Environ Ecol Stat (2016) 23:549–564
DOI 10.1007/s10651-016-0353-z

Efficient Markov chain Monte Carlo sampling for hierarchical hidden Markov models

Daniel Turek[1] · **Perry de Valpine**[1] · **Christopher J. Paciorek**[1]

Received: 22 January 2016 / Revised: 21 June 2016 / Published online: 21 July 2016

Abstract Traditional Markov chain Monte Carlo (MCMC) sampling of hidden Markov models (HMMs) involves latent states underlying an imperfect observation process, and generates posterior samples for top-level parameters concurrently with nuisance latent variables. When potentially many HMMs are embedded within a hierarchical model, this can result in prohibitively long MCMC runtimes. We study combinations of existing methods, which are shown to vastly improve computational efficiency for these hierarchical models while maintaining the modeling flexibility provided by embedded HMMs. The methods include discrete filtering of the HMM likelihood to remove latent states, reduced data representations, and a novel procedure for dynamic block sampling of posterior dimensions. The first two methods have been used in isolation in existing application-specific software, but are not generally available for incorporation in arbitrary model structures. Using the NIMBLE package for R, we develop and test combined computational approaches using three examples from ecological capture–recapture, although our methods are generally applicable to any embedded discrete HMMs. These combinations provide several orders of magnitude improvement in MCMC sampling efficiency, defined as the rate of generating effectively independent posterior samples. In addition to being computationally significant for this class of hierarchical models, this result underscores the potential for vast improvements to MCMC sampling efficiency which can result from combinations of known algorithms.

Handling Editor: Bryan F. J. Manly.

✉ Daniel Turek
danielturek@gmail.com

[1] University of California, Berkeley, 493 Evans Hall, Berkeley, CA 94720, USA

Keywords Capture–recapture · Effective sample size · Hidden Markov model · Hierarchical model · MCMC · NIMBLE · Sampling efficiency

1 Introduction

Hidden Markov models (HMMs) are widely applied for the analysis of time series data with incomplete or noisy observations together with stochastic system dynamics (Capp et al. 2006; Elliott et al. 2008). HMMs are used in a diverse range of application domains, with recent attention in areas of speech recognition and natural language processing (Gales and Young 2008). See MacDonald and Zucchini (1997) for a broad review of HMM applications in disciplines such as as medicine, finance, sociology, and climatology.

For a single discrete HMM, likelihood calculation involves summing over the distribution of a sequence of unknown latent states. This can be implemented either using standard direct filtering summations (e.g., Elliott et al. 2008, chapter 2) as part of either maximum likelihood or Bayesian analysis, or using Markov chain Monte Carlo (MCMC; Gilks 2005; Brooks et al. 2011) for Bayesian analysis. In the case of MCMC, the unknown state variables are included in MCMC sampling. However, it is often the case that one or more HMMs are embedded in a larger hierarchical model, perhaps accounting for explanatory variables of state transition probabilities or shared variation among multiple time series. In such cases practitioners may rely on MCMC to perform a Bayesian analysis, but they face a quandary of computational efficiency. If they use standard MCMC software, they often have no choice but to include the unknown latent state variables in MCMC sampling. For large models this can contribute hundreds or thousands of dimensions which require MCMC sampling, to the point of rendering this approach computationally impractical.

In theory there are computational tradeoffs between using MCMC and direct filtering summation when embedding HMMs in a larger hierarchical model, but these tradeoffs have not been explored empirically to date. Here we do so, by considering combinations of several existing computational methods for fitting HMMs. These methods include direct filtering to remove latent variables, using a reduced representation of observational data, and dynamic blocking of model parameters to achieve efficient MCMC sampling. We demonstrate that for large models, a combination of these techniques can yield several orders of magnitude improvement in sampling efficiency. This can make the analysis of such models practical, opening new possibilities for fitting complex hierarchical models.

As examples we draw upon capture–recapture and from ecological statistics (for a broad review, see Lebreton et al. 2009). In capture–recapture, each animal in a study generates a capture history over multiple observational periods. These data can be modeled using discrete HMMs, where latent states may simply represent "alive" or "dead", or in the case of multistate capture–recapture, are more detailed such as including reproductive status or location. We present a series of three examples of increasing complexity to study the tradeoffs in computational cost and MCMC mixing of several methodological approaches. Our examples include a simple Cormack–Jolly–

Seber capture–recapture model ("Dipper"), a simple multistate model ("Orchid"), and a larger multistate model with thousands of embedded HMMs ("Goose").

Some of the techniques we study are already supported in existing software, however only for specific applications or particular hierarchical structures. The standalone program MARK (White and Burnham 1999) is perhaps the industry leader for applied capture–recapture. MARK provides an application-specific MCMC algorithm for fitting multistate random effects capture–recapture models, which implements filtering over latent states to directly calculate model likelihoods. MARK also supports a reduced representation of datasets with repeated observations—known as an "m-array" in capture–recapture—however only for band-recovery analyses (Brownie et al. 1985). More recently, M-SURGE (Choquet et al. 2004) was developed specifically for multistate capture–recapture. M-SURGE supports numerical integration to remove latent states, although this is used exclusively for maximum likelihood estimation, and never in combination with MCMC. Furthermore, neither of these software programs expose these computational techniques for user control, nor are they applicable outside the domain of ecological capture–recapture.

We make use of the NIMBLE software for specifying hierarchical models and statistical algorithms (NIMBLE Development Team 2015) to generalize these computational approaches for embedded HMMs. We consider particular combinations of techniques using the flexible and transparent algorithmic control provided by NIMBLE. Although we draw upon capture–recapture for examples, our advances in efficient handling of HMMs can be embedded in any larger hierarchical model structure using NIMBLE. However, we focus attention on the computational methodologies rather than implementation details. For comparisons of interest we also include the widely used JAGS package (Plummer 2003) for MCMC.

2 Computational approaches to discrete HMMs

We begin with a general specification of discrete HMMs, and explain how multistate capture–recapture models may be framed in this context. We then provide the model likelihood, and present a variety of approaches to computing it in the context of MCMC estimation.

2.1 Discrete HMMs and multistate capture–recapture

Let row vector $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{ik})$ represent the i^{th} sequence of observations taken over sampling occasions $t = 1, \ldots, k$. Each $y_{it} \in \mathcal{Y}$, where $\mathcal{Y}$ is the finite set of observable states. Similarly, let row vector $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ik})$ be the sequence of true underlying states at occasions $t = 1, \ldots, k$, with $x_{it} \in \mathcal{X}$ for finite set of states $\mathcal{X}$. We will consider a total of n observed sequences, which comprise the rows of the observed data matrix $\boldsymbol{y} = (\boldsymbol{y}_1', \ldots, \boldsymbol{y}_n')'$ with true unknown state matrix $\boldsymbol{x} = (\boldsymbol{x}_1', \ldots, \boldsymbol{x}_n')'$, where $\boldsymbol{x}'$ denotes vector or matrix transpose. Finally, let $\boldsymbol{\theta}$ be a vector of all model parameters, which may also include random effects. Letting i take all values in $1, \ldots, n$, the general hierarchical model is

$$\begin{aligned} \boldsymbol{\Theta} &\sim p(\boldsymbol{\theta}) \\ X_{i1} &\sim f_{i1}(x_{i1} \mid \boldsymbol{\theta}) \\ X_{it} \mid X_{i,t-1} &\sim f_{it}(x_{it} \mid \boldsymbol{\theta}, x_{i,t-1}), \quad t = 2, \ldots, k \\ Y_{it} \mid X_{it} &\sim g_{it}(y_{it} \mid \boldsymbol{\theta}, x_{it}), \quad t = 1, \ldots, k. \end{aligned} \tag{1}$$

Here $p(\cdot)$ is a prior distribution for parameter vector $\boldsymbol{\theta}$, which may itself have one or more levels of stochastic interdependence. The distribution of each HMM initial state x_{i1} is $f_{i1}(\cdot \mid \boldsymbol{\theta})$. Markov state transition probabilities are given by $f_{it}(\cdot \mid \boldsymbol{\theta}, x_{i,t-1})$ and observation probabilities by $g_{it}(\cdot \mid \boldsymbol{\theta}, x_{it})$.

Discrete HMMs have long been applied in the area of ecological capture–recapture (e.g., Gimenez et al. 2007; King 2012; Langrock et al. 2012). In this context, a set of n distinct animals is monitored for k sampling occasions. Each $\boldsymbol{y}_i$ represents the observation history of animal i, for $i = 1, \ldots, n$, which can be modeled using HMMs as in (1). The set of observable states $\mathcal{Y}$ may include a state to represent "unobserved". Since all n animals are not typically observed on occasion $t = 1$, each embedded HMM will "begin" at the sampling period corresponding to the first genuine observation of that animal.

2.2 Model likelihood

We now provide the model likelihood for the general HMM formulation in (1), which is used in the Bayesian estimation procedures described next. We begin with the likelihood contribution from a single observation history,

$$L(\boldsymbol{\theta} \mid \boldsymbol{y}_i) = \sum_{\boldsymbol{x}_i \in \mathcal{X}^k} f_{i1}(x_{i1} \mid \boldsymbol{\theta}) \left(\prod_{t=2}^{k} f_{it}(x_{it} \mid \boldsymbol{\theta}, x_{i,t-1}) \right) \left(\prod_{t=1}^{k} g_{it}(y_{it} \mid \boldsymbol{\theta}, x_{it}) \right), \tag{2}$$

where $\mathcal{X}^k$ denotes the standard k-fold Cartesian product of $\mathcal{X}$. Using the likelihood components in (2), the total model likelihood is

$$L(\boldsymbol{\theta} \mid \boldsymbol{y}) = \prod_{i=1}^{n} L(\boldsymbol{\theta} \mid \boldsymbol{y}_i).$$

2.3 Computational approaches

We now describe several computational approaches to applying Bayesian estimation to embedded HMMs. These strategies will form the basis for our comparisons, using examples from capture–recapture.

2.3.1 MCMC for latent states and parameters

One approach to Bayesian estimation is to perform MCMC sampling of both the model parameters and latent states; that is, to sample from the full posterior distribution

$p(\boldsymbol{\theta}, \boldsymbol{x} \mid \boldsymbol{y})$. Doing so makes use of Bayes law in the form:

$$p(\boldsymbol{\theta}, \boldsymbol{x} \mid \boldsymbol{y}) \propto p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(\boldsymbol{x}_i \mid \boldsymbol{\theta})\ p(\boldsymbol{y}_i \mid \boldsymbol{\theta}, \boldsymbol{x}_i).$$

Using this approach, the dimension of the MCMC sampling problem can be very large, since there can be up to nk latent state variables. Although we expect the MCMC update of each individual variable will be fast, sampling this large number of latent states incurs a significant computational cost and can result in slow MCMC mixing for latent states and parameters.

2.3.2 Filtering over latent states with MCMC for parameters

An alternate approach makes use of direct filtering to calculate the likelihood contribution of each observation history. This approach relies on the discrete HMM structure underlying each observed sequence $\boldsymbol{y}_i$ in (1). Doing so, we may perform MCMC sampling of the posterior distribution of $\boldsymbol{\theta}$ only, rather than $(\boldsymbol{\theta}, \boldsymbol{x})$ as in the latent state MCMC, and use filtering to calculate each $p(\boldsymbol{y}_i \mid \boldsymbol{\theta})$ as described in Elliott et al. (2008). The filtering MCMC approach makes use of Bayes law in the form:

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \boldsymbol{\theta}). \tag{3}$$

For a general discrete HMM as specified in (1), the filtering likelihood calculation proceeds as follows. Everything pertains to the ith observation history $\boldsymbol{y}_i$ and we omit subscripts i. All probabilities are conditional on $\boldsymbol{\theta}$, and we use $\boldsymbol{y}_{1:t}$ to represent the row vector $(y_1, \ldots, y_t)$. We begin by defining discrete distribution functions for the latent state at each time step, and the scalar-valued conditional likelihood:

$$\begin{aligned}
P_t(x) &= \Pr(X_t = x \mid \boldsymbol{y}_{1:t-1}) \\
&= \sum_{x_{t-1} \in \mathcal{X}} \Pr(X_t = x \mid X_{t-1} = x_{t-1}) \Pr(X_{t-1} = x_{t-1} \mid \boldsymbol{y}_{1:t-1}) \\
Q_t(x) &= \Pr(X_t = x \mid \boldsymbol{y}_{1:t}) \\
&= \Pr(X_t = x \mid \boldsymbol{y}_{1:t-1}) \Pr(Y_t = y_t \mid X_t = x) \,/\, \Pr(Y_t = y_t \mid \boldsymbol{y}_{1:t-1}) \\
L_t &= \Pr(Y_t = y_t \mid \boldsymbol{y}_{1:t-1}) \\
&= \sum_{x_t \in \mathcal{X}} \Pr(Y_t = y_t \mid X_t = x_t) \Pr(X_t = x_t \mid \boldsymbol{y}_{1:t-1}).
\end{aligned} \tag{4}$$

Mapping the elements of $\mathcal{X}$ to the indices $\{1, 2, \ldots, |\mathcal{X}|\}$, a bijection, we express each $\boldsymbol{P}_t$ and $\boldsymbol{Q}_t$ as column vectors of length $|\mathcal{X}|$. Define $|\mathcal{X}| \times |\mathcal{X}|$ state transition matrices $\boldsymbol{T}_t$ as having (i, j) element $\Pr(X_t = i \mid X_{t-1} = j)$. Similarly, define $|\mathcal{Y}| \times |\mathcal{X}|$ observation matrices $\boldsymbol{Z}_t$ with (i, j) element $\Pr(Y_t = i \mid X_t = j)$. The elements of each $\boldsymbol{T}_t$ and $\boldsymbol{Z}_t$ are defined by f_t and g_t, respectively, from (1). We rewrite (4) in matrix form as

$$\begin{aligned} \boldsymbol{P}_t &= \boldsymbol{T}_t\, \boldsymbol{Q}_{t-1}, & t \geq 2 \\ \boldsymbol{Q}_t &= \boldsymbol{Z}_t(y_t)' * \boldsymbol{P}_t \,/\, L_t, & t \geq 1 \\ L_t &= \boldsymbol{Z}_t(y_t)\, \boldsymbol{P}_t, & t \geq 1, \end{aligned} \tag{5}$$

where $\boldsymbol{A}(i)$ is the ith row of matrix $\boldsymbol{A}$, $*$ represents element-wise multiplication, and / represents scalar division. The initial latent state distribution $\boldsymbol{P}_1$ is specified by f_1 from (1), and all other $\boldsymbol{P}_t$, $\boldsymbol{Q}_t$, and L_t terms are iteratively calculated using (5). The likelihood is calculated as $L(\boldsymbol{\theta} \mid \boldsymbol{y}) = L_1 L_2 \ldots L_k$. In related works (e.g., Kéry and Schaub 2012) matrices $\boldsymbol{T}_t$ and $\boldsymbol{Z}_t$ may be transposed, resulting only in notational changes.

A simplification of this filtering algorithm is possible for the case of single-state capture–recapture with one absorbing state. Once an animal is deceased, it is guaranteed to remain in that state thereafter, where $\mathcal{X} = \{\text{“alive”}, \text{“dead”}\}$ and $\mathcal{Y} = \{\text{“seen”}, \text{“not seen”}\}$. In this context we can express the likelihood of a capture history in terms of survival probabilities $\phi_t = \Pr(X_t = \text{“alive”} \mid X_{t-1} = \text{“alive”})$ and detection probabilities $p_t = \Pr(Y_t = \text{“seen”} \mid X_t = \text{“alive”})$ as

$$L(\boldsymbol{\theta} \mid \boldsymbol{y}) = \left(\prod_{t=1}^{t_{\text{final}}-1} \phi_t \right) \left(\prod_{t=2}^{t_{\text{final}}} p_t^{y_t} (1 - p_t)^{1-y_t} \right) \chi_{t_{\text{final}}}, \tag{6}$$

where we numerically assign $y_t = \text{“seen”}$ as $y_t = 1$ and $y_t = \text{“not seen”}$ as $y_t = 0$, t_{final} is the time index of the final observed sighting (*i.e.*, $t_{\text{final}} = \max\{t \mid y_t = 1\}$), $\chi_k = 1$, and $\chi_t = 1 - \phi_t + \phi_t(1 - p_t)\chi_{t+1}$ for $t < k$ (Lebreton et al. 1992). Use of this simplified calculation for single-state capture–recapture will dramatically speed up likelihood evaluations relative to (5), since the likelihood is expressed in closed form.

These filtering algorithms numerically integrate over sequences of latent states to directly calculate model likelihoods, removing the need to perform MCMC sampling of these latent variables. However, the MCMC sampling step for each component of $\boldsymbol{\theta}$ now requires application of a filtering algorithm for each observation history. Thus, this approach reduces the dimensionality of the MCMC sampling problem, but at the cost of increased computational complexity of each MCMC iteration.

2.3.3 Filtering MCMC with a reduced representation of the dataset

A further specialized approach arises when there are repeated instances of identical observation histories in the observed dataset $\boldsymbol{y}$. That is, multiple distinct individuals exhibited identical observation histories over the k observational periods. This may occur when n is large relative to $|\mathcal{Y}|$ and k, or when particular observation histories are common. Let n^* be the number of unique observation histories in the original dataset $\boldsymbol{y}$. We define a reduced representation $(\boldsymbol{y}^*, \boldsymbol{m}^*)$, where $\boldsymbol{y}^*$ contains the n^* unique histories appearing in $\boldsymbol{y}$. An accompanying vector of multiplicities $\boldsymbol{m}^*$ indicates how many times each unique history appears in the original dataset, where history $\boldsymbol{y}_i^*$ occurs in $\boldsymbol{y}$ a total of m_i^* times, for $i = 1, \ldots, n^*$.

Using this reduced representation, we can express (3) such that the likelihood of each unique observation history is calculated only once. This computational approach makes use of Bayes law in the form:

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = p(\boldsymbol{\theta}) \prod_{i=1}^{n^*} p(\boldsymbol{y}_i^* \mid \boldsymbol{\theta})^{m_i^*}. \tag{7}$$

Computing according to (7) requires only n^* applications of the filtering likelihood calculation, rather than n applications when using the filtering MCMC approach on the full dataset. We expect to this provide an approximate factor of n/n^* improvement in computational efficiency relative to the filtering MCMC on the original dataset.

2.3.4 Filtering MCMC with block sampling

As a final approach, we consider joint (a.k.a. block) MCMC sampling of model parameters (Roberts and Sahu 1997). In the case of correlated posteriors, it is well known that block sampling of highly-correlated parameter dimensions can result in improved MCMC mixing (e.g., Liu et al. 1994). The general problem of determining posterior dimensions for block sampling is difficult, as a practitioner cannot reliably guess what blocking arrangement will result in efficient MCMC sampling. Further, existing literature on the efficiency of block sampling generally only considers the mixing properties of univariate versus block sampling, and fails to consider computational demands (Mengersen and Tweedie 1996; Roberts and Tweedie 1996; Roberts et al. 1997, among others).

We make use of NIMBLE's automated procedure for determining an efficient problem-specific block sampling MCMC algorithm, which exemplifies how the flexibility and programmability of NIMBLE facilitates a higher level of algorithmic control than other statistical software packages (such as MARK or JAGS). This procedure dynamically determines a partition of the model parameters which results in efficient MCMC sampling, which otherwise may be difficult to determine due to the combinatorical explosion of possibilities. MCMC efficiency is defined as the number of effectively independent posterior samples generated per second of algorithm runtime, which balances improvements in MCMC mixing with computational requirements. This automated blocking procedure is described in detail in Turek et al. (2016).

The use of a block sampling strategy can be combined with filtering over latent states. Under this approach we use the filtering algorithms already described to integrate out the latent states, and require MCMC sampling for the model parameters. We use a dynamically determined block sampling strategy for the MCMC sampling of these parameters.

3 Capture–recapture example models

We use three capture–recapture examples representing different levels of complexity to asses performance of the various computational approaches to MCMC estimation. The

first is the well-studied European Dipper dataset, demonstrating single-state capture–recapture. The second is a multistate capture–recapture dataset of observations of a flowering orchid. This is considered multistate data since the orchids may be observed in multiple distinct states, in addition to the possibility of "not seen". The third and largest dataset is also a multistate example, representing observations of Canadian Geese at various locations.

3.1 Dipper model

The European Dipper (*Cinclus cinclus*) dataset has been analyzed extensively in the literature (Marzolin 1988; Lebreton et al. 1992; Gimenez et al. 2007; Royle 2008; Amstrup et al. 2010, among numerous others), and may be considered a canonical example of capture–recapture. For simplicity, we do not make use of a covariate reflecting gender or the distinction of flood years as in Lebreton et al. (1992).

The dataset consists of $n = 294$ sighting histories collected over $k = 7$ annual sighting occasions. The set of latent states is $\mathcal{X} = \{$"alive", "dead"$\}$ and the set of observable states is $\mathcal{Y} = \{$"seen", "not seen"$\}$. For computation, we use the numerical assignments $x = 1$ for "alive", $x = 0$ for "dead", $y = 1$ for "seen", and $y = 0$ for "not seen".

The model is parameterized by annual probability of survival, ϕ, and probability of detection, p, which are assumed to be constant among all sampling occasions and individuals. This reflects the most basic Cormack–Jolly–Seber model structure (Jolly 1965; Seber 1965), typically denoted as $\phi(.)\ p(.)$ to imply constant probabilities of survival and detection (e.g., Nichols and Pollock 1983). The hierarchical model specification is given below, which is a realization of the general structure provided in (1), where i assumes all values in $1, \ldots, n$.

$$\begin{aligned}
\phi &\sim \text{Uniform}(0, 1) \\
p &\sim \text{Uniform}(0, 1) \\
X_{i1} &= Y_{i1} = 1 \\
X_{it} \mid X_{i,t-1} &\sim \text{Bernoulli}(\phi\ x_{i,t-1}), \qquad t = 2, \ldots, k \\
Y_{it} \mid X_{it} &\sim \text{Bernoulli}(p\ x_{it}), \qquad t = 2, \ldots, k.
\end{aligned}$$

3.2 Orchid model

Our second example models sighting histories of the showy lady's slipper (*Cypripedium reginae*), a flowering variety of orchid which is native to north America. Here, the concept of "capture" has been generalized to observational sightings. One cannot observe these orchids with certainty due to a dormant state, in which the orchid is alive but not observable.

The Orchid model data consist of observational sighting histories of $n = 250$ unique flowers, collected over $k = 11$ annual observational periods. There are four latent states, $\mathcal{X} = \{$"vegetative", "flowering", "dormant", "dead"$\}$, but only three distinct observable states, $\mathcal{Y} = \{$"seen vegetative", "seen flowering", "not seen"$\}$ as we

cannot distinguish between dormant and deceased flowers. The presence of multiple distinct observable states (in addition to "not seen") classifies this as multistate capture–recapture. The full dataset is available in the supplementary material of Kéry and Schaub (2012).

Following Kéry and Gregg (2004) we include time-dependent survival probabilities ϕ_t, and state transition probabilities ψ_{rs} between the three living states. We use an uninformative Dirichlet prior distribution for each set $\{\psi_{1s}, \psi_{2s}, \psi_{3s}\}$, implemented using elemental Gamma(1, 1) hyperpriors as in Royle and Dorazio (2008). As flowers in the dormant state are never observed and there is no mis-identification of flowers in the vegetative or flowering states, the observation matrix $\boldsymbol{Z}$ is deterministic. In the model specification below, latent states $\boldsymbol{x}_{it}$ are represented as binary column vectors, and i assumes all values in $1, \ldots, n$.

$$\begin{aligned}
\phi_t &\sim \text{Uniform}(0, 1), && t = 2, \ldots, 11 \\
\{\psi_{1s}, \psi_{2s}, \psi_{3s}\} &\sim \text{Dirichlet}(\boldsymbol{\alpha} = \{1, 1, 1\}), && s = 1, 2, 3 \\
X_{i1} &= y_{i1} \\
X_{it} \mid X_{i,t-1} &\sim \text{Categorical}(\boldsymbol{p} = \boldsymbol{T}_t\, \boldsymbol{x}_{i,t-1}), && t = 2, \ldots, k \\
Y_{it} \mid X_{it} &\sim \text{Categorical}(\boldsymbol{p} = \boldsymbol{Z}\, \boldsymbol{x}_{it}), && t = 1, \ldots, k.
\end{aligned}$$

which makes use of state transition matrices

$$\boldsymbol{T}_t = \begin{bmatrix} \phi_t\psi_{11} & \phi_t\psi_{12} & \phi_t\psi_{13} & 0 \\ \phi_t\psi_{21} & \phi_t\psi_{22} & \phi_t\psi_{23} & 0 \\ \phi_t\psi_{31} & \phi_t\psi_{32} & \phi_t\psi_{33} & 0 \\ 1-\phi_t & 1-\phi_t & 1-\phi_t & 1 \end{bmatrix},$$

and constant observation matrix

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

3.3 Goose model

The multistate Goose model tracks $n = 11{,}200$ Canadian Geese (*Branta canadensis*) between three distinct locations over $k = 4$ years. Latent states $\mathcal{X} =$ {"site A", "site B", "site C", "dead"}, with observable states $\mathcal{Y} =$ {"seen at A", "seen at B", "seen at C", "not seen"}. There exists a large number of identical sighting histories among the 11,200 geese, allowing a reduced representation using only the $n^* = 153$ unique sighting histories. The complete dataset can be found in Amstrup et al. (2010).

Following Amstrup et al. (2010), we include site-dependent survival probabilities, and both time- and site-dependent geographic transition probabilities and probabilities of detection. We use uninformative priors for all parameters, including Dirichlet priors for each set of geographic transition probabilities. Subsequent works (e.g., McCrea and

Morgan 2011) have shown improved fits using more elaborate models for these data, but our purpose is to compare computational efficiency. We desire high efficiency regardless of model fit, so the particular choice of model is tangential to our main points. The hierarchical specification is given below, where i assumes all values in $1, \ldots, n$.

$$\begin{aligned}
\phi_r &\sim \text{Uniform}(0, 1), && r = 1, 2, 3 \\
\{\psi_{1st}, \psi_{2st}, \psi_{3st}\} &\sim \text{Dirichlet}(\boldsymbol{\alpha} = \{1, 1, 1\}), && s = 1, 2, 3, \quad t = 2, 3, 4 \\
p_{rt} &\sim \text{Uniform}(0, 1), && r = 1, 2, 3, \quad t = 1, 2, 3, 4 \\
X_{i1} &= y_{i1} \\
X_{it} \mid X_{i,t-1} &\sim \text{Categorical}(\boldsymbol{p} = \boldsymbol{T}_t \, \boldsymbol{x}_{i,t-1}), && t = 2, \ldots, k \\
Y_{it} \mid X_{it} &\sim \text{Categorical}(\boldsymbol{p} = \boldsymbol{Z}_t \, \boldsymbol{x}_{it}), && t = 1, \ldots, k.
\end{aligned}$$

which makes use of state transition matrices

$$\boldsymbol{T}_t = \begin{bmatrix} \phi_1\psi_{11t} & \phi_2\psi_{12t} & \phi_3\psi_{13t} & 0 \\ \phi_1\psi_{21t} & \phi_2\psi_{22t} & \phi_3\psi_{23t} & 0 \\ \phi_1\psi_{31t} & \phi_2\psi_{32t} & \phi_3\psi_{33t} & 0 \\ 1-\phi_1 & 1-\phi_2 & 1-\phi_3 & 1 \end{bmatrix},$$

and observation matrices

$$\boldsymbol{Z}_t = \begin{bmatrix} p_{1t} & 0 & 0 & 0 \\ 0 & p_{2t} & 0 & 0 \\ 0 & 0 & p_{3t} & 0 \\ 1-p_{1t} & 1-p_{2t} & 1-p_{3t} & 1 \end{bmatrix}.$$

4 Performance results

We now present the performance of various computational strategies for MCMC estimation applied to the three example capture–recapture models. We do not present posterior results, but instead only the algorithmic efficiencies of each computational approach to generating these. For each, the posterior results of top-level parameters closely agree with existing published analyses of the same datasets and models (Lebreton et al. 1992; Kéry and Schaub 2012; Amstrup et al. 2010), which provides validation of our computational methodologies.

We include results for the following computational strategies MCMC estimation: latent state MCMC ("Latent State") where model parameters and latent states undergo MCMC sampling, filtering MCMC ("Filter") in which we filter over latent states and only top-level parameters undergo MCMC sampling, and a combination of filtering and blocking ("Filter & Block") in which a customized blocking strategy is used for MCMC sampling of top-level parameters. When appropriate, we also use a reduced representation ("RR") of the dataset.

We use the NIMBLE package for R to generate and execute MCMC algorithms, as the algorithmic flexibility it provides facilitates these computational approaches.

The use of user-defined distribution functions in NIMBLE allows us to incorporate the filtering algorithms (5) and (6) directly into a hierarchical model specification. The generic discrete HMM filtering procedure described in (5) is used for filtering, or when permitted by the model structure we instead use the closed form likelihood calculation given in (6). NIMBLE also provides the automated parameter blocking procedure (Turek et al. 2016) we use to generate problem-specific parameter blocking strategies for MCMC sampling.

We define the efficiency of an MCMC algorithm in terms of the number of effectively independent posterior samples produced per second of algorithm runtime. This metric is denoted as effective samples per second (ESPS), and we will present both the minimum and mean ESPS among all model parameters. This metric balances the tradeoff between computationally fast algorithms which generate highly autocorrelated chains of posterior samples, versus algorithms which are more computationally demanding but result in lower posterior autocorrelation, which provides stronger inferential power.

All algorithm runtimes represent the time required to generate 100,000 posterior samples. When possible, we also provide comparisons with MCMC algorithms from the JAGS software package for R. All calculations are produced using single-threaded execution on an Intel Xeon E5-2609 processor (2.40 GHz), running under the Ubuntu Linux operating system. Datasets for each model and R scripts for all NIMBLE MCMC algorithms are available online at http://github.com/danielturek/HMM-MCMC.

4.1 Dipper model

For the Dipper model, use of the filtering MCMC compared to MCMC sampling of all discrete latent states yielded a 60-fold improvement in sampling efficiency in NIMBLE and a 15-fold improvement in JAGS (Fig. 1). The sampling efficiencies of ϕ and p are quite similar under each algorithm, although vary greatly between algorithms.

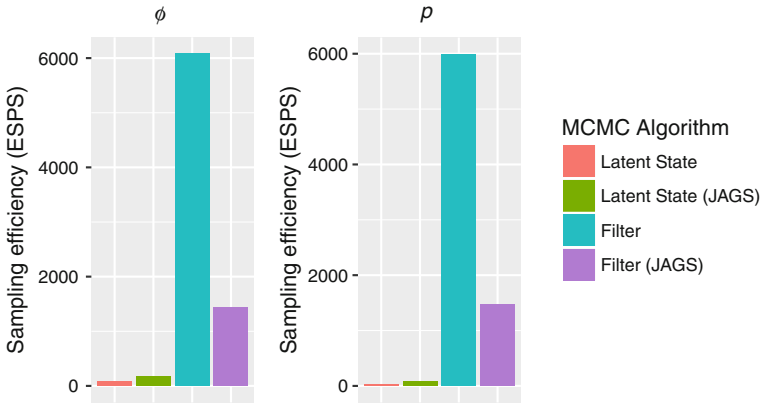

Fig. 1 Parameter sampling efficiencies for the Dipper model

The latent state MCMC requires MCMC sampling of 848 latent variables, in addition to the two top-level parameters. The performance of JAGS is slightly better, although both result in sampling efficiencies of roughly 100 ESPS for both parameters. NIMBLE and JAGS each require approximately 4 min to generate 100,000 samples. The filtering MCMC is implemented in NIMBLE according to (6), where only ϕ and p undergo MCMC sampling and runtime is reduced to 5 s. The mixing also improves relative to the latent state MCMC, yielding a sampling efficiency of roughly 6000 ESPS for both parameters, a 60-fold improvement.

For the Dipper model alone, we can also implement the filtering MCMC in JAGS. This is possible because (6) provides a closed form expression for the likelihood of each sighting history. This allows use of the "zeros-trick" (Lunn et al. 2012, pp. 204–206) where a general log-likelihood expression is incorporated into a model through the mean parameter of a Poisson distribution, using an artificial zero-valued observation. Using this technique reduces JAGS runtime to 30 s and increases sampling efficiency of both ϕ and p to approximately 1500 ESPS, a 15-fold improvement relative to the latent state MCMC. Although the underlying calculations are similar to those of NIMBLE's filtering MCMC, this approach requires the additional overhead of artificial model variables and observations.

4.2 Orchid model

For the multistate Orchid model, a combination of filtering over latent states and dynamic block sampling of parameters yielded a threefold improvement in sampling efficiency of the slowest mixing parameter, relative to the latent state MCMC (Fig. 2).

The latent state MCMC samples 2,157 latent variables in addition to 19 top-level parameters, which required 42 min to generate 100,000 samples. Efficiency results for the latent state MCMC are quite similar to the filtering MCMC, which required 36 min but with slightly inferior mixing. Both of these algorithms struggle to achieve good mixing among the nine state transition probabilities. We might expect triplets of these parameters to be highly correlated due to the Dirichlet prior imposing a sum-to-one

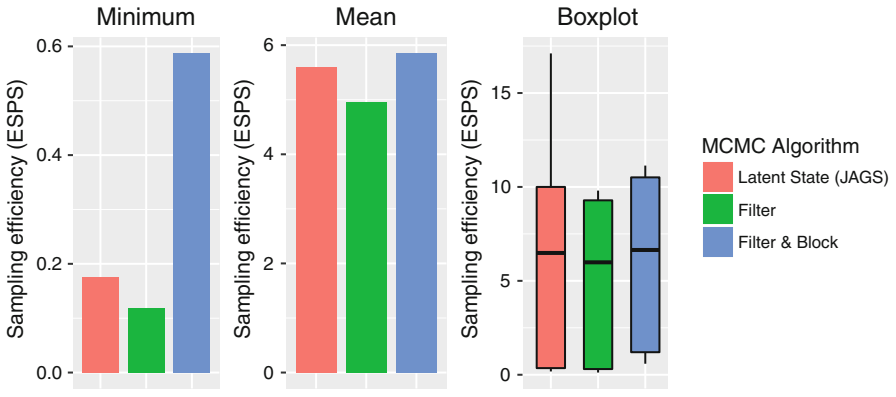

Fig. 2 Minimum, mean, and boxplots of parameter sampling efficiencies for the Orchid model

constraint, and indeed, examining the posterior correlations we find several instances of absolute pairwise posterior correlation greater than 0.9. Under the latent state and filtering MCMC algorithms, several state transition probabilities have sampling efficiencies between 0.1 and 0.3 ESPS, which dictates the minimum efficiencies shown in Fig. 2.

For the 19 parameters undergoing MCMC sampling, NIMBLE's automated parameter blocking procedure converges on two blocks each consisting of two state transition probabilities, and univariate sampling for the other 15 parameters. We observe that these pairs of transition probabilities have absolute posterior correlations of 0.98 and 0.97, the highest among all 19 parameters. Joint sampling according to this blocking scheme in combination with filtering over latent states results in a minimum sampling efficiency of 0.6 ESPS, representing a threefold improvement over the latent state MCMC.

4.3 Goose model

As the Goose model includes a large number of repeated sighting histories among the 11,200 geese, this model benefits from a reduced representation of the data using the 153 unique sighting histories. Applying the filtering MCMC to a reduced data representation produced a 70-fold improvement in sampling efficiency of the slowest mixing parameter, compared to the latent state MCMC (Fig. 3). An additional order of magnitude improvement was gained by applying dynamic blocking of model parameters.

The latent state MCMC requires sampling of 14,437 latent variables in addition to 21 top-level parameters. We cannot use a reduced data representation under the latent state approach, since for correct inference each of the 11,200 sighting histories must have a corresponding sequence of latent state variables. The latent state MCMC required approximately 24 h to generate 100,000 samples, yielding a minimum sampling efficiency of 0.0027 ESPS and a mean of 0.028 ESPS. This approach

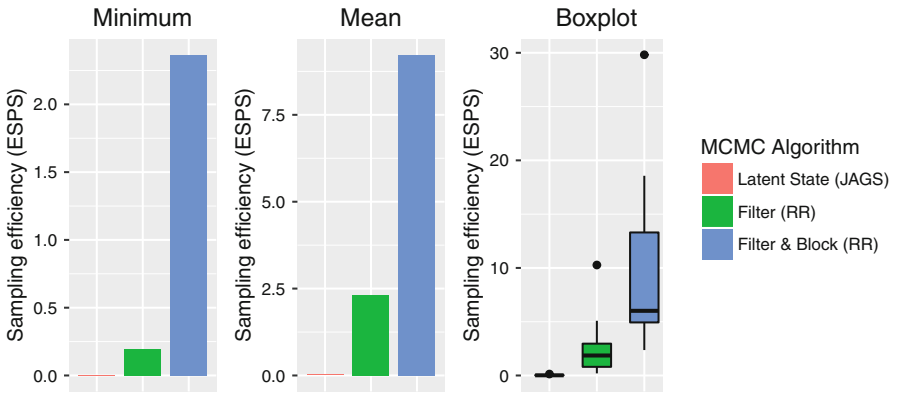

Fig. 3 Minimum, mean, and boxplots of parameter sampling efficiencies for the Goose model

can be deemed impractical, as this translates to generating ten effective samples (for the slowest mixing parameter) per hour.

Applying the filtering MCMC to a reduced data representation using the 153 unique sighting histories, the complete model likelihood is calculated according to (7), using (5) to calculate the likelihood of each unique history. Computation time is reduced to 20 min, which agrees with the expected speedup factor of $\frac{11,200}{153} \approx 73.2$. Mixing also improves to produce a minimum sampling efficiency of 0.20 ESPS, a 70-fold improvement relative to the latent state MCMC. This translates to 720 effective samples per hour, which may be considered practical.

NIMBLE's automated blocking procedure converges on seven blocks of parameters, ranging between two and five parameters each. These seven blocks include 20 of the 21 parameters, leaving only one parameter for univariate sampling. It is unlikely that a practitioner would discover this blocking scheme through expert opinion or trial and error. Runtime is comparable using this approach, but the joint sampling of correlated parameters gives a dramatic improvement in MCMC mixing. The minimum sampling efficiency improves by an additional order of magnitude to 2.4 ESPS, or generating over 8600 effective samples per hour. This represents nearly a 1000-fold improvement over the latent state MCMC.

5 Discussion

We have studied alternate computational approaches for MCMC sampling of hierarchical models which include embedded discrete HMMs. Traditional MCMC analysis of such models involves sampling the unknown latent states, whereas we propose filtering over latent states to calculate model likelihoods and limiting MCMC sampling to top-level parameters. This introduces a computational trade-off: reducing the dimensionality of MCMC sampling with the additional expense of filtering calculations. Through examples, we observe that worthwhile gains in sampling efficiency result from this approach.

Furthermore, the filtering MCMC permits a reduced representation of datasets with repeated observations. This simplification is not possible when using traditional latent state MCMC, since each (possibly duplicated) observational history requires its own sequence of latent states. When appropriate, combining our filtering MCMC with this reduced data representation provides an additional echelon of improvement in MCMC sampling efficiency, the extent of which is limited only by the degree of repetition in the initial data.

We note that the filtering MCMC approach forgoes generating posterior samples for latent states. In some analyses the distribution of latent variables at a particular observational periods may be of interest, or otherwise may be used (for example) to estimate longevity distributions. The inclusion of latent variables would also be necessary when used as explanatory variables in other parts of a hierarchical model (e.g., Risk et al. 2011), or in the case of individual-specific covariates. Our suggested approaches would not be appropriate in these analysis scenarios.

The analyses presented herein are facilitated by the NIMBLE package for R. NIMBLE allows user-defined distribution functions to be used directly in hierarchical

model specifications. We define a multivariate distribution function parametrized by state transition and observation matrices, where the probability density evaluation routine implements discrete filtering to calculate likelihood values. Models are specified using this distribution, which effectively embeds filtering into the model for the purposes of likelihood calculation. NIMBLE's MCMC engine may then be applied to the resulting model to achieve the filtering MCMC. We make use of NIMBLE's default MCMC as well as that resulting from automated parameter blocking. The distinction of allowing programmable models and statistical algorithms, as compared to other statistical software, makes such analyses possible in NIMBLE.

Acknowledgments This work was supported by the NSF under Grant DBI-1147230 and by support to DT from the Berkeley Institute for Data Science. We thank Marc Kéry, Byron Morgan, and Michael Schaub for reviewing earlier versions of the manuscript.

References

Amstrup SC, McDonald TL, Manly BFJ (2010) Handbook of capture–recapture analysis. Princeton University Press, Princeton, p 173

Brooks S et al (2011) Handbook of Markov chain Monte Carlo. CRC Press, Boca Raton, pp 3–47

Brownie C et al (1985) Statistical inference from band recovery data: a handbook. U.S. Department of the Interior, Fish and Wildlife Service

Capp O, Moulines E, Rydn T (2006) Inference in hidden Markov models. Springer Science & Business Media, Berlin

Choquet R et al (2004) M-SURGE: new software specifically designed for multistate capture–recapture models. Anim Biodivers Conserv 27(1):207–215

Elliott RJ, Aggoun L, Moore JB (2008) Hidden Markov models: estimation and control, vol 29. Springer Science & Business Media, Berlin

Gales M, Young S (2008) The application of hidden Markov models in speech recognition. Found Trends Signal Process 1(3):195–304

Gilks WR (2005) Markov Chain Monte Carlo. Encyclopedia of Biostatistics. Wiley, New York

Gimenez O et al (2007) State-space modelling of data on marked individuals. Ecol Model 206(34):431–438

Jolly GM (1965) Explicit estimates from capture–recapture data with both death and immigration-stochastic model. Biometrika 52(1/2):225–247

Kéry M, Gregg KB (2004) Demographic analysis of dormancy and survival in the terrestrial orchid Cypripedium reginae. J Ecol 92(4):686–695

Kéry M, Schaub M (2012) Bayesian population analysis using WinBUGS: a hierarchical perspective. Academic Press, San Diego, pp 261–265

King R (2012) A review of Bayesian state-space modelling of capture–recapture–recovery data. Interface Focus 2(2):190–204

Langrock R et al (2012) Flexible and practical modeling of animal telemetry data: hidden Markov models and extensions. Ecology 93(11):2336–2342

Lebreton J-D et al (1992) Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. Ecol Monogr 62(1):67–118 (JSTOR: 2937171)

Lebreton J-D et al (2009) Modeling individual animal histories with multistate capture–recapture models. Adv Ecol Res 41:87–173

Liu JS, Wong WH, Kong A (1994) Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. Biometrika 81(1):27–40

Lunn D et al (2012) The BUGS book: a practical introduction to Bayesian analysis. CRC Press, San Diego, p 402

MacDonald IL, Zucchini W (1997) Hidden Markov and other models for discrete-valued time series, vol 110. CRC Press, San Diego

Marzolin G (1988) Polygynie du Cincle plongeur (Cinclus cinclus) dans les côtes de Lorraine. Oiseau et la Revue Francaise d'Ornithologie 58(4):277–286

McCrea RS, Morgan BJ (2011) Multistate mark-recapture model selection using score tests. Biometrics 67(1):234–241
Mengersen KL, Tweedie RL (1996) Rates of convergence of the Hastings and Metropolis algorithms. Ann Stat 24(1):101–121
Nichols JD, Pollock KH (1983) Estimation methodology in contemporary small mammal capture–recapture studies. J Mammal 64(2):253–260
NIMBLE Development Team (2015) NIMBLE: an R package for programming with BUGS models, Version 0.5–1. http://r-nimble.org
Plummer M (2003) JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In: Proceedings of the 3rd international workshop on distributed statistical computing, vol 124. Vienna, p 125
Risk BB, De Valpine P, Beissinger SR (2011) A robust-design formulation of the incidence function model of metapopulation dynamics applied to two species of rails. Ecology 92(2):462–474
Roberts GO, Gelman A, Gilks WR (1997) Weak convergence and optimal scaling of random walk Metropolis algorithms. Ann Appl Probab 7(1):110–120
Roberts GO, Sahu SK (1997) Updating schemes, correlation structure, blocking and parameterization for the gibbs sampler. J R Stat Soc Ser B (Stat Methodol) 59(2):291–317
Roberts GO, Tweedie RL (1996) Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. Biometrika 83(1):95–110
Royle JA (2008) Modeling Individual effects in the CormackJollySeber model: a statespace formulation. Biometrics 64(2):364–370
Royle JA, Dorazio RM (2008) Hierarchical modeling and inference in ecology: the analysis of data from populations, metapopulations and communities. Academic Press, San Diego
Seber GA (1965) A note on the multiple-recapture census. Biometrika 52(1/2):249–259
Turek D et al (2016) Automated parameter blocking for efficient Markov-Chain Monte Carlo sampling. Bayesian Analysis (Advance Publication)
White GC, Burnham KP (1999) Program MARK: survival estimation from populations of marked animals. Bird Study 46(S1):S120–S139

Daniel Turek Daniel is part of an interdisciplinary team at the University of California, Berkeley, developing a statistical algorithmic package for use in R. Daniel's current research interests are in computational statistics, hierarchical model analysis, and MCMC sampling algorithms.

Perry de Valpine Perry is a population ecologist, mathematical modeler, and statistician at the University of California, Berkeley. He works primarily on biologically motivated population models to reach statistical conclusions about biological hypotheses.

Christopher J. Paciorek Chris is the statistical computing consultant, as well as a researcher and lecturer, in the Department of Statistics at the University of California, Berkeley. His statistical expertise is in the areas of Bayesian statistics and spatial statistics with primary application to environmental and public health research.