Methods in Ecology and Evolution

BRITISH ECOLOGICAL SOCIETY

Methods in Ecology and Evolution 2016 doi: 10.1111/2041-210X.12681

COMMENTARY

Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo

Cole C. Monnahan[1]*, James T. Thorson[2] and Trevor A. Branch[3]

[1]Quantitative Ecology and Resource Management, University of Washington, Box 352182, Seattle, WA 98195, USA; [2]Fisheries Resource Assessment and Monitoring Division, Northwest Fisheries Science Center, National Marine Fisheries Service, National Oceanic and Atmospheric Administration, 2725 Montlake Blvd. East, Seattle, WA 98112, USA; and [3]School of Aquatic and Fishery Sciences, University of Washington, Box 355020, Seattle, WA 98195, USA

Introduction

Bayesian inference is used widely throughout ecology, including population dynamics, genetics, community ecology and environmental impact assessment, among other subfields (Ellison 2004). In the Bayesian paradigm, the likelihood of the observed data is combined with prior distributions on parameters, resulting in a posterior probability distribution of parameters, from which inference is made (Gelman *et al.* 2014). Expectations of posterior quantities, such as means or quantiles, are commonly approximated using numerical techniques, with Markov chain Monte Carlo (MCMC) being the most common (Brooks *et al.* 2011).

The popularity of Bayesian inference grew particularly fast with the development of generic and flexible software platforms, with the BUGS family (here defined as BUGS, WINBUGS, OPENBUGS and JAGS; see Appendix A, Supporting Information) being by far the most common (Fig. 1). For a given model, BUGS automatically selects an MCMC algorithm and arguments controlling its behaviour (i.e. tuning parameters), where necessary. The analyst can thus focus on the model and scientific questions, rather than the mechanics of the underlying MCMC algorithms. As such, these platforms have been the workhorse for Bayesian analyses in ecology and other fields for the last 20 years.

However, for certain models, the time required for inference (run-time) using BUGS is prohibitively long. Long run-times often occur in BUGS because the underlying MCMC algorithms are inefficient, which is further compounded when the model needs to run many times during development, model selection (e.g. cross-validation; Hooten & Hobbs 2015), or simulation testing. These issues remain despite the increasing power of computers because data sets are increasing in size and models are becoming more complex (Bolker *et al.* 2013). At the same time, hierarchical modelling is becoming increasingly popular, as this type of model is widely recognized as a natural tool for formulating and thinking about problems in many ecological subfields (Royle & Dorazio 2008; Cressie *et al.* 2009; Thorson & Minto 2014). Thus, there is a need for alternatives to BUGS that are faster across a range of model size, complexity and hierarchical structure.

*Correspondence author. E-mail: monnahc@uw.edu

A family of MCMC algorithms called Hamiltonian Monte Carlo (HMC; Neal 2011) promises improved efficiency over the algorithms used by BUGS, but until recently have been slow to be adopted for two reasons. First, HMC requires precise gradients (i.e. derivatives of the log-posterior with respect to parameters), but analytical formulas are rare and numerical techniques are imprecise, particularly in higher dimensions. Secondly, the original HMC algorithm requires expert, hands-on tuning to be efficient (Neal 2011). Both of these hurdles have recently been overcome, the first with automatic differentiation (e.g. Griewank 1989) and the second with an HMC algorithm known as the no-U-turn sampler (NUTS; Hoffman & Gelman 2014). These advances have been packaged into the open-source, generic and flexible modelling software Stan (Gelman, Lee & Guo 2015; Stan Development Team 2016, Carpenter *et al.* in press), which effectively aims to replace the BUGS family and is quickly gaining traction across diverse fields (Fig. 1).

Despite the potential of HMC, and the availability of Stan, adoption has been slow in ecology, likely because ecologists are either unaware of its existence, or are unsure when it should be preferred over BUGS. Here, we illustrate the principles that underlie HMC and then compare the efficiency between Stan and a BUGS variant, JAGS (Plummer 2003), across a range of models in population ecology. Specifically, we test how HMC performance scales with model size and complexity, and its suitability for hierarchical models. Our goal is to explore the relative benefits of Stan and JAGS and to provide guidance for ecologists looking to use the power of HMC for faster and more robust Bayesian inference.

Principles of Hamiltonian Monte Carlo

The existing literature on HMC tends to focus on mathematical proofs of statistical validity and is accessible primarily to statisticians. We therefore first illustrate the principles of HMC using simple models, and contrast it with other MCMC algorithms.

Markov chain Monte Carlo algorithms sequentially generate posterior samples (i.e. vectors containing a value for each parameter), resulting in a finite number of autocorrelated samples which are used for inference (Gelman *et al.* 2014). Many algorithms *transition* between samples by proposing a new sample, based on the current sample and

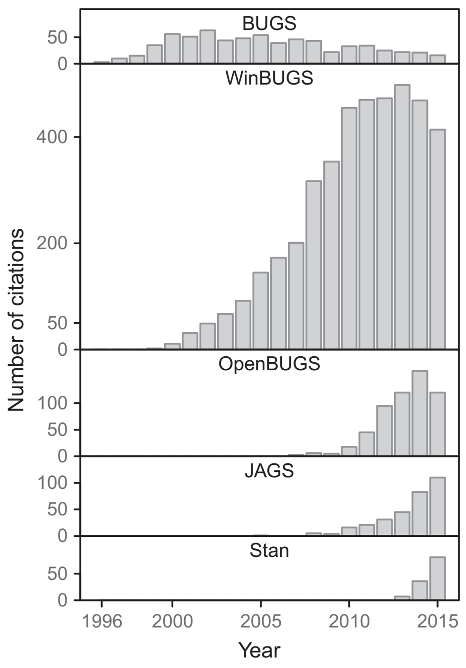

Fig. 1. Citation patterns of Stan and the BUGS family of Bayesian software platforms, for all journals in all fields. Data are from ISI Web of Science Core Collection. The *y*-axis units are the same, despite variable ranges.

tuning parameters, and then accept it with known probability. If rejected, the current iteration is the same as the previous one.

For example, the widely used random walk Metropolis algorithm (Metropolis *et al.* 1953) typically proposes a multivariate normal sample, centered at the current sample and uses the proposed to current posterior density ratio to determine the acceptance probability. In this case, all parameters are proposed and updated simultaneously, and the covariance of the proposal distribution is tuned to achieve an optimal acceptance rate (Roberts & Rosenthal 2001). Other algorithms update a single parameter at a time, looping through each within a transition. This is the behaviour typically used by BUGS, which uses Gibbs sampling if possible, and alternatives if not.

If an algorithm cannot propose samples in regions of the posterior distant to the current state, then it exhibits random walk behaviour: multiple transitions are necessary to move between regions, leading to higher autocorrelation and slow mixing. HMC avoids this inefficient random walk behaviour because it can propose values (almost) anywhere in the posterior from anywhere else. It does this using a physical system known as Hamiltonian dynamics.

HAMILTONIAN DYNAMICS

A Hamiltonian system can be conceptualized as a ball moving about a frictionless surface over time (e.g. imagine a marble inside a large bowl). The ball is affected by gravity and its own momentum: gravity pulls it down while momentum keeps it going in the same direction. A set of differential equations govern the movement of the ball over time (its *path*).

There are some important concepts associated with the ball. The *position* of the ball is its coordinate vector (i.e. where it is on the surface) and associated with each position variable is a *momentum* variable. The *potential energy* is the height of the surface at a given position. The *kinetic energy* is related to the momentum, assumed for now to be the sum of the squared momenta. Because the surface is frictionless, the total energy (potential plus kinetic), known as the *Hamiltonian* (*H*), remains constant over time. Later, we will see that, in the context of MCMC, the position vector corresponds to the model parameters and the potential energy to the negative log of the posterior density.

For now, consider the parabola $y = x^2$ (Fig. 2a), which has a single position variable (x) and thus a single momentum variable. We place the ball at position $x = -1$ and height (potential energy) $y = 1$, and let it go such that it has no initial momentum or kinetic energy. Gravity pulls it down, building speed over time as potential energy is converted to kinetic energy (Fig. 2b,c). Momentum carries it past position $x = 0$, where all potential energy has been converted into kinetic energy. As there is no friction, it stops exactly at $x = 1$ and $y = 1$, where the potential and kinetic energies return to their initial states (Fig. 2c). At this point, it will reverse course (Fig. 2a–c red lines) and oscillate forever with the energies varying but their sum (*H*) remaining constant.

Now consider a 2D parabola, $y = x_1^2 + x_2^2$ (i.e. a bowl shape). The position and momentum vectors are of length two, but the kinetic and potential energies are scalars. We place the ball as before, but this time we flick it, imparting momentum with a direction and magnitude (Fig. 2d). If flicked sideways, it will move in a circle of constant height. If flicked straight down, it will cross the bottom and go up the other side. An elliptical path occurs when flicking the ball at a downward angle. A more complex surface typical of a real model, such as a logistic growth model (see ‘Case studies’ below), leads to more complex paths (Fig. 2e,f), but which obey the same principles and intuition as these simple examples.

The principles of Hamiltonian dynamics relate directly to MCMC by providing a way to generate efficient transitions. The ball could move (almost) anywhere given the right length of time and initial momentum, thus providing transitions with directed movement and avoiding inefficient random walk behaviour. MCMC algorithms that utilize Hamiltonian dynamics are generally referred to as HMC, and we briefly review two: static HMC and NUTS.

STATIC HMC

Static HMC was the first MCMC algorithm to utilize Hamiltonian dynamics (Duane *et al.* 1987). Although replaced by more advanced algorithms, static HMC is simpler to explain and contains most of the properties relevant for understanding NUTS. A static HMC transition occurs by simulating the ball from the current position with random momenta for a finite length of time and proposing the state (position) at the end of this simulated, finite path.

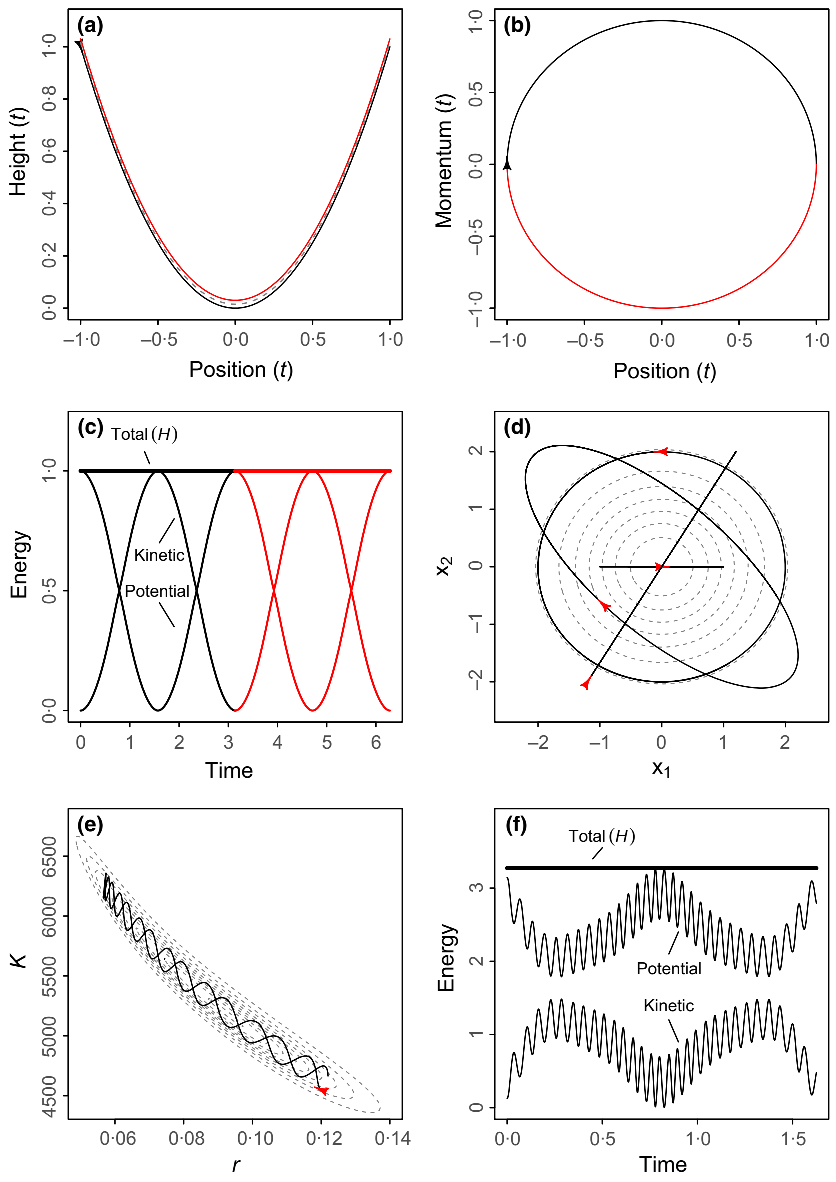

Fig. 2. Basics of Hamiltonian dynamics. (a) An example where a ball is dropped from the black point, it rolls down the surface over time (*t*), and momentum carries it up the other side where it reverses direction (red line), returning to where it started. The lines are offset to distinguish black and red paths. The position and momentum variables (b) and energies (c) over time corresponding to the path in (a). (d) Multiple paths for a 2d parabola. Grey dashed lines show posterior contours; initial positions and paths are red arrows and black lines. (e) Partial path (black line) on a posterior of a logistic population model with intrinsic growth rate (*r*) and carrying capacity (*K*). Red arrow shows initial position. (f) The energies for the trajectory in (e).

However, three issues complicate this process. The first is how to simulate movement on arbitrary log-posteriors (i.e. generate paths). Simple models like a parabola have analytical solutions to the underlying differential equations; thus, exact, continuous paths are possible. However, for most models, the continuous paths must be approximated using a numerical method known as the leapfrog integrator (we refer to approximated paths as *trajectories*). A trajectory depends on the *step size* (ε) and the *number of steps* (*L*; Fig. 3a,b). The position vector at step *L* is the proposed sample for that transition, while the intermediate steps are discarded (Fig. 3c). Approximation errors cause the ball to deviate from the continuous path, and thus, *H* is not constant over time (Fig. 3d).

The next challenge is determining the optimal *trajectory length* (i.e. ε*L*). If the trajectory length is too short, distant proposals are impossible, leading to an inefficient random walk. If it is too long, the trajectory will retrace its steps (e.g. Fig. 3a), which is wasteful computationally. Thus, efficiency depends on the trajectory length, but the optimal length is difficult to determine and a crucial tuning step required for static HMC (Betancourt 2016b).

The last issue is determining the step size, given a trajectory length. The same length can be attained by taking fewer steps of larger size, or more steps of smaller size (Fig. 3a,b). As each step is computationally costly, the fewer the steps the faster the transition. However, there is a downside to large step sizes: they lead to more variation in *H*, and in some cases, the approximation error accumulates such that the total energy (*H*) goes to infinity, known as a *divergent transition* (red trajectory, Fig. 3b). A Metropolis acceptance step accounts for variation in *H* by accepting the proposed state with probability min(1, α), where α is the exponential of the energy lost. Thus, proposals are always accepted if the total energy has decreased, whereas increased energy is accepted with a probability <1 (Fig. 3d). Increasing the step size reduces run-time, but increases approximation error, leading to more rejected states and divergent transitions, degrading the efficiency of the

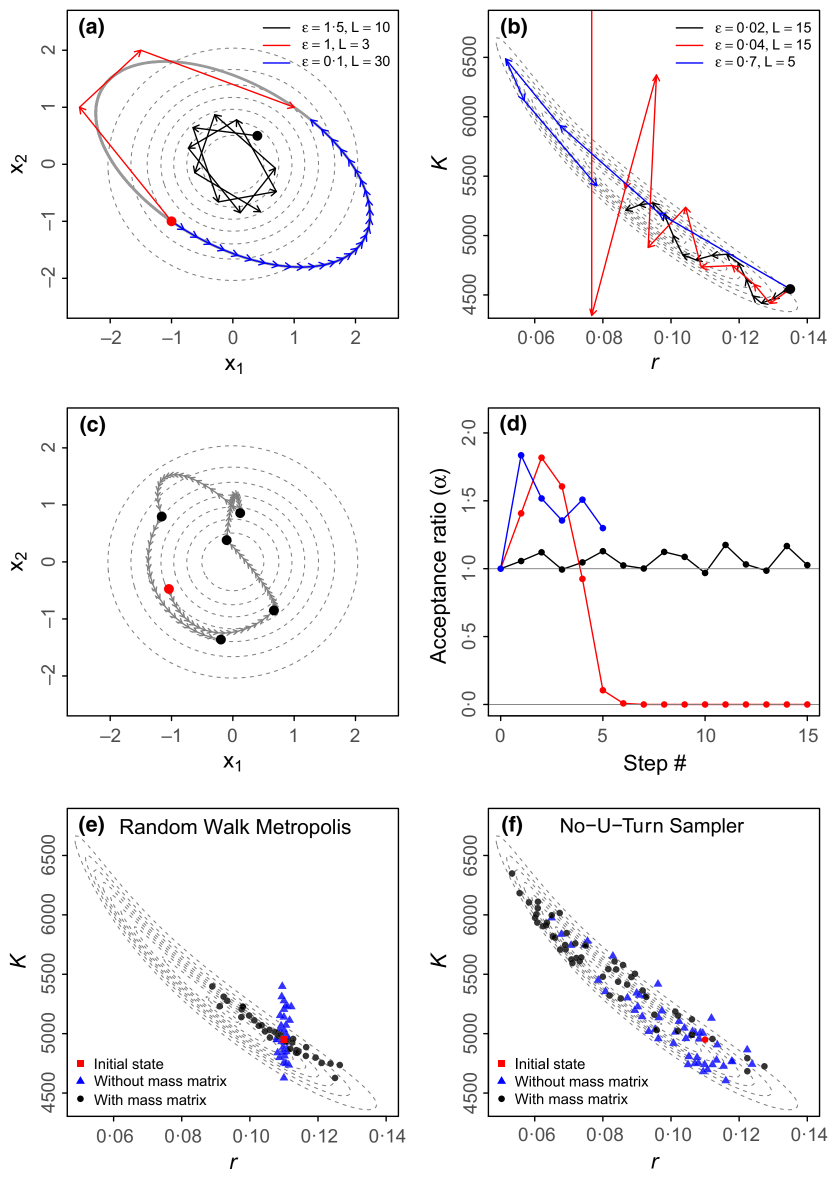

Fig. 3. Examples demonstrating the basics of HMC. (a) The effect of different step sizes (ε) and number of steps (*L*) on trajectories. The blue and red trajectories approximate the same path (solid grey line), with the same initial position (red point) and trajectory length (ε*L*), but opposite momentum. (b) Trajectories on a logistic posterior surface with identical initial position (black point) and momentum vectors. The black trajectory is slow to traverse the surface, while the red trajectory shows accumulating approximation errors, causing it to diverge. The blue trajectory utilizes a mass matrix, making the surface easier to traverse. (c) Multiple iterations of static HMC; black points are and accepted and intermediate steps (grey arrows) are discarded. (d) The acceptance ratios (α) of the trajectories in (b), with corresponding acceptance probability of min(1, α). Multiple draws from the same initial position using a random walk Metropolis (e) or NUTS (f) algorithm, with and without an appropriate mass matrix (colours).

algorithm. Optimizing the step size is thus another crucial step in static HMC (Betancourt, Byrne & Girolami 2014a).

Given a step size and number of steps, the last step is to specify a kinetic energy function. In HMC, it is typically the log density of a multivariate normal random vector where the covariance matrix is known as the *mass matrix*. Previously, we assumed the kinetic energy was the sum of the squared momenta, corresponding to an identity mass matrix. The effect of the mass matrix is to globally transform the posterior to have a simpler geometry for sampling. The variances stretch the posterior so all parameters have the same scale, while the covariances rotate it so they are approximately independent. When successful, the transformed parameters have a scale of 1 and no correlations, resembling iid standard normal random variables (blue trajectory, Fig. 3b.)

The mass matrix is analogous to the covariance of the proposal function sometimes used in Metropolis-Hastings samplers, which can have substantial impacts on sampling (Fig. 3e). Depending on the model, HMC algorithms can be efficient with an identity mass matrix (Fig. 3f), but it will require more leapfrog steps per transition and more time (Fig. 3b). Thus, to get efficient sampling with HMC, the mass matrix should approximate the covariance of the posterior, but this information is often not known *a priori*.

Specifying an optimal trajectory length, step size and mass matrix is critical for static HMC to work efficiently, leading it to require expert hands-on tuning and *a priori* knowledge (Neal 2011). Fortunately, NUTS automates this process and provides efficient sampling with minimal or no tuning.

THE NO-U-TURN SAMPLER

No-U-turn sampler extends static HMC by automating tuning: neither the step size nor number of steps need be specified by the user. NUTS determines the number of steps via a sophisticated tree building algorithm, which we briefly describe here. A single NUTS trajectory is built by iteratively accumulating steps. In the first iteration, a single leapfrog step is taken from the current state so the trajectory has a total of two steps. Then, two more steps are added (total of four), then four more (total of eight), and so forth, with each iteration doubling the length of the trajectory. This doubling procedure repeats until the trajectory turns back on itself and a 'U-turn' occurs, or the trajectory diverges (i.e. H goes to infinity). The number of doublings is known as the *tree depth*. The key aspect of this tree building algorithm is that it automatically creates trajectories that are neither too short nor too long. In practice, this means trajectory lengths vary among transitions: it may take eight steps or 128, depending on the position and momentum vectors.

The no-U-turn sampler determines the step size by adapting it during the warm-up (burn-in) phase to a target acceptance rate (`adapt_delta` in Stan). The tuned step size is then used for all sampling iterations. In contrast to static HMC, NUTS does not use a Metropolis acceptance step, so an analogous statistic is used for adaptation. Betancourt, Byrne & Girolami (2014a) found this target acceptance rate should generally be between 0·6 and 0·9, with larger values being more robust in practice. Thus, NUTS effectively reduces static HMC to a single, user-specified tuning parameter: the target acceptance rate.

HMC IN PRACTICE

One disadvantage of HMC is that, unlike BUGS, only continuous parameters are possible because discrete parameters do not have gradients. A manual implementation could overcome this by alternating Gibbs updates and HMC (Neal 2011), and future versions of Stan may implement such a scheme. Alternatively, in some cases, they can be marginalized out manually by the user (Chapter 10 and 12, Stan Development Team 2016).

Another disadvantage is that HMC is developed using sophisticated mathematics and statistics (e.g. Betancourt *et al.* 2014b), making it difficult to develop a deep understanding or intuition about their behaviour. We provide implementations of the static HMC and NUTS algorithms, written in R (R Core Team 2016), in Appendix B. We encourage the interested reader to experiment with the samplers to further their understanding of HMC, while using the faster and more robust Stan implementation for inference of real problems.

No-U-turn sampler (and static HMC) is similar to other MCMC algorithms: valid inference is conditioned on a converged chain, but this is impossible to prove (Gelman *et al.* 2014). The analyst is responsible for assessing convergence before making inference, and for NUTS, this includes assessing adaptation. Information about step size, tree depths and mass matrix quantities are reported in the output of a Stan run, and they should be checked routinely. For example, the adapted step size should be consistent across multiple chains, post-warm-up divergences should be minimized (by increasing target acceptance rate) and the maximum tree depth increased if necessary. The user manual (Stan Development Team 2016) has more information, advice on fitting strategies and details of the adaptation procedure for the mass matrix and step size.

Key concepts that arise when using NUTS in Stan are summarized briefly below:

- Smaller step sizes have higher acceptance rates, but require more steps and thus time. Larger step sizes reject more states and can have more divergences. The optimal step size depends on the model and is tuned to achieve a *target acceptance rate* set by the user (`adapt_delta`), defaulting to 0·8, but higher values needed for more difficult posteriors.
- The number of steps is determined dynamically for each transition using a tree building algorithm, where the trajectory repeatedly doubles in length until a U-turn occurs. The number of doublings is known as the *tree depth*.
- If the mass matrix approximates the covariance of the posterior, the algorithm 'sees' a simpler surface and is more efficient. By default only the diagonal terms are estimated, accounting for differences in scales, but not correlations, between parameters. Mass matrices with nonzero covariance terms, referred to as *dense,* are available in Stan but are not commonly used.
- The optimal step size depends on the mass matrix, and the mass matrix cannot be well estimated without sampling from the entire posterior, which requires a reasonable step size. Thus, sufficiently long warm-ups are needed for effective adaptation and efficient sampling.

Case studies

We tested the efficiency of Stan and JAGS for simulated and empirical models from population ecology. To quantify efficiency, we used the minimum number of effective samples per unit time, $E = \widehat{N}_{\mathrm{ESS}}/t$, a standard approach to compare among algorithms and software platforms. Further details of how this was calculated can be found in Appendix C. This definition of efficiency (E) can be roughly thought of as the number of independent samples generated per unit time.

We used matching parameterizations for Stan and JAGS, but explored two parameterizations for each hierarchical model and platform. MCMC efficiency for hierarchical models depends on the random effect parameterization, with the *centered* and *non-centered* complementary forms being useful for a broad class of models (Papaspiliopoulos, Roberts & Skold 2007; Betancourt & Girolami 2015). Briefly, the centered form models the random effects (τ) directly: $\tau \sim N(\mu, \sigma^2)$, while the non-centered form does it indirectly by letting $\tau = \mu + \sigma Z$, where $Z \sim N(0, 1)$ are the model parameters and implying $\tau \sim N(\mu, \sigma^2)$. See Appendix D for further information and references. We test both forms because the most efficient can depend on the amount of information about σ.

Initial values, random seeds and length of adaptation can have large impacts, particularly for HMC, so we ran 20 chains of length 40 000 without thinning, initialized from a random sample from a previously run long chain. We used the first half of each chain as a warm-up, discarding those samples but including warm-up time (but not compilation time) in the total run-time. We also did not include time to tune the target acceptance rate for Stan, as the analyst will often determine acceptable tuning parameters during model development. We used default settings for JAGS and Stan, except increasing the target acceptance rate from its default of 0·8 where needed (see Appendix E). We checked convergence, as is typically carried out for MCMC output, such as the potential scale reduction, $\widehat{R}$, being close to 1 (Gelman *et al.* 2014), in addition to the specific diagnostics for NUTS.

Our tests included two simulated models and four models with real data (Table 1). The simulated models were a multivariate normal with random covariances (MVND) or repeated correlations (MVNC), both of which were easy to vary in the number of fixed effects and covariance structure. Our simulated nonlinear mixed effects somatic Growth model varied in the number of individuals. The first two real data models were fit to mark–recapture data of birds and differed in their size and complexity: the Redkite model only estimates survival while the Swallows model estimates survival and detection probabilities using environmental covariates in a complex hierarchical state-space formulation. We also fit a state-space Logistic population dynamics model to fisheries data to estimate temporal trends in abundance. Lastly, our Wildflower model was a generalized linear mixed effects model with crossed random effects estimating flowering success. The case studies ranged from 5 to 1101 parameters and were a mixture of hierarchical and non-hierarchical models. Further details can be found in Appendix E, and model files for both Stan and JAGS in Appendix B. We did our analyses using R and the packages RSTAN and RJAGS.

Results

For the multivariate normal models (MVND and MVNC), the run-time of JAGS increased at a faster rate than Stan with increasing number of parameters, although the minimum effective sample size for a given run was similar between the two software platforms. Stan was more efficient by several orders of magnitude because its run-time for each sample was faster, and increasingly better with more parameters (Fig. 4a,b). For the growth model, Stan consistently outperformed JAGS at higher dimensions for both parameterizations. However, Stan had more variable efficiencies than JAGS with fewer individuals.

Stan was more efficient for the real-world models as well (Table 2), up to 63 times for the Logistic model in the non-centered form. JAGS was faster for the centered Swallows and Wildflower models, but for both the non-centered Stan model was the fastest option overall. Thus, Stan was faster for all models (using the optimal parameterization), although the variability in Stan's efficiency tended to be higher than for JAGS (results not shown), likely reflecting HMC's sensitivity to tuning compared to other algorithms.

We also found clear differences between software platforms in the effect of the parameterization for hierarchical models. For Stan, the non-centered form was consistently faster than the centered form for models with real data: 4·3 times faster for the Logistic, 2·8 times for the Swallows and 129 times for the Wildflower model. In contrast, JAGS was slower for all three: 0·90, 1·00 and 0·67, respectively. For the simulated Growth model, the non-centered form was faster for Stan, but slower for JAGS across all dimensionalities (Fig. 4c).

Discussion

Hamiltonian Monte Carlo is a family of MCMC algorithms which utilizes the posterior geometry and properties of Hamiltonian dynamics to make directed MCMC transitions, minimizing the inefficient random walk behaviour that degrades the performance for many algorithms used by JAGS. HMC is available to ecologists in the form of Stan, a generic and flexible software package with a similar workflow to JAGS. Here, we demonstrated that Stan outperformed JAGS for all simulated

Table 1. Summary of case studies used to compare efficiency between Stan and JAGS. Further details are available in Appendix E. Latent parameters are those modelled as random effects

Model name	Description	Data	Parameters (Latent)	Hierarchical structure	Reference
MVND	Multivariate normal with covariances generated from inverse Wishart	Simulated	Varies: 2–200	None	Simulated
MVNC	Multivariate normal with all off-diagonals set to ρ	Simulated	Varies: 5–50	None	Simulated
Growth	Nonlinear somatic growth with repeated measures	Lengths at age	Varies: 16–406 (10–400)	Normal on growth rate and maximum length, in log space	Simulated; see Schnute (1981)
Redkite	Age-dependent survival probabilities	Mark–recapture of birds	5	None	Section 8·4 of Kéry & Schaub (2012)
Swallows	State-space survival and detection with environmental covariates	Mark–recapture of birds	177 (172)	Year and family effects for survival, family effects for detection	Section 14·5 of Korner-Nievergelt *et al.* (2015)
Logistic	State-space fisheries logistic population dynamics	Annual catch per unit effort; catches	28 (22)	Annual biomass dynamics deviations	Millar & Meyer (2000)
Wildflower	Binomial generalized linear model of flowering success	Stages, flower, and seed pod production	1101 (1072)	Year effects on intercept; crossed effects on intercept and slope for covariate	Bolker *et al.* (2013)

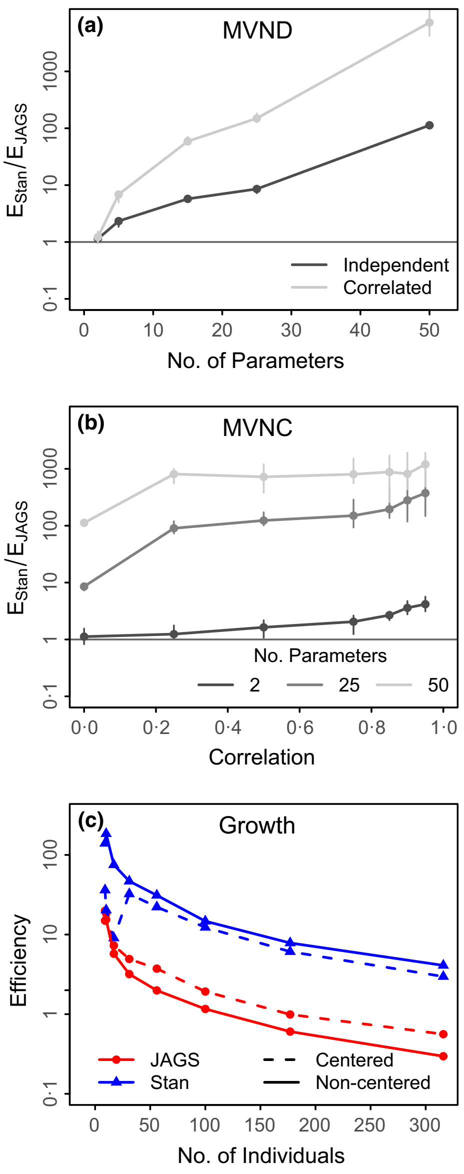

Fig. 4. Comparison of efficiency (*E*) for Stan and JAGS across simulated models. The means (points) and ranges (segments) are across 20 replicates. (a) A multivariate normal with increasing dimensionality (MVND), either independent or with random correlations from an inverse Wishart distribution. Ranges are too narrow to be visible. (b) A multivariate normal with repeated correlations on the off-diagonals for varying dimensions (MVNC). (c) A nonlinear mixed effects model with two latent parameters per individual (Growth); ranges were left out for visual clarity.

and real-world models from population ecology across a range of dimensions and complexity. Stan was more sensitive to the parameterization of the random effects, suggesting analysts use non-centered parameterizations to improve performance (Appendix D).

Our findings corroborate studies from other fields (e.g. Grant *et al.* 2016), but come with caveats when trying to extrapolate. For example, our simulated models might not reflect nuances in real data, or might not be representative of typical models in other subfields of ecology. Fair comparisons between software are also difficult, because many factors influence performance, including, but not limited to, priors, tuning parameters, length of chains and parameterization chosen. For instance, a model that is faster in Stan with a specific prior or parameterization may be faster in JAGS with alternatives. Nevertheless, the results from our case studies suggest that Stan will often be more efficient and thus provide faster inference.

Although our focus was on quantifying sampling efficiencies, the software platforms also behave differently for pathological models. Pathologies are properties of the posterior which obstruct an algorithm's ability to explore the entire posterior, resulting in biased inference of quantities of interest (Betancourt 2016a). For instance, posteriors with regions of very low or high curvature (gradients) can be pathological for HMC (section 6.6, Livingstone *et al.* 2016). Pathologies affect both Stan and JAGS, but Stan naturally diagnoses them: regions of high curvature are identified by divergences, and flat regions by excessive tree depths (Betancourt 2016a). JAGS provides no such feedback, and pathologies may not be apparent using traditional MCMC diagnostics. Pathologies using Stan occur in practice: centered hierarchical models can exhibit biased hypervariances due to high curvature (Fig. 5a). A Stan user can try to eliminate potential bias by reducing the step size, reparameterizing (e.g. non-centering, Fig. 5b–d), changing priors or restructuring their model. Thus, Stan is not only more efficient than JAGS, but it may also provide more robust inference because a user is more likely to detect and eliminate potential biases.

Despite its promise, HMC has some clear disadvantages, with the most critical that discrete parameters are disallowed, such as a discrete latent states or population numbers (e.g. Dail & Madsen 2011). HMC can still be used if the parameters can be marginalized out analytically, as in the binary states of the Swallows model, and this technique is often possible and can also make substantial improvements for JAGS as well (results

Table 2. Case study results comparing efficiency of Stan and JAGS. Max correlation is the largest absolute pairwise correlation, calculated from converged samples. Efficiency (*E*) is the number of effective samples per time

Model	Random effects parameterization	Max correlation	Median E_{stan}	Median E_{jags}	Median E_{stan}/E_{jags} (Range)
Redkite	NA	0·83	1102·85	302·99	3·54 (1·14–10·03)
Logistic	Centered	0·96	12·35	0·98	12·2 (7·88–34·54)
Logistic	Non-centered	0·96	53·60	0·88	63·33 (18·25–132·02)
Swallows	Centered	0·90	0·12	0·10	0·94 (0–2·96)
Swallows	Non-centered	0·81	0·34	0·10	2·4 (0·1–10·04)
Wildflower	Centered	0·96	0·01	0·06	0·14 (0·02–1·03)
Wildflower	Non-centered	0·96	1·29	0·04	34·2 (13·11–60·7)

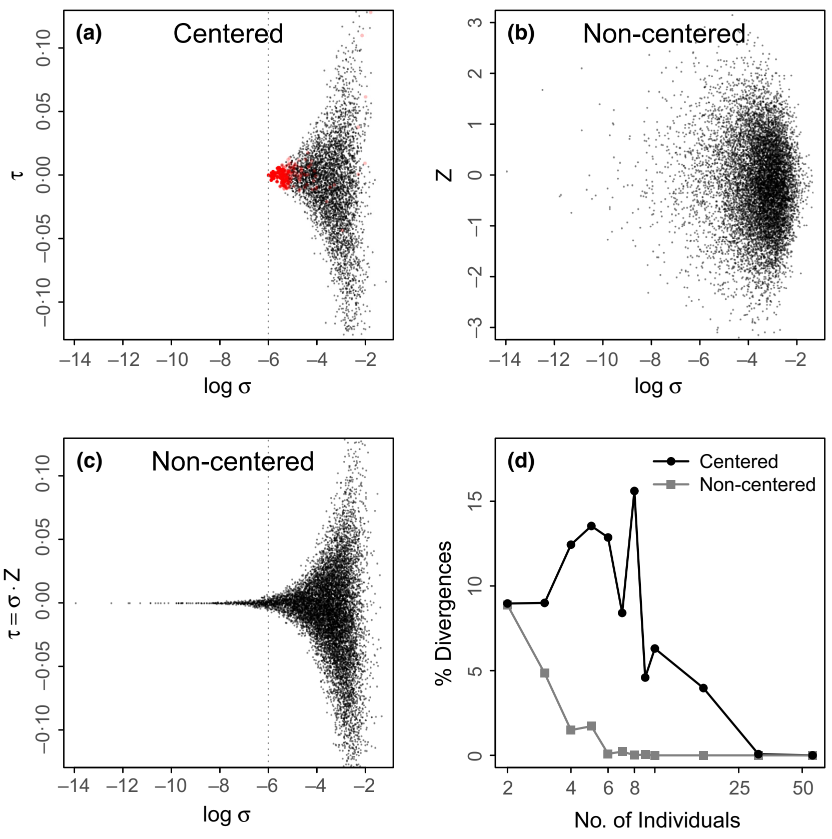

Fig. 5. Effects of non-centering on divergences and bias for the random effects on growth rate in the Growth model with 10 individuals. τ is the deviation from the mean for an arbitrary individual and the parameters in the centered model, σ its standard deviation and $Z \sim N(0, 1)$ the parameters in the non-centered model. Samples from: (a) the centered model (target acceptance rate δ = 0·95); (b) the non-centered model (δ = 0·80); and (c) the transformed non-centered parameters, $\tau = \sigma Z$. Divergences in (a), shown in red, arise because the adapted step size is too large for the high gradients at low σ, creating an inaccessible region and leading to biased σ (i.e. no samples below log σ = −6). The non-centered parameterization eliminates the curvature and hence the divergences and bias (c). (d) Median rate of divergent transitions using δ = 0·80 for both parameterizations. As information increases about σ (i.e. more individuals) the marginal distribution of σ narrows, simplifying the geometry and lowering the rate of divergences.

Table 3. Summary of key differences between JAGS and Stan

	JAGS	Stan
Inference	Bayesian only (MCMC)	Bayesian (MCMC with NUTS and variational inference) and penalized maximum likelihood
Tuning	Automatic with no options	Automatic with options for target acceptance rate (`adapt_delta`), mass matrix (diagonal or dense)
Discrete parameters	Use directly	Incompatible– must be marginalized out analytically
General pros	Easy to use, no tuning, discrete parameters	Scales well with dimensionality, posterior complexity; suitable for hierarchical models, especially the non-centered form
General cons	Few alternatives to reduce run-time when prohibitively slow	No discrete parameters, more difficult modelling language and additional MCMC diagnostics to check
Potential pathologies	No feedback	Divergences and excessive tree depths warn of steep or flat curvature, respectively

not shown). HMC is also sensitive to tuning, despite the automation provided by NUTS. For instance, if warm-up periods are too short to effectively explore the entire posterior, then the step size and mass matrix will be suboptimal and efficiency may suffer. Users must also be more involved in assessing tuning for Stan, and be familiar with the principles of HMC to understand diagnostic output from Stan.

There are other HMC algorithms in addition to NUTS, and other gradient-based algorithms for Bayesian inference, which were not tested here. For instance, Riemann Manifold HMC varies the mass matrix along the trajectory (Girolami & Calderhead 2011; Betancourt 2013) and variational inference is a faster alternative to MCMC which approximates the posterior (Kucukelbir *et al.* 2016). There are also alternative software platforms not tested here, such as NIMBLE (de Valpine *et al.* 2016) and ensemble sampling (Goodman & Weare 2010), and future work comparing these to JAGS and Stan would be worthwhile. Stan is also not the only platform coupling automatic differentiation and HMC that is used by ecologists. Both AD Model Builder (Fournier *et al.* 2012) and Template Model Builder (Kristensen *et al.* 2015) have HMC capabilities, but neither are as well developed or mature as Stan (author CCM is a developer of them). Our results suggest improving HMC capabilities in these software programs would be worthwhile for their user bases.

The preferred software depends on the situation (Table 3), and JAGS will clearly remain a valuable tool when run-time is not prohibitive, but also likely in additional cases such as prototyping models or introducing Bayesian techniques. Stan is clearly the best option for highly parameterized models or smaller models with more difficult geometries (e.g. high or anisotropic correlations). One promising application for HMC is fisheries stock assessment models, which are often extremely large, nonlinear hierarchical models that rarely use Bayesian inference because of prohibitively slow run-times (e.g. Stewart *et al.* 2013). Many other fields likely have similar examples where Bayesian inference is currently infeasible, and we anticipate that HMC will make some of these problems tractable for the first time.

Increasingly large and complex data sets, and powerful software tools, allow analysts to investigate ecological processes which were previously infeasible. Here we demonstrated that Stan, which implements HMC in a flexible modelling platform, is a promising tool when status quo methods such as JAGS are prohibitively slow. We believe Stan should be in the methodological toolbox for every quantitative ecologist because it will extend the boundaries of feasible models for applied problems and lead to better understanding of ecological processes.

Acknowledgements

We thank Bob Carpenter and Michael Betancourt for insights on a variety of conceptual issues and constructive feedback on an earlier draft. Margaret Siple, Eric Buhle, Kevin See, Jim Hastie and two anonymous reviewers provided valuable feedback on an earlier version of this manuscript. This publication is partially funded by the Joint Institute for the Study of the Atmosphere and Ocean (JISAO) under NOAA Cooperative Agreement NA10OAR4320148 (2010–2015) and NA15OAR4320063 (2015–2020), Contribution No. 2016-01-23. This work was partially funded in part by a grant from Washington Sea Grant, University of Washington, pursuant to National Oceanic and Atmospheric Administration Award No. NA14OAR4170078. TAB was also funded by the Richard C. and Lois M. Worthington Endowed Professorship in Fisheries Management. The views expressed herein are those of the authors and do not necessarily reflect the views of NOAA or any of its subagencies.

Data accessibility

We used a combination of real data taken from previously published studies and simulated data using R scripts. All data, R scripts and results files are publicly available at https://github.com/colemonnahan/gradmcmc/tree/v1.0 and are also archived at Monnahan (2016).

References

Betancourt, M. (2013) Generalizing the no-U-turn sampler to Riemannian manifolds. arXiv preprint arXiv:1304.1920.

Betancourt, M. (2016a) Diagnosing suboptimal cotangent disintegrations in Hamiltonian Monte Carlo. arXiv preprint arXiv:1604.00695.

Betancourt, M. (2016b) Identifying the optimal integration time in Hamiltonian Monte Carlo. arXiv preprint arXiv:1601.00225.

Betancourt, M., Byrne, S. & Girolami, M. (2014a) Optimizing the integrator step size for Hamiltonian Monte Carlo. arXiv preprint arXiv:1411.6669.

Betancourt, M. & Girolami, M. (2015) Hamiltonian Monte Carlo for hierarchical models. *Current Trends in Bayesian Methodology with Applications*, (eds S. K. Upadhyay, U. Singh, D.K. Dey & A. Loganathan), pp. 79–101. CRC Press, Boca Raton, FL, USA.

Betancourt, M., Byrne, S., Livingstone, S. & Girolami, M. (2014b) The geometric foundations of Hamiltonian Monte Carlo. arXiv preprint arXiv:1410.5110.

Bolker, B.M., Gardner, B., Maunder, M. *et al.* (2013) Strategies for fitting nonlinear ecological models in R, AD Model Builder, and BUGS. *Methods in Ecology and Evolution*, **4**, 501–512.

Brooks, S., Gelman, A., Jones, G. & Meng, X.-L. (2011) *Handbook of Markov Chain Monte Carlo*. CRC Press, Boca Raton, FL, USA.

Carpenter, B., Gelman, A., Hoffman, M. *et al.* (in press) Stan: a probabilistic programming language. *Journal of Statistical Software*.

Cressie, N., Calder, C.A., Clark, J.S., Ver Hoef, J.M. & Wikle, C.K. (2009) Accounting for uncertainty in ecological analysis: the strengths and limitations of hierarchical statistical modeling. *Ecological Applications*, **19**, 553–570.

Dail, D. & Madsen, L. (2011) Models for estimating abundance from repeated counts of an open metapopulation. *Biometrics*, **67**, 577–587.

Duane, S., Kennedy, A.D., Pendleton, B.J. & Roweth, D. (1987) Hybrid Monte Carlo. *Physics Letters B*, **195**, 216–222.

Ellison, A.M. (2004) Bayesian inference in ecology. *Ecology Letters*, **7**, 509–520.

Fournier, D.A., Skaug, H.J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M.N., Nielsen, A. & Sibert, J. (2012) AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods and Software*, **27**, 233–249.

Gelman, A., Lee, D. & Guo, J.Q. (2015) Stan: a probabilistic programming language for Bayesian inference and optimization. *Journal of Educational and Behavioral Statistics*, **40**, 530–543.

Gelman, A., Carlin, J.B., Stern, H.S. & Rubin, D.B. (2014) *Bayesian Data Analysis*. Taylor & Francis, Boca Raton, FL, USA.

Girolami, M. & Calderhead, B. (2011) Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **73**, 123–214.

Goodman, J. & Weare, J. (2010) Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science*, **5**, 65–80.

Grant, R.L., Furr, D.C., Carpenter, B. & Gelman, A. (2016) Fitting Bayesian item response models in Stata and Stan. arXiv preprint arXiv:1601.03443.

Griewank, A. (1989) On automatic differentiation. *Mathematical Programming: Recent Developments and Applications*, **6**, 83–107.

Hoffman, M.D. & Gelman, A. (2014) The no-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, **15**, 1593–1623.

Hooten, M.B. & Hobbs, N.T. (2015) A guide to Bayesian model selection for ecologists. *Ecological Monographs*, **85**, 3–28.

Kéry, M. & Schaub, M. (2012) *Bayesian Population Analysis Using WinBUGS: A Hierarchical Perspective*. Academic Press, Amsterdam, Netherlands.

Korner-Nievergelt, F., Roth, T., von Felten, S., Guélat, J., Almasi, B. & Korner-Nievergelt, P. (2015) *Bayesian Data Analysis in Ecology Using Linear Models with R, BUGS, and Stan: Including Comparisons to Frequentist Statistics*. Academic Press, Amsterdam, Netherlands.

Kristensen, K., Nielsen, A., Berg, C.W., Skaug, H. & Bell, B. (2015) TMB: automatic differentiation and Laplace approximation. arXiv preprint arXiv:1509.00660.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A. & Blei, D.M. (2016) Automatic differentiation variational inference. arXiv preprint arXiv:1603.00788.

Livingstone, S., Betancourt, M., Byrne, S. & Girolami, M. (2016) On the geometric ergodicity of Hamiltonian Monte Carlo. arXiv preprint arXiv:1601.08057.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. (1953) Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21**, 1087–1092.

Millar, R.B. & Meyer, R. (2000) Non-linear state space modelling of fisheries biomass dynamics by using Metropolis-Hastings within-Gibbs sampling. *Journal of the Royal Statistical Society Series C: Applied Statistics*, **49**, 327–342.

Monnahan, C.C. (2016) Data and code for "Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo". doi: 10.5281/zenodo.159596

Neal, R.M. (2011) MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, Vol. **2** (eds S. Brooks, A. Gelman, G. L. Jones & X.-L. Meng), pp. 113–162. CRC Press, Boca Raton, FL, USA.

Papaspiliopoulos, O., Roberts, G.O. & Skold, M. (2007) A general framework for the parametrization of hierarchical models. *Statistical Science*, **22**, 59–73.

Plummer, M. (2003) JAGS: a program for analysis of Bayesian graphical models using Gibbs sampling. Page 125 *in* Proceedings of the 3rd international workshop on distributed statistical computing. Technische Universit at Wien, Wien, Austria.

R Core Team (2016) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roberts, G.O. & Rosenthal, J.S. (2001) Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, **16**, 351–367.

Royle, J.A. & Dorazio, R.M. (2008) *Hierarchical Modeling and Inference in Ecology: The Analysis of Data from Populations, Metapopulations and Communities*, 1st edn. Academic Press, London, UK.

Schnute, J. (1981) A versatile growth model with statistically stable parameters. *Canadian Journal of Fisheries and Aquatic Sciences*, **38**, 1128–1140.

Stan Development Team (2016) *Stan Modeling Language Users Guide and Reference Manual*, version 2.12.0. Available at https://github.com/stan-dev/stan/releases/download/v2.12.0/stan-reference-2.12.0.pdf.

Stewart, I.J., Hicks, A.C., Taylor, I.G., Thorson, J.T., Wetzel, C. & Kupschus, S. (2013) A comparison of stock assessment uncertainty estimates using maximum likelihood and Bayesian methods implemented with the same model framework. *Fisheries Research*, **142**, 37–46.

Thorson, J.T. & Minto, C. (2014) Mixed effects: a unifying framework for statistical modelling in fisheries biology. *ICES Journal of Marine Science*, **72**, 1245–1256.

de Valpine, P., Turek, D., Paciorek, C.J., Anderson-Bergman, C., Lang, D.T. & Bodik, R. (2016) Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, 1–28. doi:10.1080/10618600.2016.1172487.

Received 31 August 2016; accepted 6 October 2016
Handling Editor: Robert B. O'Hara

Supporting Information

Additional Supporting Information may be found online in the supporting information tab for this article:

Appendix S1 (Appendices A to E). Further details of the citation analysis, how MCMC efficiency was calculated, and the effect of noncentering.