Methods in Ecology and Evolution

British Ecological Society

Methods in Ecology and Evolution 2013, **4,** 501–512 doi: 10.1111/2041-210X.12044

Strategies for fitting nonlinear ecological models in R, AD Model Builder, and BUGS

Benjamin M. Bolker[1]*, Beth Gardner[2] †, Mark Maunder[3], Casper W. Berg[4], Mollie Brooks[5], Liza Comita[6] ‡, Elizabeth Crone[7], Sarah Cubaynes[8] §, Trevor Davies[9], Perry de Valpine[10], Jessica Ford[11], Olivier Gimenez[8], Marc Kéry[12], Eun Jung Kim[13], Cleridy Lennert-Cody[3], Arni Magnusson[14], Steve Martell[15], John Nash[16], Anders Nielsen[4], Jim Regetz[6], Hans Skaug[17] and Elise Zipkin[2]

[1]*Departments of Mathematics and Statistics and Biology, McMaster University, 1280 King St W, Hamilton, ON, L8S 4K1, Canada;* [2]*USGS Patuxent Wildlife Research Center, Laurel, MD,USA;* [3]*Inter-American Tropical Tuna Commission, La Jolla, CA,USA;* [4]*National Institute of Aquatic Resources, Technical University of Denmark, Charlottenlund, Denmark;* [5]*Department of Biology, University of Florida, Gainesville, FL,USA;* [6]*National Center for Ecological Analysis and Synthesis, Santa Barbara, CA,USA;* [7]*Harvard University, Harvard Forest, Petersham, MA,USA;* [8]*CNRS, Centre d'Ecologie Fonctionnelle et Evolutive, Montpellier,France;* [9]*Department of Biology, Dalhousie University, Halifax, NS,Canada;* [10]*Environmental Science, Policy and Management, University of California – Berkeley, Berkeley, CA,USA;* [11]*CSIRO-University of Tasmania, Institute for Marine and Antarctic Studies, Sandy Bay, Tas., Australia;* [12]*Swiss Ornithological Institute, Sempach,Switzerland;* [13]*School of Ocean and Earth Science and Technology, University of Hawai'i at Manoa, Honolulu, HI,USA;* [14]*Marine Research Institute, Reykjavik, Iceland;* [15]*UBC Fisheries Centre, University of British Columbia, Vancouver, BC,Canada;* [16]*Telfer School of Management, University of Ottawa, Ottawa, ON,Canada;* [17]*Department of Mathematics, University of Bergen, Bergen, Norway*

Summary

1. Ecologists often use nonlinear fitting techniques to estimate the parameters of complex ecological models, with attendant frustration. This paper compares three open-source model fitting tools and discusses general strategies for defining and fitting models.

2. R is convenient and (relatively) easy to learn, AD Model Builder is fast and robust but comes with a steep learning curve, while BUGS provides the greatest flexibility at the price of speed.

3. Our model-fitting suggestions range from general cultural advice (where possible, use the tools and models that are most common in your subfield) to specific suggestions about how to change the mathematical description of models to make them more amenable to parameter estimation.

4. A companion web site (https://groups.nceas.ucsb.edu/non-linear-modeling/projects) presents detailed examples of application of the three tools to a variety of typical ecological estimation problems; each example links both to a detailed project report and to full source code and data.

Key-words: JAGS, optimization, parameter estimation, R, AD Model Builder, WinBUGS

Introduction

The size and scope of ecological data sets, and the computational power available to analyse them, have exploded in recent years; ecologists' ambition to understand complex ecological systems has expanded in proportion. As a result, ecologists are fitting ever more complicated models to their data. While quantitatively sophisticated ecologists are gleaning rich insights from cutting-edge techniques, ecologists without formal training in statistics or numerical computation can become horribly frustrated when trying to estimate the parameters of complex models:

- Software for fitting such models may be platform-dependent or prohibitively expensive.
- Inflexible software forces users either to change their models or to modify the software.
- The documentation for fitting complex models is often sparse – software developers assume that users who are fitting complex models understand the associated, highly technical, statistical and computational issues.
- Model fitting may stop with errors, or produce obscure warnings, or get stuck at an obviously bad fit, depending on subtle changes in the way the model or the starting values are specified.

*Correspondence author. E-mail: bolker@mcmaster.ca

†Present address: Department of Forestry and Environmental Resources, North Carolina State University, Raleigh, NC, USA

‡Present address: Department of Evolution, Ecology and Organismal Biology, The Ohio State University, Columbus, OH, USA

§Present address: Department of Life Sciences, Imperial College London, Ascot, Berkshire, UK

- Software may get stuck at bad fits that are not obvious, or local optima, without reporting convergence problems; few diagnostics are provided to determine whether the model is appropriate.
- Debugging capabilities are often poorly developed.

These challenges are a far cry from the old-school procedure of designing a well-controlled field experiment with response variables that are normally distributed (or transformable: O'Hara & Kotze 2010; Warton & Hui 2011) and analysing them according to simple ANOVA frameworks (Underwood 1996; Quinn & Keough 2002; Gotelli & Ellison 2004). Even when logistical constraints required an experiment to be performed in experimental blocks, the results could still be analysed by figuring out the right category of experimental design (e.g. nested or randomized block) and looking up the appropriate sums of squares and degrees of freedom for F tests. 'New school' ecologists want to handle data, often observational and unbalanced, that are intrinsically non-normal, may be heteroscedastic, display nonlinear responses to continuous predictor variables and involve complex correlation structures that do not fit into the classical framework of nested and partly nested designs. Rather than being restricted to models that fit into classical statistical frameworks, ecologists should be able to apply the model that seems most appropriate for their questions. Even well-behaved experimental data that are traditionally analysed using ANOVA may be analysed with more appropriate models, such as time-structured population dynamics models (de Valpine 2003), to improve precision or accuracy or address more complex ecological questions. Of course, there is no free lunch: model complexity should always be constrained by the available data (Ludwig & Walters 1985; Adkison 2009).

In a nonlinear statistical model, the predicted values are nonlinear functions of the *parameters*, not necessarily of the predictor variables: thus, a quadratic model ($y = a + bx + cx^2$) is linear in the statistical sense (y is a linear function of the parameters a, b and c even though it is a nonlinear function of the predictor variable x), while a power-law model ($y = ax^b$) is not: in the linear regression model $y = a + bx$, y is a linear function of both the parameters a and b and the predictor variable x.

In the power-law example, the model could be linearized by taking logarithms – $\log y = \log a + b \log x$. Note, however, that the nonlinear model `y~a*x^b` (using `nls()` in R) is different from the linear model `log(y)~1 + log(x)` (using `lm()`), because the former assumes an error term with a constant standard deviation, while the latter assumes a constant coefficient of variation. However, most nonlinear models, such as the supplementary examples listed in Table 1, require the use of more general numerical optimization algorithms to estimate best-fit parameter values. The user must explicitly define an *objective function* that measures the fit of the model – typically this computes the residual sum of squares, log-likelihood or posterior probability – and pass it to the software as input. The software then uses numerical methods to find either a single value representing the best fit (in the case of maximum likelihood estimation) or a sample of values from near the best fit that represent a sample from the posterior distribution (in the case of Bayesian estimation). In order to define the objective function properly, users generally need to understand the properties of a variety of probability distributions and deterministic response functions (Clark 2007; McCarthy 2007; Royle & Dorazio 2008; King *et al.* 2009; Link & Barker 2010). Even once a model has been properly formulated, however, fitting it to the data to estimate the parameters is often challenging.

Table 1. List of model-fitting projects executed in R, ADMB, and BUGS (Detailed project reports and full source code and data for each project are available from https://groups.nceas.ucsb.edu/non-linear-modeling/projects.)

Name	Description
OrangeTree	Nonlinear growth model (normal/least-squares)
Theta	Theta-logistic population growth model (state-space)
Tadpole	Size-dependence in predation risk (binomial response)
Weeds	Simple population growth model
Min	Time series of mineralization (matrix-exponential solution of ODEs): normal/least-squares
Owls	Zero-inflated count data with random effects
Skate	Bayesian state-space model of winter skate mortality (ADMB, BUGS only)
Nmix	N-mixture model with random observer effects (ADMB, BUGS only)
Wildflower	Flowering probability as a function of size; binomial GLMM with multiple random effects

The bottom line is that if ecologists want to fit complex models to complex data, they will need powerful, flexible model-fitting tools. The good news is that these tools do exist: the bad news is that there is a dearth of worked examples and guidance for using them. In this paper, we report on the results of a National Center for Ecological Analysis and Synthesis (NCEAS) working group whose mission was to apply a set of different tools to a broad spectrum of nonlinear ecological modelling problems (Table 1), with the goals of (i) comparing the performance and applicability of the tools in different situations and (ii) producing a series of worked examples that could serve as guides for ecologists. The full results are available at https://groups.nceas.ucsb.edu/non-linear-modeling/projects; we encourage interested readers to browse and provide feedback.

In the interests of addressing the first problem above (expensive and/or platform-dependent tools), we restricted our scope to several general-purpose, powerful, but free and open-source software (FOSS) tools: R, AD Model Builder and BUGS, described below. Because they are free, they are available to researchers with restricted budgets – such as students and researchers in developing countries or at smaller, less research-intensive institutions. Because they are open-source, they offer transparency consistent with the philosophy of reproducible research (Peng 2009) and allow end-users to modify the code according to their particular needs. In practice, few working ecologists are likely to look at the underlying source code for these software tools, let alone modify it, but the availability of the code for modification does allow rapid diversification and

improvement by more computationally sophisticated ecologists. In the same spirit, all the source code for the worked examples is available on the companion website to this article.

We chose a variety of problems to exercise the capabilities of these tools, to illustrate a wide range of challenges and to create examples that would be useful to a broad set of users (Table 1). All the problems are nonlinear, ranging from simple models of normally distributed data to more complex models incorporating random effects, unusual distributions (mixture distributions or zero-inflation), spatial and temporal correlation and imperfect detection. One way in which our scope is restricted is that all of our data sets are moderate-sized: the largest data set was around 3600 observations of 10 variables (size on disk *c*. 160 kB). Thus, we are not exploring 'big data' in this exercise, and our methods emphasize parametric model fitting rather than exploration of patterns (Breiman 2001). We also do not investigate highly complex models that are used in some applications such as fisheries stock assessment (Maunder *et al*. 2009; Fournier *et al*. 2012).

In the rest of this paper, we (i) describe the scientific and cultural context for our work – what tools exist, in which fields they are used and how the development of statistical methods, software tools and particular scientific research projects can interact for mutual benefit; (ii) provide details of our methodology for implementing the examples; and (iii) attempt to synthesize useful general lessons from our experience that will help nonstatistical ecologists with their model-fitting challenges. The authors of this paper are all experts in at least one area of computational or statistical ecology: while we tried conscientiously to see things from the perspective of mainstream, nonstatistically expert ecologists, readers are cautioned to take terms like 'straightforward' and 'simple' with a grain of salt.

Scientific and cultural environment

The current scientific and cultural climate is ripe for rapid development and dissemination of new computational and statistical tools. Statistical and computational literacy of ecologists is increasing. On the other hand, there is lots of room for improvement – many new approaches in nonlinear estimation are still challenging even for motivated and statistically savvy ecologists. Tools useful to ecologists are often under rapid development, and as such they may be buggy or lack documentation, or have obscure interfaces.

We settled on three tools for constructing and fitting nonlinear ecological models: **R** is well known within the statistical and ecological communities and was released as free software in 1995. A variety of books specific to ecological modelling or data analysis are based on R (Bolker 2008; Reimann *et al*. 2008; Soetaert & Herman 2008; Stevens 2009; Zuur *et al*. 2009), while other more general R-based books are written by and accessible to ecologists (Crawley 2002, 2005, 2007). R is mature and offers a convenient working environment: of the tools we describe, R is the only one that offers a general platform for data management and analysis – in fact, all of the members of our group (even those who preferred other tools for model fitting) relied on R for managing and preparing data and for generating tabular and graphical output. A large variety of alternative graphical or script-editing interfaces are available for R (e.g. Emacs/ESS, Vim-R, Notepad++, Tinn-R, RStudio, RKward), as well as interfaces with many other tools such as relational database management systems, geographical information systems and other modelling tools such as the ones we describe below.

Advantages:

- Interactive environment with convenient high-level syntax for common tasks in statistical analysis and graphics.
- Very easy to install on all common platforms.
- As the most commonly used of these software tools, R has the largest quantity of help and documentation available in the form of books, mailing lists, courses and the likelihood of a nearby colleague who is well versed in R.
- A very large number of packages is available for R – more than 4000 packages in the central repository, including more than 100 specifically related to ecological modelling.

This profusion can also be viewed as a disadvantage. Despite the fact that all of these packages are easy to install from a central location, it can be difficult to find and evaluate the quality of third-party packages. Some resources that attempt to remedy this problem are the R Environmetrics Task View (http://cran.r-project.org/web/views/Environmetrics.html), the `sos` package, and the CRANtastic website (http://crantastic.org/search?q=ecology).

- It is relatively easy for users and beginning developers to create their own packages and, if appropriate, post them to a centralized archive site.

Disadvantages:

- Originally designed for interactive data analysis, R is generally slower than compiled programming languages such as Java, FORTRAN or C++ (or AD Model Builder, which is based on compiled C++ code), although carefully written code often compares favourably.
- Although lots of documentation is available, the documentation that comes with R is unquestionably terse and directed towards non-novice users. The standard advice given to hopeful R users is to find an R-oriented book (some are listed above) that covers their area of interest.

AD Model Builder (ADMB; Fournier *et al*. 2012; http://admb-project.org) is the most powerful but the least known and least polished of the software tools we use. First released in 1993 and an open-source project since 2007, ADMB has a vibrant user community within the fields of resource management. In fisheries science, more than 90 peer-reviewed papers have cited AD Model Builder. An integrated development environment (ADMB-IDE) is available (Magnusson 2009), facilitating the installation and use of the software. It is possibly the fastest and most robust FOSS tool for general-purpose nonlinear estimation. The user first writes a definition of the objective function (typically the negative log-likelihood function) in an extension of the C++ language containing utility functions for statistics and linear algebra. ADMB then compiles the model definition into an executable file that minimizes the objective function for a specified set of data. In addition to the speed advantage from compiling, ADMB implements

automatic differentiation (AD), an algorithm that rapidly and accurately calculates the derivatives of the objective function (Griewank & Corliss 1992), unlike the optimization routines in R, which typically rely on less-stable finite-difference approximations.

Advantages:

- ADMB was often the most robust and fastest of the tools we tested.
- Several alternative tools to evaluate the uncertainty of both estimated parameters and derived quantities: the delta method, profile likelihood and a *post hoc* Markov chain Monte Carlo (MCMC) implementation (the [`skate`] example shows an example of MCMC in ADMB).
- Estimation of random-effects models via a general Laplace approximation routine (Skaug & Fournier 2006) that allows the incorporation of continuous random effects into a general model ([`theta`], [`skate`], [`owls`] projects). Our other software tools are limited either to a specific subset of model types (R) or to a specified list of deterministic functions and stochastic distributions (BUGS).
- Support for constrained optimization (see Section 'Constrain parameters') and optimization in phases (or 'masks' Nash & Walker-Smith 1987), where some estimated parameters remain constant until the final stages of the optimization, when all parameters are estimated. Masks are also available in the R packages `bbmle`, `Rcgmin` and `Rvmmin`, although they cannot be switched on in the course of a single optimization run as in ADMB.
- ADMB's algorithm is sufficiently robust that one can fit simple models with the default (all zero) starting parameters – something that is rarely possible with the other tools we evaluated. This is partly due to ADMB's use of exact numerical derivatives calculated by automatic differentiation.
- Once a model is successfully built in ADMB, the compiled executable can be distributed as a stand-alone program and run with new data sets (on the same platform/OS) independently of any other tools, unlike R or BUGS code which require full installations. For researchers who already use **R**, the `R2admb` interface to R simplifies the task of preparing data for input to ADMB and analysing results from ADMB fits.

Disadvantages:

- Little documentation is available for ADMB: mainly the user's manual, an overview paper (Fournier *et al.* 2012), resources on the ADMB project website and an active mailing list. There is a single published book describing how to use ADMB (Millar 2011) and the user community is small.
- Although it is difficult to make a precise comparison between the ease of learning to use different tools, an informal rating exercise of the participants in our group (all experienced modellers) found that ADMB rated lowest on ease of use. Scoring on a range from 1 = very hard to 5 = very easy, most (11/16) participants gave ADMB a score of 2 (mean 2·1, range 1–3), while most (9/16) gave R a score of 4 (mean 3·6, range 2–5). BUGS was intermediate, with a modal value of 3 (6/15, mean 3·3, range 2–5).
- ADMB is still a relatively young project. The latest release (11.0, July 2012) included several important bug fixes, as well as new user functions that were not yet covered in the user manual at the time of release.

BUGS (Bayesian inference Using Gibbs Sampling) describes a family of tools that includes the original 'classic' BUGS, the widely used WinBUGS with a graphical front-end for Windows, its open-source version OpenBUGS and the independently developed JAGS, which uses a largely compatible model description language. The original BUGS and WinBUGS were developed in the mid-1990s, the current open-source version (OpenBUGS) first appeared in 2004, and JAGS was released in 2007 (Lunn 2009). Like ADMB, the user writes a model definition in a specialized language – in the case of BUGS, the language is a special-purpose language designed for describing hierarchical Bayesian models, with a syntax based on defining relationships using probability distributions. After specifying data and initial values for the parameters, the user then runs one or more Markov chains based on the model definition, evaluates the success of the chains in converging on a stable posterior distribution, either graphically or numerically, and draws conclusions from the posterior sample (Lunn *et al.* 2012). One obvious difference between BUGS and the other software tools is that BUGS uses an explicitly Bayesian framework. ADMB and R users most often work in the frequentist or likelihood frameworks, although both tools have the capability to use Bayesian inference as well. In our analyses, we rarely found big differences between the results of our Bayesian and frequentist analyses. The point estimates sometimes differed slightly due to the difference between the posterior mean reported by BUGS and the maximum likelihood estimate, which is approximately equal to the *mode* of the posterior distribution when the prior distribution is uninformative. (The estimated posterior densities in the [`theta`] project were clearly asymmetric and non-Gaussian, leading to a large difference between the posterior modes, medians and means.) The confidence intervals reported by BUGS were often slightly wider, because BUGS allows more naturally for nonquadratic log-likelihood (or log-posterior) surfaces and because its MCMC algorithm more easily accounts for diverse sources of variation than the default algorithms used by other tools: see the [`owls`] project for an example.

Advantages:

- BUGS makes the power of the hierarchical Bayesian approach available in a reasonably simple way for a wide range of possible models.
- BUGS defines relationships among observations and parameters using shorthand notation for probability distributions, which some users find more intuitive than writing out full likelihood equations and priors.
- By requiring the user to write out hierarchical models explicitly, users often gain a clearer understanding of their models than when using more black-box approaches such as the basic generalized linear models available in some R packages.

- BUGS handles discrete random variables, for example discrete mixture models, which are not possible in ADMB and which can only be done in R using special-purpose packages.
- Provides posterior distributions and confidence regions for all parameters in the model and for quantities computed from them, which can be challenging to do via other approaches.

Disadvantages:

- BUGS is generally the slowest by far of the approaches we tested, although the results of a BUGS run do provide more information on confidence intervals than the corresponding deterministic fit via R or ADMB. Part of this speed penalty is a characteristic of Bayesian analysis rather than of BUGS itself; for example, MCMC analyses with ADMB usually take considerably longer than ADMB's maximum likelihood estimation.
- BUGS is quirky, and debugging BUGS code is well known to be challenging, due to the opacity of the underlying computations, cryptic error messages and the inherent difficulty of building robust MCMC samplers for complex models.
- BUGS has the smallest range of available distributions and functions of the three software tools tested, although there are tricks for defining arbitrary distributions in WinBUGS or JAGS (Spiegelhalter *et al.* 2002, p. 36; McCarthy 2007, p. 201), while OpenBUGS offers a generic `dloglik` distribution (Spiegelhalter *et al.* 2011).
- BUGS has a confusing array of available variants (OpenBUGS/LinBUGS/WinBUGS/JAGS) and interfaces to R (`iBUGS`, `R2jags`, `R2OpenBUGS`, `R2WinBUGS`, `rbugs`, `rjags`, `runjags`), running on various platforms. WinBUGS and its R interface `R2WinBUGS` will run natively under Windows, and under Linux or MacOS via WINE (a Windows compatibility library which must be installed separately); OpenBUGS will run natively on Windows and Linux, but requires WINE to run on MacOS, and its standard R interface (BRugs) is not available from the central R package repository (CRAN) and will only run on Windows. JAGS will run on all three platforms, but is incompatible with some WinBUGS extensions (GeoBUGS, PKBUGS, WBDiff), and has several different R interface packages. Even the BUGS experts present at the meeting had a hard time determining which versions could run on which platforms!
- BUGS often has difficulty with complex, parameter-rich models. Reformulating models in statistically equivalent but computationally more stable and efficient forms can often help, but doing so requires a great deal of experience and/or understanding of the theory underlying the sampling algorithms (or simple trial and error).
- BUGS enforces a Bayesian perspective, which users may not prefer, although a relatively new method called *data cloning* (Lele 2007; Ponciano *et al.* 2009), implemented in the R package `dclone` (Sólymos 2010), leverages the power of MCMC to do frequentist analyses.
- Because BUGS uses Bayesian MCMC methods, users are confronted with a number of additional decisions about which priors are appropriate, how many chains to run for how long and how to assess convergence. It may be especially difficult to detect problems with unidentifiability (models whose parameters cannot be estimated from the available data: see Section 'Keep it simple, at least to start'); deterministic approaches implemented in R and ADMB are more likely to (correctly) report failure to fit such models.

For further comparisons between ADMB and BUGS, see Pedersen *et al.* (2011).

Case studies

We brainstormed to develop a diverse collection of problems. In most cases, we had access to a real, sampled data set. To assess metrics such as bias, mean squared error and coverage that can only be computed when the truth is known, we wrote simple programs to simulate new data sets, either with parameter values based on the original fit or with reasonable values in the same general region of parameter space. We then used an automated framework to fit each model to each of the simulated data sets, gather the estimated parameters and estimate bias, variance, mean squared error and coverage. We attempted to implement identical statistical models with each computational tool (R, ADMB and BUGS), so the parameter estimates should have been identical for all models for a given simulated data set, but in fact this procedure was a good test of the robustness of the approaches. Even with a correct model all the programs would sometimes fail to converge to the maximum likelihood estimate. (Stochastic approaches such as the MCMC algorithms implemented by BUGS give slightly different results on each run, but the answers should at least have been very similar, taking into account the differences between Bayesian and likelihood-based estimation). Furthermore, estimating reliable confidence intervals that incorporate all relevant components of variation is often the most unstable and difficult part of an analysis, and the different packages often used different approaches to confidence interval estimation.

Almost all data analyses involve an iterative process of adjusting the statistical model to fit the characteristics of the data (McCullagh & Nelder 1989, pp. 390–391). For the purposes of comparison among the three software packages, we tried to stick to our originally proposed model, even if data exploration revealed problems such as overdispersion. This approach kept the scope of our exploration contained and was also useful because adjusting models to handle deviations from the originally proposed model often had to be carried out differently in different packages. In the associated write-ups of the methods, however, we felt free to explore sensible variations of the original models, even if they could only be implemented in a subset of the packages we covered.

Advice

It is hard to find accessible, practical advice on making numerical optimization work better: there is no 'Dummies' Guide to Ecological Model Fitting', and the guides that exist tend either to assume a high level of mathematical and computational sophistication or to be scattered across a wide range of fields:

we suggest McCullough (2004), Press *et al.* (2007) and Jones *et al.* (2009, ch. 12) as reasonable starting points. In this section, we give some recommendations that emerged from our working sessions.

FOLLOW THE HERD

It is generally wise to use the tools that are most popular among researchers in your area. In addition to the greater availability of examples and help, it will also be easier to convince reviewers of the validity of familiar techniques, and reviewers will be more likely to detect potential problems with the methods used. That said, one should not hesitate to try new methods when they are clearly more powerful than classical ones, for example approaches based on modelling discrete distributions rather than transforming data (O'Hara & Kotze 2010), or mixed models for handling data with unbalanced blocks (Pinheiro & Bates 2000).

Similarly, when formulating a problem, it is often a good idea to use existing definitions, both because they will be more easily accepted by reviewers and peers and because the stability and other numerical properties of an established model are more likely to have been considered by experts. For example, Vonesh & Bolker (2005) used a novel equation to model a unimodal (hump-shaped) relationship for predation risk as a function of prey size. While they did get useful results, they later realized (Bolker 2008) that they had found only one of two possible 'best' fits to the data, that is, a local maximum of the likelihood surface. A previously proposed model (Persson *et al.* 1998), which we used in the [`tadpole`] project, allows for similar shapes but appears to have only a single global maximum. Out of many possible relationships, the [`wildflower`] project chose to use a logistic relationship between the number of seed pods and the probability of flowering, in part so that the model would fit into a standard generalized linear mixed modelling framework.

When a nonstandard formulation is used, the results should be compared to the standard definition, and the reason for any deviations should be well understood.

KEEP IT SIMPLE, AT LEAST TO START

Most complex models are extensions of simpler models. During the initial stages of model fitting, it often makes sense to fit reduced versions of the model to build up working code blocks, to find potential problems with the data and to get initial estimates of parameters for more complex models (see next section). For model/code development, choose a subset of your data that makes your code run fast during the debugging phase.

In their Chapter 19 on 'Debugging and speeding convergence', focussed on BUGS but applicable to complex models in general, Gelman & Hill (2006) say:

> 'Our general approach to finding problems in statistical modelling software is to get various crude models (for example, complete pooling or no pooling, or models with no predictors) to work and then gradually build up to the model that we want to fit. If you set up a complicated model and you cannot get it to run – or it will but its results do not make sense – then either build it up from scratch, or strip it down until you can get it to work and make sense.'

Their illustration of this concept (fig. 19.1, p. 416) shows a continuum between simple models that can be fit successfully and complex models that cannot be fit, or that give nonsensical results. Uriarte & Yackulic (2009) show a similar figure, although they emphasize inference more than the nuts and bolts of getting a working model.

In extreme cases, ecologists try to fit *unidentifiable* models – models that cannot, sometimes in principle and more often in practice, be fitted at all with the available data. This happens especially to inexperienced and enthusiastic modellers, but even experts can get caught occasionally. Bolker (2009) says:

> '[u]nfortunately, it is hard to give a general prescription for avoiding weakly unidentifiable parameters, except to stress common sense again. If it is hard to imagine how one could in principle distinguish between two sources of variation – if different combinations of (say) between-year variation and overdispersion would not lead to markedly different patterns – then they may well be unidentifiable.'

There are more formal methods for detecting unidentifiability (Luo *et al.* 2009; Cole *et al.* 2010; Lele *et al.* 2010), but they are rather technical: common sense, and (in the spirit of the previous section) using models that are similar to ones that have previously been successfully fitted by other researchers in the field is the only advice about identifiability that fits within the scope of this paper.

Some specific suggestions to overcome problems when fitting models to data:

- Initially, omit complexities of the model such as random effects, zero-inflation or imperfect detection. The 'complete pooling' referred to by Gelman and Hill above means leaving the blocking factor out of the model completely, while 'no pooling' means fitting the blocking factor as a fixed effect. In some cases, such as analysis of nested designs (Murtaugh 2007), averaging over blocks gives exactly the same answers for the fixed effects as a more complex mixed model. Do not fit a complex model if a simple one will do.
- Hold some parameter values constant, or in Bayesian models use strong priors such as normal distributions with large precision (i.e. small variances) to restrict parameters to a narrow range.
- Reduce the model to a simpler form by setting some parameters, especially exponents or shape parameters, to their null values. For example, fit a model with Poisson errors first before trying one with negative binomial errors, or fit an exponential survival model before a more complex model with Gamma- or Weibull-distributed survival. ADMB formalizes this approach by defining *phases*, where some model parameters are initially held constant at their initial values, but estimated along with the other parameters in later phases.

PICK REASONABLE STARTING VALUES

Specifying good initial parameter values is important when fitting complex models. New users are often surprised by this requirement – if we already know the parameters, why are we spending so much effort to fit the model? – but starting the optimization *sufficiently* close to the best values often makes the difference between success and failure.

- ADMB's optimization methods are sufficiently robust that one can often get by without explicitly stating initial parameter values. In ADMB, unconstrained parameters are initially set to zero by default and constrained parameters are set to the midpoint of the constraint region. However, the [`weeds`] project demonstrated a situation where ADMB found a false minimum when starting from the default set of all-zero parameters.
- BUGS can in principle be used without initial parameter values; initial values for the Markov chains are chosen randomly from the prior distributions of the parameters. For complex problems, or for models with unobserved (latent) categorical variables in the definition, WinBUGS is very likely to crash or have extreme difficulty converging when sensible initial values are not set explicitly.
- R's tools for fitting models almost all require initial parameter values to be specified, although the nonlinear least-squares function `nls` does allow for a class of 'self-starting' models. R's optimizing functions are more likely than ADMB's to be sensitive to the choice of starting values.

The most important step in specifying initial parameter values is simply to make sure that the values are of the right order of magnitude. Problems at this stage can happen when a user takes a model from the literature, or inherits model-fitting software from a colleague, whose parameter definitions they do not understand. If you understand the definitions of parameters and the biology of your system, you should be able to guess parameter values at least within one or two orders of magnitude. For parameters that are very uncertain (and whose values must be positive), estimating the logarithms of the original parameters (e.g. estimating the log of the growth rate rather than the growth rate itself) can be helpful.

Here are some other strategies for finding reasonable starting values for parameters:

- If possible, plot the data and 'eyeball' initial values for parameters, or overlay predictions from suggested starting values to check that the predictions for the initial values are in the same range as the observed responses.
- Fit simple models to subsets of the data. For example, approximate the initial slope of a saturating function by fitting a linear regression model, or estimate an intercept by averaging the first 5% of the data, or estimate an asymptote by averaging the last 5% of the data.
- Fit approximate models to transformed data. For example, estimate an exponential growth rate by fitting $\log(y)$ as a function of x, or the parameters of a power function by fitting $\log(y)$ vs. $\log(x)$. Similarly, estimate a Holling type II or Michaelis–Menten function $y = a/(b+x)$ by fitting a linear regression to the inverse of y: $1/y = (1/a) \cdot x + (b/a)$. If zeros in the data preclude this transformation, either omit them or add a small constant – the goal of this step is a decent first approximation, not precise answers.
- As in Section 'Keep it simple, at least to start', start by building a model that is a restricted version of the target model, and use its estimated parameters as starting points for estimation in the full model.

Even these procedures can be difficult for very complex data sets that are hard to represent graphically. In this case, one must fall back on the 'know the units of your parameters and use common sense' suggestions above.

RESHAPE THE GOODNESS-OF-FIT SURFACE

All model-fitting exercises can be thought of geometrically, as an attempt to find the highest peak of the likelihood/posterior surface (representing the maximum likelihood estimate or the mode of the posterior density in Bayesian analyses) and explore its neighbourhood (to construct confidence or credible regions). In general, numerical estimation and calculation of confidence intervals works best for likelihood surfaces with circular contours. Strongly anisotropic contours such as long and skinny ellipses, or banana shapes, represent differences in variance among parameters; ellipses that run at angles to the axes represent correlated parameters; and nonelliptical contours represent parameters whose sampling distribution or posterior densities are non-Gaussian (Bolker 2008, fig. 6.14).

One can often improve the shape of the likelihood surface, and hence the stability and efficiency of model fitting, without changing the biological meaning of the model or its goodness-of-fit to the data, by changing the way the model is parameterized. Like specifying starting values, the need to change parameterizations varies somewhat among software tools. Depending on the robustness of the tool (ADMB is generally the most robust, followed by R, JAGS and WinBUGS in that order), reparameterization may be unnecessary, helpful or essential.

Remove eccentricity by scaling

Parameters with strongly different scales lead to likelihood surfaces with different slopes or curvatures in different directions. In turn, such surfaces can cause numerical problems for methods that (i) approximate the slope of the goodness-of-fit surface (e.g. most of the built-in optimization methods in R use so-called *finite-difference approximations* to compute derivatives) or (ii) solve matrix equations to find the best directions in parameter space to explore, or to estimate the curvature of the surface at the best fit in order to construct confidence intervals for the parameters. Rescaling parameters by appropriate constants can thus improve the robustness of fit, as well as improving parameter interpretability (Schielzeth 2010). For interpretation, researchers often scale the predictor variables by their standard deviations (Gelman & Hill 2006). For numerical stability, the goal is for the derivatives of the scaled variables to be within an order of magnitude of each other. Similarly, it is useful to scale the parameters so that their expected starting values are all within an order of magnitude.

In its original form, the [weeds] project problem had parameters that ranged by three orders of magnitude, requiring parameter scaling.

The parscale option in R's optim function sets implicit scales on the parameters. For example, using control=list(parscale=abs(startvals)) scales the parameters according to their starting values startvals (this works if all the starting values are nonzero), while parscale=abs(coef(fit)) would work to scale the parameters when re-starting a fit (e.g. from a stopping point of an algorithm that might not be a true optimum). However, some of the optimizers available in contributed packages do not allow for scaling in this way – although scaling can always be performed manually. The R package optimx provides parameter scaling for a wider range of optimization algorithms.

The set_scalefactor option in ADMB allows parameter scaling, but only in models without random effects. In models with random effects, any necessary parameter scaling must be performed manually.

Remove correlation in the likelihood surface

Strongly correlated likelihood surfaces can be difficult for both hill-climbing algorithms (i.e. ADMB, R optim) and MCMC algorithms (BUGS).

Centring. One simple strategy for removing correlation among the parameters is to centre the predictor variables, by subtracting their mean or by subtracting some meaningful round number near the centre of the distribution of the predictor variables (e.g. one might choose to subtract 10 rather than $\bar{T} = 10{\cdot}792$ from a temperature variable, thus using 'difference from 10 °C' rather than 'difference from 10·792 °C' as the new predictor). Centring redefines the intercept or reference level of the model and strongly reduces or eliminates the correlation between intercept and slope parameters. While it is often recommended for purposes of interpretability (Schielzeth 2010), it can also improve fitting significantly. For example, the BUGS code used for the [owls] project converged much faster for centred than for noncentred predictors, although the [wildflower] project did not show a similar difference.

Centring only makes sense when the parameters enter the model in a linear way, and when the relevant parameter is not constrained to be positive. For example, switching from $y = \exp(a + bx)$ to $y = \exp(a + b(x - \bar{x}))$ leaves the meaning of the model unchanged, but switching from $y = ax^b$ to $y = a(x - \bar{x})^b$ changes the model fundamentally. (On the other hand, changing from $\log(y) = a + b \log(x)$ to $\log(y) = a + b(\log(x) - \overline{\log(x)})$, or even $\log(y) = a + b(\log(x) - \log(\bar{x}))$, is OK.)

Orthogonalization. If parameters are still correlated after centring, one may be able to change parameters to reduce the correlation. This can be done formally by working with matrix transformations of the original parameters. More informally, one can work with the known structure of the problem to reduce correlation. For example, the shape (a) and scale (s) parameters of a Gamma distribution are often strongly correlated, leading to a curving ridge in the likelihood surface. If so, reparameterizing the distribution in terms of the mean ($=a{\cdot}s$) and variance ($= a{\cdot}s^2$) will improve fitting. Changing the parameterization of a nonlinear model can separate the problem in such a way that uncertainty does not contaminate all of the parameters. For example, the [weeds] project used a model for the expected density of weeds w at time t: $w(t) = b_1/(1 + b_2 \exp(-b_3 t))$, where $b_1 = w_\infty$ is the asymptotic density, b_2 is a combination of the initial density w_0 and the asymptotic density, and b_3 is the maximum growth rate, also proportional to the asymptotic density. The data for the weeds example show only an accelerating curve, with little evidence of saturation, making the asymptote (w_∞) hard to estimate. Because b_1, b_2 and b_3 all involve w_∞, the estimation problem is challenging (although ADMB can solve it if given reasonable starting values). Re-parameterizing the model to change the second parameter from b_2 to w_0 separates the poorly determined asymptotic density w_∞ from the other parameters (w_0, b_3), making the model fitting faster and more robust.

Make contours elliptical

Finally, by transforming parameters appropriately, for example log-transforming, one can make the contours of the likelihood surface more elliptical or equivalently make the log-likelihood surface a quadratic function of the transformed parameters: for example, log transformation is essential in the [theta] project. While most optimization methods can handle smooth surfaces that are not quadratic (surfaces with discontinuities or sharp transitions present special challenges), quadratic surfaces have particular advantages for inference and computation of confidence intervals.

- Wald significance tests and confidence intervals, which are based on a quadratic approximation to the likelihood surface at its maximum, are most reliable when the surface is nearly quadratic. Alternative approaches such as likelihood profile confidence intervals relax this requirement, but require much more computation, increase the chance of convergence problems and may not be available in all software tools.
- Bayesian MCMC approaches do not depend on quadratic surfaces, but many convenient analytical approximations such as the Bayes (Schwarz) information criterion (BIC) and deviance information criteria (DIC; Spiegelhalter *et al.* 2002) do. In particular, they depend on multivariate normality of the posterior distribution, which is equivalent to the log-posterior surface being quadratic.
- When the posterior density is multivariate normal, all Bayesian posterior distributions are symmetric and hence the two alternative approaches for constructing Bayesian confidence intervals, quantiles and highest posterior density intervals, agree with each other (and with frequentist confidence intervals, if the priors are uninformative).

CONSTRAIN PARAMETERS

When 'box constraints' (independent bounds on each parameter) are available, it is often a good idea to specify them for each parameter. This prevents parameters wandering to extreme values where the surface may be very flat (and hence derivatives may be calculated poorly, or MCMC chains get stuck for a long time), or where numeric underflow or overflow may lead to errors. (Numeric under- or overflow occurs when some intermediate values in a computation are too small or large to be represented as numeric floating-point variables at a given precision. For example, in a typical modern computing environment values smaller than about 10^{-308} are rounded down to zero, and values larger than about $\pm 10^{308}$ are flagged as infinite. While these problems can sometimes be solved by increasing the precision of the calculation, it is usually more useful to either rearrange the computation (for example fitting parameters on a logarithmic scale) or avoid problematic regions of parameter space by setting constraints.) The [`weeds`] project required that the parameters be kept positive; either fitting log-transformed parameters or setting box constraints worked well.

Box constraints are available in ADMB, and constraints are reasonably easy to set up in BUGS/JAGS by imposing priors. The `I()` operator in WinBUGS/OpenBUGS or the `dinterval()` operator in JAGS can be used to impose truncation on an existing prior distribution. Box constraints are less widely available in R. The main implementation of box constraints in base R, `optim`'s `L-BFGS-B` method, is more fragile than the other `optim` algorithms: for example, it fails on `NA` values when other optimizers can sometimes keep going. The `optimx`, `minqa` and `nloptr` packages in R do offer a variety of box-constrained algorithms.

Of course, as with starting values, one needs to know enough about the problem to be able set reasonable bounds on the parameter: trying to be conservative by setting extremely wide bounds (such as $\pm 10^8$) both negates any advantages of constraining the parameter in the first place and may lead to crashes if the program tries to evaluate the objective function at the bounds as part of its start-up process.

In addition to the general value of box constraints for keeping optimization algorithms within sensible bounds, there are some situations where an estimated parameter really lies on the boundary of its set of possible values. Common cases are random-effects variances or overdispersion parameters whose best estimate is zero, or probabilities in a demographic model that are estimated as zero due to a small sample. In this case, using constraints to bound the variance parameter at zero works better than the alternative strategy of fitting the variance parameter on the log scale, because transformation will just move the best estimate of the parameter to $-\infty$. Researchers who inappropriately try to use transformation when the best-fit parameters are really on the boundary are likely to see both parameter estimates with very large magnitudes (and huge standard errors) and warnings about convergence; both symptoms arise because the optimization algorithm is trying to move towards a point at infinity on a nearly flat surface.

Unfortunately, fitting with constraints can also add to the challenge of optimization and inference. When the best-fitting parameters are on the boundary, optimization algorithms can behave badly. More generally, many of the standard approaches to inference, such as inverting the negative Hessian matrix to estimate the variance–covariance matrix of the parameters, finding likelihood ratio test intervals, or using AIC, are not applicable when parameters are on the boundary of their feasible space (Pinheiro & Bates 2000; Hughes 2003; Bolker, 2008). In some cases, simplifying the model can avoid these problems, for example removing random effects with estimated variances of zero.

CONSIDER ALTERNATE OPTIMIZERS

If none of the previous approaches have worked, one can attempt to switch optimization algorithms, change to a different implementation of the same algorithm or tune the parameters that control the behaviour of the algorithm, such as the convergence tolerance. These tricks are a last resort: if all of the previously discussed problem-taming strategies have failed, then these variations may not help. Furthermore, BUGS offers little control of the MCMC samplers used, and ADMB uses a single (albeit extremely robust) optimizer with few tunable parameters. For those cases where there is room for improvement, R does provide many different optimizers. A large variety of add-on packages augments the half-dozen choices available within the built-in `optim()` function (see the useful R Optimization Task View at http://cran.r-project.org/web/views/Optimization.html#GeneralPurposeSolvers). In particular, the `optimx` package (Nash & Varadhan 2011), used in the `min`, `tadpole`, and `weeds` projects, provides a wrapper for a variety of optimizers coded in other packages. Roughly speaking, users can choose among (i) derivative-free optimizers, generally robust but slow, and particularly useful for problems with thresholds (the Nelder-Mead and BOBYQA optimizers are good examples of this class); (ii) local optimizers that use derivative information in some form (conjugate-gradient and variable-metric methods) and (iii) stochastic optimizers that handle problems with multiple peaks, at the cost of greatly increased tuning needs and greatly decreased speed (simulated annealing, genetic algorithms). Bolker (2008, chapter 7) and Nash & Varadhan (2011) provide further details.

SIMULATE YOUR DATA

As has been pointed out before (Hilborn & Mangel 1997; Hobbs & Hilborn 2006; Bolker 2009; Kéry & Schaub 2012), simulating data that matches the estimation model is a good idea. This is a best-case scenario – simulated data are always well behaved, and the estimator is correctly specified because we know the distributions that were used to generate the data – but even in this best-case scenario, a complex model can fail. Fitting a model to simulated data rather than to real data separates the process of identifying coding errors from the challenge of understanding whether your model is appropriate for your data in the first place.

• Some models in R have a built-in `simulate` method that will simulate data consistent with a fitted model, but one usually needs to start by fitting a model, so this tool is actually more useful for testing model output than for generating input to models. However, R has a sufficiently large set of low-level tools, such as random-number generators for a wide range of distributions, with which users can simulate almost any model. All of our projects used R to simulate test data with which to evaluate the reliability of the model fits.
• If all parameters are completely defined, that is, the parameters are set to constants rather than having priors defined, BUGS will simulate data from the appropriate distribution (in `R2jags`, one must specify `DIC=FALSE` to stop JAGS from trying to compute goodness-of-fit statistics).
• ADMB has built-in random-number generators and so can also be used as a simulation tool, although many users prefer to simulate in R.

SPEED THINGS UP

A fitting method may be reasonably robust but too slow. For a single estimate, one might be willing to wait an hour or a day for an answer, but if one wants to use the method on many data sets or use a computationally intensive method such as bootstrapping or profile likelihood to find confidence intervals, slow methods are infeasible.

One option is to switch to another platform, for example from R or BUGS to AD Model Builder or from BUGS to a custom MCMC sampler written in R. Re-coding an estimation method is tedious, but often much faster than coding it in the first place, because the major problems with the model or the data will have been ironed out. Furthermore, having a comparable fit from a completely independent method greatly reduces the chances of undiscovered bugs or undiagnosed convergence failures.

Some approaches, in particular the MCMC algorithms of BUGS, can be accelerated by the use of distributed computation – multiple Markov chains can be run on different processors, either within a single multi-core machine, on a computational cluster or via cloud services, for example by using built-in capabilities of JAGS or the `bugsparallel` package (http://code.google.com/p/bugsparallel/) for WinBUGS.

New, faster tools are always on the horizon. Some recent candidates are INLA, a package for complex (especially spatio-temporal) Bayesian models in R (Eidsvik *et al.* 2012; Ruiz-Cárdenas *et al.* 2012); Stan (http://mc-stan.org/), a BUGS-like language that promises greater speed and modularity; `LaplacesDemon` (Hall 2012), an R package that implements BUGS-like Bayesian samplers in a flexible way; and the Julia language (http://julialang.org/), which aims to combine the flexibility of R with the speed of lower-level compiled languages. However, not all ecologists want to be early adopters of new technology; using older, better-tested and better-documented tools has many advantages.

Unfortunately, the other alternatives for speeding up optimization, besides finding a faster computer, are package specific and often require great expertise in the underlying mechanics of the package.
• In R, computations can often be sped up by appropriate vectorization. For moderate acceleration, one can byte-compile R code. For large acceleration, one can re-write the likelihood function in a lower-level language such as C++. However, these changes will not help very much if the likelihood function is already relying mostly on operations that R executes efficiently, such as matrix manipulations, which are done by optimized system libraries.
• The largest potential speed gain for ADMB users is in the context of random-effects models, where using so-called separable functions can greatly reduce memory use and increase speed. See the ADMB-RE manual, and the [`wildflower`], [`owls`], and [`theta`] projects, for details.
• BUGS models can sometimes be sped up simply by changing the formulation of the model. In Pedersen *et al.* (2011), changing priors improved OpenBUGS's speed, although the same phenomenon was not seen when using JAGS on the same model in the [`theta`] project; the [`wildflower`] project achieved faster convergence by changing the form of the priors of the random-effect variances. Reparameterizing to remove correlations (See Section 'Remove correlation in the likelihood surface') can also speed convergence, as can adding redundant parameters (an advanced technique described by Gelman *et al.* (2008)). Although it may take considerable effort, re-coding one's own MCMC sampler from scratch, as recommended by Clark (2007), can sometimes pay off.

Discussion and conclusions

The breadth of knowledge required for successful modelling cannot be conveyed in a single article – the suggestions above are obviously just a starting point. We hope that interested readers will visit our collection of worked examples (https://groups.nceas.ucsb.edu/non-linear-modeling/projects), where they will find much more detailed and particular examples of modelling practise.

In the examples, we tried to cover a reasonably broad spectrum of problems, but we can easily identify topics that were left largely unaddressed. These include generalized additive models, spatial and spatiotemporal estimation problems and the estimation of systems defined in terms of continuous-time dynamics, such as differential equations or continuous-time Markov chains (Kristensen *et al.* 2004; Ionides *et al.* 2006; Wood 2006: Diggle & Ribeiro 2007).

While the variety of software tools can be confusing, it is good that multiple approaches, and even multiple implementations of the same approach, are available to ecologists. If they are FOSS, so much the better. Given how hard it is to be absolutely certain that a model is fitted correctly, it is extremely useful to compare results among software tools. We look forward to better integration among the various tools (beyond the improvements that were made as a result of our workshop), so

that researchers can switch between platforms and compare among methods without having to reformat their data or redefine their problems. Estimating the parameters of complex ecological models will never be simple, but the widening availability of powerful computational engines, the improvement of interfaces and the dissemination of basic principles and worked examples can ease the burden for ecologists who want to apply these tools to their data.

Acknowledgements

The National Center for Ecological Analysis and Synthesis supported this work. B.M.B. was further supported by an NSERC Discovery Grant. Any use of trade, product or firm names is for descriptive purposes only and does not imply endorsement by the US Government.

References

Adkison, M.D. (2009) Drawbacks of complex models in frequentist and Bayesian approaches to natural-resource management. *Ecological Applications*, **19**, 198–205.

Bolker, B.M. (2008) *Ecological Models and Data in R*. Princeton University Press, Princeton, NJ, USA.

Bolker, B. (2009) Learning hierarchical models: advice for the rest of us. *Ecological Applications*, **19**, 588–592.

Breiman, L. (2001) Statistical modeling: The two cultures. *Statistical Science*, **16**, 199–215.

Clark, J.S. (2007) *Models for Ecological Data: An Introduction*. Princeton University Press, Princeton, NJ, USA.

Cole, D.J., Morgan, B.J.T. & Titterington, D.M. (2010) Determining the parametric structure of non-linear models. *Mathematical Biosciences*, **228**, 16–30.

Crawley, M.J. (2002) *Statistical Computing: An Introduction to Data Analysis Using S-PLUS*. Wiley, Chichester.

Crawley, M.J. (2005) *Statistics: An Introduction Using R*. Wiley, Chichester.

Crawley, M.J. (2007) *The R Book*, 1st edn. Wiley, Chichester.

Diggle, P.J. & Ribeiro Jr, P.J. (2007) *Model-Based Geostatistics*. Springer, New York, NY, USA.

Eidsvik, J., Finley, A.O., Banerjee, S. & Rue, H. (2012) Approximate Bayesian inference for large spatial datasets using predictive process models. *Computational Statistics and Data Analysis*, **56**, 1362–1380.

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. & Hill, J. (2006) *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge, UK.

Gelman, A., van Dyk, D.A., Huang, Z. & Boscardin, J.W. (2008) Using redundant parameterizations to fit hierarchical models. *Journal of Computational and Graphical Statistics*, **17**, 95–122.

Gotelli, N.J. & Ellison, A.M. (2004) *A Primer of Ecological Statistics*. Sinauer, Sunderland, MA.

Griewank, A. & Corliss, G.F. (1992) *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, Philadelphia, PA, USA.

Hall, B. (2012) *LaplacesDemon: Complete Environment for Bayesian Inference*. R package version 12.10.01. URL http://cran.r-project.org/web/packages/LaplacesDemon/

Hilborn, R. & Mangel, M. (1997) *The Ecological Detective: Confronting Models with Data*. Princeton University Press, Princeton, NJ, USA.

Hobbs, N.T. & Hilborn, R. (2006) Alternatives to statistical hypothesis testing in ecology: A guide to self teaching. *Ecological Applications*, **16**, 5–19.

Hughes, A.W. (2003) Model selection using AIC in the presence of one-sided information. *Journal of Statistical Planning and Inference*, **115**, 397–411.

Ionides, E.L., Bretó, C. & King, A.A. (2006) Inference for nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, **103**, 18438–18443.

Jones, O., Maillardet, R. & Robinson, A. (2009) *Introduction to Scientific Programming and Simulation Using R*, 1st edn. Chapman & Hall/CRC, Boca Raton, FL, USA.

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

King, R., Morgan, B.M., Gimenez, O. & Brooks, S. (2009) *Bayesian Analysis of Population Ecology*. Chapman & Hall/CRC, Boca Raton, FL, USA.

Kristensen, N.R., Madsen, H. & Jørgensen, S.B. (2004) Parameter estimation in stochastic grey-box models. *Automatica*, **40**, 225–237.

Lele, S.R. (2007) Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. *Ecology Letters*, **10**, 551–563.

Lele, S., Nadeem, K. & Schmuland, B. (2010) Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association*, **105**, 1617–1625.

Link, W. & Barker, R. (2010) *Bayesian Inference with Ecological Applications*. Academic Press, London.

Ludwig, D. & Walters, C.J. (1985) Are age-structured models appropriate for catch-effort data? *Canadian Journal of Fisheries and Aquatic Sciences,* **42,** 1066–1072.

Lunn, D. (2009) The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, **28**, 3049–3067.

Lunn, D., Jackson, C., Best, N., Thomas, A. & Spiegelhalter, D. (2012) *The BUGS Book: A Practical Introduction to Bayesian Analysis*, 1st edn. Chapman & Hall/CRC, Boca Raton, FL, USA.

Luo, Y., Weng, E., Wu, X., Gao, C., Zhou, X. & Zhang, L. (2009) Parameter identifiability, constraint, and equifinality in data assimilation with ecosystem models. *Ecological Applications*, **19**, 571–574.

Magnusson, A. (2009) ADMB-IDE: Easy and efficient user interface. *ADMB Foundation Newsletter*, **1**, 1–2.

Maunder, M.N., Schnute, J.T. & Ianelli, J.N. (2009) Computers in fisheries population dynamics. *Computers in Fisheries Research* (eds B.A. Megrey & E. Moksness), pp. 337–372. Springer Netherlands, Dordrecht, Netherlands.

McCarthy, M. (2007) *Bayesian Methods for Ecology*. Cambridge University Press, Cambridge.

McCullagh, P. & Nelder, J.A. (1989) *Generalized Linear Models*, 2nd edn. Chapman and Hall, London.

McCullough, B.D. (2004) Some details of nonlinear estimation. *Numerical Issues in Statistical Computing for the Social Scientist*, chapter 8 (eds M. Altman, J. Gill & M.P. McDonald), pp. 199–218. Wiley, Chichester.

Millar, R.B. (2011) *Maximum Likelihood Estimation and Inference: With Examples in R, SAS and ADMB*. John Wiley & Sons, Hoboken, NJ, USA.

Murtaugh, P.A. (2007) Simplicity and complexity in ecological data analysis. *Ecology*, **88**, 56–62.

Nash, J.C. & Varadhan, R. (2011) Unifying optimization algorithms to aid software system users: `optimx` for R. *Journal of Statistical Software*, **43**, 1–14.

Nash, J.C. & Walker-Smith, M. (1987) *Nonlinear Parameter Estimation: An Integrated System in BASIC*. Marcel Dekker Inc., New York, NY, USA. Republished combined with the previous item in electronic form by Nash Information Services Inc., Ottawa, Canada, 1996.

O'Hara, R.B. & Kotze, D.J. (2010) Do not log-transform count data. *Methods in Ecology and Evolution*, **1**, 118–122.

Pedersen, M., Berg, C., Thygesen, U., Nielsen, A. & Madsen, H. (2011) Estimation methods for nonlinear state-space models in ecology. *Ecological Modelling*, **222**, 1394–1400.

Peng, R.D. (2009) Reproducible research and biostatistics. *Biostatistics*, **10**, 405–408.

Persson, L., Leonardsson, K., de Roos, A.M., Gyllenberg, M. & Christensen, B. (1998) Ontogenetic scaling of foraging rates and the dynamics of a size-structured consumer-resource model. *Theoretical Population Biology*, **54**, 270–293.

Pinheiro, J.C. & Bates, D.M. (2000) *Mixed-Effects Models in S and S-PLUS*. Springer, New York, NY, USA.

Ponciano, J.M., Taper, M.L., Dennis, B. & Lele, S.R. (2009) Hierarchical models in ecology: confidence intervals, hypothesis testing, and model selection using data cloning. *Ecology*, **90**, 356–362.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (2007) *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd edn. Cambridge University Press, Cambridge.

Quinn, G.P. & Keough, M.J. (2002) *Experimental Design and Data Analysis for Biologists*. Cambridge University Press, Cambridge, UK.

Reimann, C., Filzmoser, P., Garrett, R. & Dutter, R. (2008) *Statistical Data Analysis Explained: Applied Environmental Statistics with R*. Wiley, Chichester, UK.

Royle, J. & Dorazio, R. (2008) *Hierarchical Modeling and Inference in Ecology: The Analysis of Data from Populations, Metapopulations and Communities*. Academic Press, New York, NY, USA.

Ruiz-Cárdenas, R., Krainski, E.T. & Rue, H. (2012) Direct fitting of dynamic models using integrated nested Laplace approximations: INLA. *Computational Statistics and Data Analysis*, **56**, 1808–1828.

Schielzeth, H. (2010) Simple means to improve the interpretability of regression coefficients. *Methods in Ecology and Evolution*, **1**, 103–113.

Skaug, H. & Fournier, D. (2006) Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models. *Computational Statistics and Data Analysis*, **51**, 699–709.

Soetaert, K. & Herman, P.M.J. (2008) *A Practical Guide to Ecological Modelling: Using R as a Simulation Platform*, 1st edn. Springer, New York, NY, USA.

Sólymos, P. (2010) dclone: Data cloning in R. *The R Journal*, **2**, 29–37.

Spiegelhalter, D.J., Best, N., Carlin, B.P. & Van der Linde, A. (2002) Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society B*, **64**, 583–640.

Spiegelhalter, D., Thomas, A., Best, N. & Lunn, D. (2011) *OpenBUGS User Manual*, 3rd edn. URL http://www.openbugs.info/Manuals/Manual.html. Retrieved 17 Nov 2011.

Stevens, M.H.H. (2009) *A Primer of Ecology with R. Use R*. Springer, New York, NY, USA.

Underwood, A.J. (1996) *Experiments in Ecology: Their Logical Design and Interpretation Using Analysis of Variance*. Cambridge University Press, Cambridge, UK.

Uriarte, M. & Yackulic, C.B. (2009) Preaching to the unconverted. *Ecological Applications*, **19**, 592–596.

de Valpine, P. (2003) Better inferences from population-dynamics experiments using Monte Carlo state-space likelihood methods. *Ecology*, **84**, 3064–3077.

Vonesh, J.R. & Bolker, B.M. (2005) Compensatory larval responses shift tradeoffs associated with predator-induced hatching plasticity. *Ecology*, **86**, 1580–1591.

Warton, D.I. & Hui, F.K.C. (2011) The arcsine is asinine: the analysis of proportions in ecology. *Ecology*, **92**, 3–10.

Wood, S.N. (2006) *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton, FL, USA.

Zuur, A.F., Ieno, E.N., Walker, N.J., Saveliev, A.A. & Smith, G.M. (2009) *Mixed Effects Models and Extensions in Ecology with R*, 1st edn. Springer, New York, NY, USA.

Received 22 November 2012; accepted 19 February 2013
Handling Editor: Satu Ramula