CHAPTER

3

WinBUGS

OUTLINE

3.1 What Is WinBUGS? 29

3.2 Running WinBUGS from R 30

3.3 WinBUGS Frees the Modeler in You 30

3.4 Some Technicalities and Conventions 31

3.1 WHAT IS WinBUGS?

The BUGS language and program was developed by epidemiologists in Cambridge, UK in the 1990s (Gilks et al., 1994; Lunn et al., 2009). The acronym stands for **B**ayesian analysis **U**sing **G**ibbs **S**ampling. In later years, a Windows version called WinBUGS was developed (Spiegelhalter et al., 2003). Despite imperfections, (Win)BUGS is a groundbreaking program; for the first time, it has made really flexible and powerful Bayesian statistical modeling available to a large range of users, especially for users who lack the experience in statistics and computing to fit such fully custom models by maximizing their likelihood in a frequentist mode of inference. (Although no doubt some statisticians may deplore this because it may also lead to misuse; Lunn et al., 2009.)

WinBUGS lets one specify almost arbitrarily complex statistical models using a fairly simple model definition language that describes the stochastic and deterministic "local" relationships among all observable and unobservable quantities in a fully specified statistical model. These statistical models contain prior distributions for all top-level quantities, i.e., quantities that do not depend on other quantities. From this, WinBUGS determines the so-called full conditional distributions and then constructs a Gibbs or other MCMC sampler and uses it to produce the desired number of random samples from the joint posterior distribution.

Introduction to WinBUGS for Ecologists
DOI: 10.1016/B978-0-12-378605-0.00003-X

To let an ecologist really grasp what WinBUGS has brought us, it is instructive to compare the code (say in program R) required to find the maximum likelihood estimates (MLEs) of a custom model using numerical optimization with the code to specify the same model in WinBUGS: the difference is dramatic! With the former, some ugly likelihood expression appears at some point. In contrast, in WinBUGS, the likelihood of a model is specified implicitly as a series of deterministic and stochastic relationships; the latter types of relationships are specified as the statistical distributions assumed for all random quantities in the model.

WinBUGS is a fairly slow program; for large problems, it may fail to provide posterior samples of reasonable size within practical time limits. Custom-written samplers in more general programming languages such as Fortran or C++, or even R, can easily beat WinBUGS in terms of speed (Brooks, 2003), and often do so by a large margin. However, for many ecologists, writing their own samplers is simply not an option, and this makes WinBUGS so unique.

3.2 RUNNING WinBUGS FROM R

In contrast to most other WinBUGS introductions (e.g., McCarthy, 2007; Ntzoufras, 2009), all examples in this book will be analyzed with WinBUGS run from within program R by use of the R2WinBUGS package (Sturtz et al., 2005). R has become the *lingua franca* of statistical computing and conducting a Bayesian analysis in WinBUGS directly from within one's normal computing environment is a great advantage. In addition, the steps before and after an analysis in WinBUGS are greatly facilitated in R, e.g., data preparation as well as computations on the Markov chain Monte Carlo (MCMC) output and the presentation of results in graphs and tables.

Importantly, after conducting an analysis in WinBUGS, R2WinBUGS will import back into R the results of the Bayesian analysis, which essentially consist of the Markov chains for each monitored parameter. Hence, these results must be saved before exiting the program, otherwise all results will be lost! (Although the coda files are saved into the working directory, see 5.3.) This would not be dramatic for most examples in this book, but can be very annoying if you have just run a model for 7 days.

3.3 WinBUGS FREES THE MODELER IN YOU

Fitting statistical models in WinBUGS opens up a new world of modeling freedom to many ecologists. In my experience, writing WinBUGS code, or understanding and tailoring to one's own needs WinBUGS code written by others, is much more within the reach of typical ecologists

than writing or adapting a similar analysis in some software that explicitly maximizes a likelihood function for the same problem. The simple pseudo-code model specification in WinBUGS is just wonderful. Thus, in theory, and often also in practice, WinBUGS really frees your creativity as a modeler and allows you to fit realistically complex models to observations from your study system.

However, it has been said (by Box or Cox, I believe) that statistical modeling is as much an art as a science. This applies particularly to modeling in WinBUGS, where for more complex problems, a lot of art (and patience!) may often be required to get an analysis running. WinBUGS is great when it works, but on the other hand, there are many things that may go wrong (e.g., traps, nonconvergence) without any obvious error in one's programming. For instance, cryptic error messages such as "trap 66" can make one feel really miserable. Of course, as usual, experience helps a great deal, so in an appendix on the book Web site I provide a list of tips that hopefully allow you to love WinBUGS more unconditionally (see *Appendix—A list of WinBUGS tricks*). I would suggest you skim over them now and then refresh your memory from time to time later.

3.4 SOME TECHNICALITIES AND CONVENTIONS

As a typographical convention in this book, WinBUGS and R code is shown in Courier font, `like this`. When part of the output was removed for the sake of clarity, I denote this by `[ ... ]`. When starting a new R session, it is useful or even obligate to set a few options, e.g., choose a working directory (where WinBUGS will save the files she creates, such as those containing the Markov chain values), load the R2WinBUGS package and tell R where the WinBUGS program is located. Each step may have to be adapted to your particular case. You can set the R working directory by issuing a command such as `setwd("C:\")`. The R2WinBUGS function `bugs()`, which we use all the time to call WinBUGS from within R, has an argument called `bugs.directory` that defaults to `"C:/Program Files/WinBUGS14/"` (though this is not the case under Windows VISTA). This is fine for most Anglosaxon computers but will have to be adapted for other language locales. Most recent versions of the `bugs()` function have another option called `working.directory` that partly overrides the global R setting from `setwd()`. It is best set to set `working.directory = getwd()`.

To conduct the analyses in this book, you will need to load some R packages, most notably R2WinBUGS, but sometimes also lme4, MASS, or lattice. At the beginning of every session, execute `library("R2WinBUGS")` and do the same for other required packages. One

useful R feature is a simple text file called `Rsite.profile`, which sits in `C:\Program Files\R\R-2.8.1\etc`., and contains global R settings. For instance, you could add the following lines:

```
library("R2WinBUGS")
library("lme4")
bd <- "C:/Program files/WinBUGS14/"
```

This causes R to load the packages R2WinBUGS and lme4 whenever it is started and to have an object called `bd` that contains the usual WinBUGS address. We could add the option `bugs.directory = bd` whenever calling the `bugs()` function, and when running an analysis on a German-language locale, redefine `bd` appropriately.

In all analyses, I have striven for a consistent presentation. First, we write into the R working directory a text file containing the WinBUGS model description. After that, the other components required for the analysis are written into R objects: data, initial values, a list of parameters to be estimated and MCMC settings. Finally, these objects are sent to WinBUGS by the R2WinBUGS workhorse function `bugs()`. Within the WinBUGS model description, I have also sought consistency of layout by grouping as far as possible statements defining priors, the likelihood and derived quantities. This is not required; indeed, within some limits (e.g., putting code inside or outside of loops or pairs of braces), WinBUGS code lines may be swapped freely. This is not commonly the case with other programming languages.

When conducting a WinBUGS analysis from R, the WinBUGS model description, i.e., the text file that contains the model definition in the BUGS language, could be written in any text editor and then saved into the R working directory. However, I find it useful to use for that the `sink()` function and keep both WinBUGS and R code in the same file. However, you must not get confused about which code is R and which is WinBUGS: put simply, everything inside of the pair of `sink()` calls and inside the paired curly braces after the key word `model` will be WinBUGS code. The book Web site contains a text file with all the code shown in this book, and copying and pasting the code into an open R window works well. However, when using the popular R editor Tinn-R, a correct WinBUGS model description file will not always be produced when using `sink()`, so the WinBUGS trick list on the book Web site shows an alternative that works when using Tinn-R.

All analyses have been tested out with R 2.8.1 and WinBUGS 1.4.3. I would hope that some backwards compatibility is present. Also, most or all of the WinBUGS code should work fine in OpenBugs, but I have not verified this.