A guide for Bayesian analysis in ADMB

Cole C. Monnahan — monnahc@uw.edu

August 6, 2014

Abstract

The point of this document is.....

Contents

1 Bayesian inference **1**

2 MCMC **2**
2.1 Workflow . . . 2
2.2 MCMC Phases . . . 2
2.3 Output files . . . 3
2.3.1 Meta data: The hst file . . . 3
2.3.2 Parameter draws: The psv file . . . 3
2.3.3 Derived quantity draws . . . 3
2.4 Restarting a chain . . . 4
2.5 Convergence diagnostics . . . 4
2.6 Starting values and scaling . . . 4
2.7 Metropolis-Hastings . . . 4
2.7.1 Algorithm . . . 4
2.7.2 MCMC Arguments . . . 5
2.7.3 mcprobe . . . 5
2.7.4 mcrb . . . 5
2.7.5 User-supplied correlation matrix . . . 6
2.7.6 Example MCMC . . . 6
2.8 Hybrid . . . 6
2.8.1 Algorithm . . . 6
2.8.2 Arguments . . . 7

3 Running analyses in R **7**
3.1 Code . . . 7

1 Bayesian inference

A short intro with citations to better papers. Maybe merge in what is currently shown?

In general, Bayesian inference is used to quantify the full range of uncertainty around both models and parameter values and incorporate findings from historical studies. In a fisheries specific context, Bayesian inference allows scientists to investigate the consequences of alternative management strategies.

2 MCMC

Markov chain Monte Carlo (MCMC) is a common algorithm used to sample and compute expectations with respect to complex and high-dimensional probability distributions. ADMB implements two different MCMC algorithms: the ubiqitous Metropolis-Hastings and a Hamiltonian (or "hybrid") sampler, both described in this section.

John Kruschke in "Doing Bayesian Data Analysis: A Tutorial with R and BUGS" has a simple example that illustrates the mechanics of the Metropolis-Hastings algorithm. A politician lives on a long chain of islands. The politician's goal is to stay in the public eye by traveling to each island proportional to their relative population. In other words, he will spend more time on highly populated islands and less time on less populated islands. He flips a fair coin to propose a move to an island east or west of his current location. If the proposed island has a higher population than his current island he moves. If the proposed island has a smaller population than his current island he will only move with a probability $P_{proposed}/P_{current}$ where $P_{proposed}$ is the population of the proposed island and $P_{current}$ is the population of the current island.

All MCMC algorithms work in this same general way. The user will start the algorithm at an arbitrary starting point, and the algorithm will propose new parameter sets or states. The algorithm will propose new states (i.e. parameter draws) and move to that state, or not, depending on its density relative to the current state. Just like the politician, the algorithm will spend most of the time in states with high densities.

MCMC algorithms typically need a "burn-in" period, especially if the starting point is in a region with low density. The goal of a MCMC algorithm is to explore areas with the highest probabilities in a distribution. It will perhaps take thousands of draws for the algorithm to find the peaks of the distribution, especially if it starts in a low density region. Discarding these intial draws allows the algorithm to explore the main regions of the target distribution.

Autocorrelated MCMC samples are close together and probably not a true represenation of the underlying distribution. Samples are typically "thinned" every nth-step to generate uncorrelated samples. The tradeoff is that excessive thinning increases run-time for the algorithm.

The R package `CODA` is a commonly used tool for checking MCMC diagnostics such as autocorrelation.

2.1 Workflow

The following steps outline the basic workflow typically used for conducting a MCMC in ADMB.

1. Build, run, and verify an ADMB model. This model must explicitly include the contribution of the priors to the objective function, such that the ADMB estimate is the posterior mode, rather than a maximum likelihood estimate.

2. Run an MCMC using the command line argument `-mcmc` N `-mcsave` $N_{\texttt{save}}$ (among other options, see below). The thinned draws are discarded, leaving a total of $N_{\text{out}} = N/N_{\text{save}}$ saved draws. For example `-mcmc 1e6 -mcsave 1000` will run 1 million draws but only save (i.e."thin") every 1000th, for a total kept of $Nout = 1000$.

3. After completion, run the model again with argument `-mceval`. This command tells ADMB to loop through the saved iterations (in the `.psv` file) and execute in the `mceval_phase()`.

4. Pull results into R or other program to ensure the sample is sufficiently thinned, either visually or with tools using, for example, the `CODA` package.

5. If necessary, rerun the chain with more thinning, drop the first part of the chain as a "burn-in," or run longer for more iterations.

6. Make whatever Bayesian inference is desired using the $Nout$ independent samples.

2.2 MCMC Phases

ADMB is designed with two phases that are used to produce MCMC output: (1) the `mcmc` phase and (2) `mceval` phase. While the use of these phases is not common (is this true??) among other MCMC software,

and may be a source of confusion for new ADMB users, they provide a powerful and efficient framework for MCMC analyses.

The `mcmc` phase is the one with which most people are already familiar. This is where ADMB generates new parameter sets by proposing a set, and then determining whether to move there or stay at the current set. This process is repeated N times, and how sets are proposed depend on the algorithm used (see section 2.7 and 2.8). During this phase, the $Nout$ saved parameter values are written to a `.psv` file (described below). Note that if `-mcsave` *nsave* is not specified, ADMB will run the MCMC but no values will be saved.

The `mceval` is an optional phase that is designed to be run after the `.psv` file has been produced. During this phase, ADMB loops through the $Nout$ parameter combinations in the `.psv` file and reruns the `PROCEDURE_SECTION` in the `mceval()` phase. This phase is extremely powerful because it allows the user to minimize wasted calculations by parsing calculations into two groups: those that affect the objective function (i.e. posterior calculations for Bayesian analyses) and those that do not. Calculations done for discarded draws (which often is most iterations) simply slow down the analysis. Thus, an analysis can be made to run faster by minimizing calculations in the `mcmc` phase. For example, a user may want to extrapolate (e.g. project a time series into the future) or calculate values derived from the parameters and intermediate values. By putting these calculations inside an `mceval_phase` clause they are only done for saved draws and the MCMC will run faster. In practice, some chains need to be thinned significantly more than 1 in 1000, so the time saved can be substantial, especially if the `mceval_phase` calculations are time-consuming.

While the `mceval` phase was designed specifically for MCMC analyses, it can be coopted for use in other types of analyses. In essence it is a convenient framework in which to get ADMB to quickly evaluate arbitrary sets of parameters, while only initializing once. Examples of alternative uses are the SIR algorithm (**??**), evaluating a grid of points for plotting and exploration of the posterior surface, or trying random parameter sets to investigate local minima. Getting ADMB to evaluate these parameter sets is as simple as writing them to the `.psv` file and then executing ADMB with the option `-mceval`. See section (2.3) for details on how to do this.

2.3 Output files

2.3.1 Meta data: The hst file

2.3.2 Parameter draws: The psv file

During the `mcmc` phase, saved parameter values, in bounded space, are written to a binary file called `<model name>.psv`. This file can be read into R using the following commands:

```
psv <- file("<model name>.psv", "rb")
nparams <- readBin(psv, "integer", n=1)
mcmc <- matrix(readBin(psv, "numeric", n=nparams*Nout), ncol=nparams, byrow=TRUE)
close(psv)
```

The first element in the `.psv` file is the number of active parameters in the model, which then tells R how to parse the following elements into parameter values. Note that the value of $Nout$ in `nparams*Nout` depends on `Nmcmc` and `mcsave` and must be specified manually. This is the main file that was designed to be used to extract MCMC draws from ADMB. However, this file only contains parameter values and not derived quantities or other quantities of interest (e.g. MSY or biomass trajectories) which often are of interest.

2.3.3 Derived quantity draws

A simple way of extracting this information is to bypass the `psv` file altogether and use a `C++` function to write a `.csv` file containing whatever elements are desired. This can be accomplished inside the ADMB `.tpl` file with just a few lines of code. Inside the `DATA_SECTION` section use the following code to create an IO object that writes values to a `.csv` file, similar to the function `cout` which prints to screen.

```
!!CLASS ofstream MCMCreport("MCMCreport.csv",ios::app);
```

Then, inside the `PROCEDURE_SECTION` the function can be used to write both parameters, derived quantities, or other information about the model.

```
if(mceval_phase()){
  if(header==1) {
      MCMCreport << "a,b,NLL,ab" << endl;
      header=0;
  }
  MCMCreport << a <<"," << b << "," << NLL << "," << ab << endl;
}
```

The `MCMCreport` object is used just like `cout` and is executed only during the `mceval` phase so that only saved values are written to the file. Naturally this code can be used anywhere in the procedure section, and this may be a useful diagnostic tool in some situations. New draws are appended to the `MCMCreport.csv` file so that it must be deleted in between MCMC runs.

2.4 Restarting a chain

How to restart a chain if you need more samples. How does this work?

2.5 Convergence diagnostics

Burn-in and thinning, how to check this? What happens if not independent?

2.6 Starting values and scaling

Where the algorithm starts from (MLE) and the scaling process (default and user options). Must discard these draws!

2.7 Metropolis-Hastings

The default MCMC algorithm used by ADMB is the Metropolis-Hastings (MH) algorithm. This algorithm has been around for decades, is simple to implement and used widely.

This algorithm will be most efficient when the posterior surface mimics a multivariate Normal distribution.

2.7.1 Algorithm

Let

$$\begin{aligned} f &= \text{the ADMB objective function} \\ c &= \text{an unknown normalization constant} \\ Xcur &= \text{current parameter vector} \\ Xprop &= \text{a proposed parameter vector} \\ U &= \text{a randomly drawn uniform value in [0,1]} \end{aligned}$$

Then

$$Xnew = \begin{cases} Xprop & \text{if} \quad U \leq \dfrac{cf(Xprop)}{cf(Xcur)} \\ Xcur & \text{otherwise} \end{cases} \tag{1}$$

The proposal (or "jump") function proposes new parameter vectors given the current set. The default behavior for ADMB is to use a multivariate normal distribution[1] centered at the current vector:

$$Xprop \sim MVN(Xcur, \Sigma)$$

where Σ is the covariance matrix obtained by inverting the Hessian at the posterior mode.

In ADMB there are options to modify the proposal function to achieve better efficiency.

2.7.2 MCMC Arguments

`-mcmc N`	Run N MCMC iterations
`-mcsave N`	Save every Nth MCMC iterations
`-mcscale N`	Rescale step size for first N iterations

Table 1: ADMB runtime arguments for Metropolis-Hastings MCMC

2.7.3 mcprobe

2.7.4 mcrb

The `-mcrb N` option (which stands for rescaled bounded) alters the covariance matrix used to propose new parameter sets in the MH algorithm. Its intended use is to create a more efficient MCMC sampler so the analyses run faster.

The option will be most effective under circumstances where the correlation between parameters at the MPD is higher than other regions of the parameter space. In this case, the algorithm may make efficient proposals at the MPD, but inefficient proposals in other parts of the space. By reducing the correlation using `mcrb` the proposal function may be more efficient on average across the entire parameter space and require less thinning.

The `mcrb` option is a set of calculations performed on the original correlation matrix, as follows.

$$\boldsymbol{\Sigma}_{\text{old}} = \begin{bmatrix} 1 & \cdots & \rho_{1,n} \\ \vdots & \ddots & \vdots \\ \rho_{n,1} & \cdots & 1 \end{bmatrix} \quad \text{The original correlation matrix}$$

$$\mathbf{L} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ L_{n,1} & \cdots & L_{n,n} \end{bmatrix} \quad \text{Lower Choleski decomposition of } \boldsymbol{\Sigma}_{\text{old}}$$

$$\hat{\mathbf{L}} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ L_{n,1}^{N/10} & \cdots & L_{n,n}^{N/10} \end{bmatrix} \quad \text{Raise elements to power user supplied } N$$

$$\tilde{\mathbf{L}} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\hat{L}_{n,1}}{|\hat{L}_{n,\cdot}|} & \cdots & \frac{\hat{L}_{n,n}}{|\hat{L}_{n,\cdot}|} \end{bmatrix} \quad \text{Normalize rows of } \hat{L}$$

$$\boldsymbol{\Sigma}_{\text{rb}} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T \quad \text{Calculate new correlation matrix}$$

By working with the Choleski decomposition of the correlation matrix, the algorithm ensures that the rescaled bounded matrix used in the MCMC remains a valid correlation matrix (i.e. positive definite).

[1] Technically a bounded multivariate normal

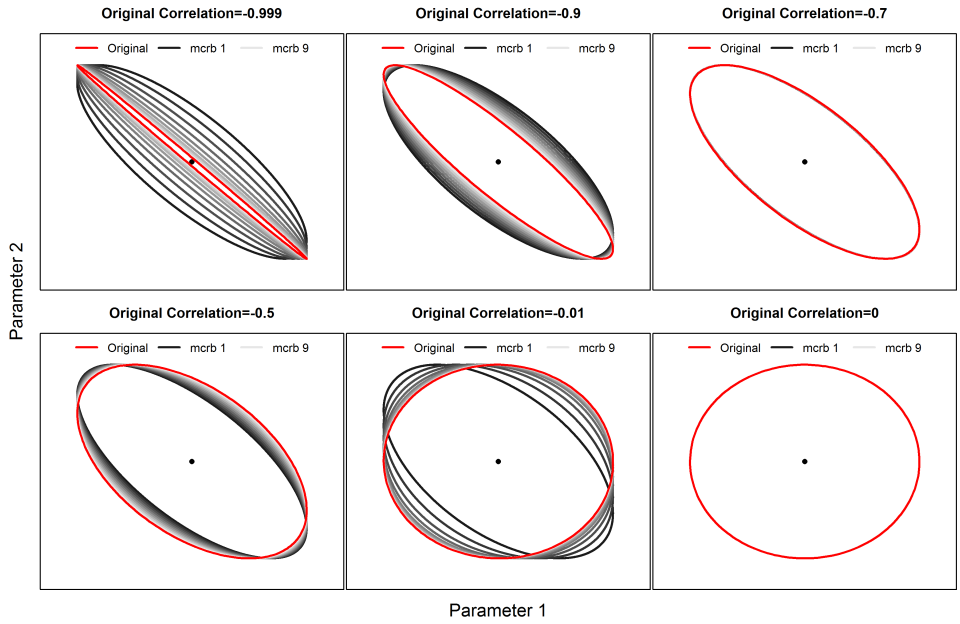

Figure 1: The effect of `mcrb` on a variety of correlations between two hypothetical parameters. Note that the effect of setting $N = 9$ depends on the original correlation.

2.7.5 User-supplied correlation matrix

If the MLE covariance is inefficient at proposing. Why do this? How? Show code for how to do this, and talk about positive definitness.

2.7.6 Example MCMC

Use a simple model to demonstrate some of these concepts. Especially the workflow and examining convergence properties, but also maybe restarting.

2.8 Hybrid

The "hybrid" option in ADMB is an implementation of an MCMC algorithm based on Hamiltonian dynamics[2]. It is different from the MH algorithm in how it proposes new values. Instead of proposing random states based on the current value, the hybrid method uses derivatives to follow a contour of the posterior surface. By doing so, it (in theory) only proposes states that are very likely to be accepted, and as such will have less autocorrelation.

In practice, a well tuned hybrid chain will need less thinning, if any at all. The downside of the algorithm is that it is more difficult to tune than the MH algorithm.

2.8.1 Algorithm

The algorithm utilizes the properties of a physical system known as Hamiltonian dynamics. Hamiltonian dynamics, while based in physics, provides some extremely useful properties for Bayesian integration.

A Hamiltonian system consists of two parameter vectors of equal length: "position" (qq) and "momentum" (pp). How these parameters change over time is described by the Hamiltonian function, $H(qq, pp)$. This system can be conceptualized as a frictionless surface about which an object moves. At some time t an object has a certain height (position) and momemtum. The height of the surface is equal to the objective

[2]See [1], which provides a nice background to the algorithm

function of our model, and the momentum variables are introduced nuisance parameters. Samples from our posterior are generated by simulating the movement of the object moving about the surface, governed by H.

The fundamental equations of motion are

$$\frac{dqq}{dt} = \frac{\partial H}{\partial pp} \tag{2}$$

$$\frac{dpp}{dt} = -\frac{\partial H}{\partial qq} \tag{3}$$

ADMB uses the "leapfrog" method to discretize equations (2). This method is similar to the familiar Newton, but is more reliable. It has the same tuning parameters: the number of steps to take, and the step size.

Each iteration of the algorithm, as implemented in ADMB, has two steps:

1. **Propose new momentum variables.** New momentum values are generated from a Normal distribution based on the estimated covariance matrix, and independent of the current position variables.
2. **Metropolis-Hastings update of all variables.** Given the current state of the system, (q, p), new position variables are generated with the leapfrog algorithm using `hynstep` steps and a step size of `hyeps`. The new state is then updated with a MH step (i.e. accepted or rejected).

The algorithm considerers our parameters of interest as "position" variables, and introduces a new set of "momentum" variables (of equal length). The MCMC thus explores the joint posterior of these two vectors, and we do inference on the position variables, and ignore the momentum posteriors.

2.8.2 Arguments

Only show those different than the MH algorithm.

3 Running analyses in R

How to create top-level functions for running analyses programmatically.

3.1 Code

References

[1] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, 2011.