad \theta_i \mid G^{\text{ind}} \sim G, \quad G \mid \alpha, H_\eta \sim DP(\alpha, H_\eta), \quad i = 1, \ldots, n, \tag{1}\]

where \(\psi(\cdot \mid \theta, \phi)\) is a suitable kernel that depends on random effects \(\theta\) and fixed effects \(\phi\). \(\alpha\) is the concentration parameter of the Dirichlet process prior, and \(H_\eta\) is a parametric base distribution indexed by the vector of parameters \(\eta\). More sophisticated versions of the model include additional levels in the hierarchy, such as priors for the hyperparameters \(\phi, \alpha\), and \(\eta\).

Our implementation of models involving DP mixtures uses the Chinese Restaurant Process (CRP) representation (Blackwell & MacQueen [1973], Pitman [1995, 1996]). Introducing a vector of auxiliary variables \(z = (z_1, \ldots, z_n)\) that indicate which component of the mixture generated each observation, and integrating over the random measure \(G\), the model in (1) can be rewritten as

\[ y_i \mid z_i, \phi, \theta_1, \theta_2, \ldots \sim \psi(y_i \mid \theta_{z_i}, \phi), \quad z \mid \alpha \sim \text{CRP}(\alpha), \quad \theta_j \mid \eta \sim H_\eta, \quad i = 1, \ldots, n, \tag{2}\]

where CRP(\(\alpha\)) denotes the CRP distribution with concentration parameter \(\alpha\), with probability mass function

\[ p(z \mid \alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + n)} \alpha^{K(z)} \prod_k \Gamma(m_k(z)), \tag{3}\]

where \(K(z) \leq n\) is the number of unique values in the vector \(z\), and \(m_k(z)\) is the number of times the \(k\)-th unique value appears in \(z\).

Alternatively, DP mixture models can be specified in NIMBLE using a (truncated) stick-breaking representation of the random distribution \(G\) (Sethuraman [1994]):

\[ y_i \mid \xi_i, v_1, \theta_{1i}, \theta_{2i}, \ldots \sim \psi(y_i \mid \theta_{\xi_i}, \phi), \quad Pr(\xi_i = l \mid v) = v_l \prod_{m < l} (1 - v_m), \quad i = 1, \ldots, n, \tag{4}\]

where \(v_l \sim \text{Beta}(1, \alpha)\) for \(l = 1, \ldots, L - 1\) and \(v_L = 1\), while \(\theta_{li} \sim H_\eta\) for \(l = 1, \ldots, L\).

Each of the two representations leads to a different default choice for the MCMC algorithm. Formulations based on the CRP representation use a collapsed Gibbs sampler (Neal [2000]). Specifically, either algorithm 2 or algorithm 8 from (Neal [2000]) is used depending on whether \(\psi\) and \(H\) form a conjugate pair or not. When the truncated stick-breaking representation is used, a blocked Gibbs sampler is used (Ishwaran & James [2001, 2002]).
3 Illustration

We illustrate the use of NIMBLE for fitting nonparametric models in the context of a meta-analysis of the side effects of a formerly very popular drug for diabetes called Avandia. Here we only present the main highlights, a full description of the code used for this example can be found on the online supplementary materials. The question of interest is whether Avandia use increases the risk of myocardial infarction (heart attack). There are 48 studies (the 49th study in the data file is different in some ways and excluded here), each with treatment and control arms. The data corresponds to four vectors: \( n \) and \( x \) contain, respectively, the total number of patients and the number suffering from myocardial infarctions in the control group of each study, while vectors \( m \) and \( y \) contain similar information for patients receiving the drug Avandia. The model for the data takes the form

\[
x_i \mid \theta, \gamma_i \sim \text{Bin} \left( n_i, \frac{1}{1 + \exp\{-\gamma_i\}} \right), \quad y_i \mid \theta, \gamma_i \sim \text{Bin} \left( m_i, \frac{1}{1 + \exp\{-\theta + \gamma_i\}} \right),
\]

for \( i = 1, \ldots, 49 \). The random effects \( \gamma_1, \ldots, \gamma_J \) follow a Dirichlet process mixture with Gaussian kernels and a product of Gaussian and inverse gamma base distributions. The parameter \( \theta \) (which is the fixed effect quantifying the difference in risk between the control and treatment arms) is given a flat prior. The following \texttt{nimbleCode} function provides the specification of the model using the CRP representation of the model:

```nimble
codeBNP <-nimbleCode({
  for(i in 1:I) {
    y[i] ~ dbin(size = m[i], prob = q[i]) # avandia MIs
    x[i] ~ dbin(size = n[i], prob = p[i]) # control MIs
    q[i] <- expit(theta + gamma[i]) # Avandia log-odds
    p[i] <- expit(gamma[i]) # control log-odds
    gamma[i] ~ dnorm(mu[i], var = tau[i])
    mu[i] <- muTilde[xi[i]]
    tau[i] <- tauTilde[xi[i]]
    muTilde[i] ~ dnorm(mu0, sd = sd0)
    tauTilde[i] ~ dinvgamma(a0, b0)
    xi[1:I] ~ dCRP(alpha, size = I)
    alpha ~ dgamma(1, 1)
    mu0 ~ dflat()
    sd0 ~ dunif(0, 100)
    a0 ~ dunif(0, 100)
    b0 ~ dunif(0, 100)
    theta ~ dflat()})
```

Figure 1: Left: Posterior distribution of the effect of Avandia on MI. Right: Nonparametric estimate of the random effects distribution
Most of the model specification above uses standard NIMBLE distributions. The main addition is the function dCRP, which assigns the CRP prior in (3) to the vector of indicators \( x_i \). Model compilation and execution then proceeds in the same way as for any other NIMBLE model. Since the CRP representation of a nonparametric mixture model integrates out the random mixing measure, general inferences for the distribution of the random effects is not directly available from the MCMC output. To address this gap we have implemented the sampling approach described in Gelfand & Kottas (2002) in the function getSamplesDPmeasure(). This function takes as its argument a compiled or uncompiled MCMC object, and generates samples for the weights and atoms of (a truncated version of) the underlying random measure.

Using the code presented in the online supplement we generate Figures 1. Note that the posterior for \( \theta \) is a unimodal, symmetric distribution centered around 0.3 and with little mass on negative values. This suggests that there is a positive overall difference in risk between the treatment and control arms, i.e., that Avandia may increase the risk of myocardial infarction. The random effect distribution is readily estimated using the getSamplesDPmeasure() function, and while it shows little evidence of non-normality, doing the nonparametric analysis has ensured robustness to the random effect specification.

The same model can also be estimated using a truncated stick-breaking representation:

```r
codeBNP <- nimbleCode({
  for(i in 1:I) {
    y[i] ~ dbin(size = m[i], prob = q[i]) # avandia MIs
    x[i] ~ dbin(size = n[i], prob = p[i]) # control MIs
    q[i] <- expit(theta + gamma[i]) # Avandia log-odds
    p[i] <- expit(gamma[i]) # control log-odds
    gamma[i] ~ dnorm(mu[i], var = tau[i])
    xi[i] ~ dcat(w[1:L])
    mu[i] ~ muTilde[xi[i]]
    tau[i] ~ tauTilde[xi[i]]
    muTilde[i] ~ dnorm(mu0, sd = sd0)
    tauTilde[i] ~ dinvgamma(a0, b0)
  }
  w[1:L] <- stick_breaking(v[1:(L-1)])
  for(i in 1:(L-1)){
    v[i] ~ dbeta(1, 1)
  }
  alpha ~ dgamma(1, 1)
  mu0 ~ dflat()
  sd0 ~ dunif(0, 100)
  a0 ~ dunif(0, 100)
  b0 ~ dunif(0, 100)
  theta ~ dflat()})
```

The function stick_breaking() builds the stick breaking weights from the stick-breaking ratios contained in the vector \( v \). Note that the construction is general and does not depend on the fact that the stick-breaking ratios are Beta distributed. Hence, the function allows for the implementation of more general stick-breaking priors.

### 4 Future work

We are currently working to extend NIMBLE to accommodate more general nonparametric priors and their associated species sampling models (e.g., Poisson-Dirichlet process, normalized random measures), as well as models for collections of distributions (e.g., hierarchical Dirichlet processes).

### 5 Acknowledgments

The authors were supported by Center for Research in Open Source Software at UCSC and by NSF/DMS-1622444 and NSF/CCF-1740850.
References


