APPENDIX

A List of WinBUGS Tricks

I provide here a list of tips that hopefully allow you to love WinBUGS more unconditionally. I would suggest you skim over the list now and then refer back to it later as necessary.

1. *Do read the manual*: WinBUGS may not have the best documentation available for a software, but its manual is nevertheless very useful. Be sure to at least skim over most of it once when you start getting into WinBUGS (i.e., now for many of you), so you may remember that the manual has something to say about a particular topic when you need it. Don't forget the sections entitled "Tricks: Advanced Use of the BUGS Language" and "Tips and Troubleshooting."
2. *How to begin*: When starting a new analysis, always start from a template of a similar analysis. Only ever try to write an analysis from scratch if you want to test yourself.
3. *Initial values 1*: The wise choice of inits can be the key to success or failure of an analysis in WinBUGS, although we don't see this so much in the fairly simple models in this book (the nonstandard generalized linear mixed models in Chapters 20 and 21 are an exception). With more complex models, WinBUGS needs to start the Markov chains not too far away from their stationary distribution or it will crash or not even start to update. Of course, the requirement to start the chains close to the solution goes counter the requirement to start them at dispersed places to assess convergence, so some reasonable intermediate choice is important.
4. *Initial values 2*: Inits must not be outside of the possible range of a parameter. For instance, negative inits for a parameter that has prior mass only for positive values (such as a variance) will cause WinBUGS to crash and so do inits outside of the range of a uniform prior.
5. *Be ignorant, but not too ignorant*: When you want your Bayesian inference to be dominated by your data and choose priors intended to be vague, don't specify too much ignorance, otherwise traps may result or convergence may not be achieved. For instance, don't specify the limits of a uniform prior or the variance of a normal prior to be too wide.

Introduction to WinBUGS for Ecologists
DOI: 10.1016/B978-0-12-378605-0.00027-2

6. *Missing values (NAs)*: NAs are not an issue in this book, since they do not occur in our "perfect" data sets. However, in your real-world data sets, there will always be NAs. In WinBUGS, NAs are dealt with less automatically than in conventional stats programs with which you are likely familiar; hence, it is important to know how to deal with them: briefly, missing responses (i.e., missing y s) are no problem, but NAs in the explanatory variables (the x s) need attention. A missing response is simply estimated, and indeed, adding missing responses for selected covariate values is one of the simplest ways to form predictions for desired values of explanatory variables. On the other hand, a missing explanatory variable must either be replaced with some number, e.g., the mean observed value for that variable, or else given some prior distribution. In general, the former is easier and should not pose a problem unless the number of missing x s is large.
7. *Think in a box (and know your box)*: When coding an analysis in WinBUGS, you often have to deal with data that come in arrays that have more than one dimension. For instance, when analysing animal counts from different sites, over several years, and taken at various months in a year, it may be useful to format them into a three-dimensional array. Some covariates of such an analysis will then have two or even three dimensions, too. Obviously, you must then be absolutely clear about the dimensions of theses "boxes" in which your data are and not get confused by the indexing of the data. In my experience, knowing how to format data into such arrays and then not getting lost is one of the most difficult things to learn about the routine use of WinBUGS.
8. *Know your box (2)*: In "serious" analyses, your modeling often requires the data to be formatted in some multidimensional array. For instance, for a multispecies version of a site-occupancy model (see Chapter 20), you have at least three dimensions corresponding to species (i), site (j), and replicate survey (k). It appears that how you build your array and, especially, how you loop over that array in the definition of the likelihood can make a huge difference in terms of the speed with which your Markov chains in WinBUGS evolve. It appears that you should loop over the longest dimension first and over the shortest last. For instance, if you have data from 450 sites, 100 species, and for two surveys each, then it appears best to format the data as $y[j, i, k]$ and then loop over sites (j) first, then over species (i), and finally over replicate surveys (k).
9. *Don't define things twice*: Every parameter in WinBUGS can only be defined once. For instance, writing `y ~ dnorm(mu, tau)` and then adding `y[3] <- 5` will cause an error. There is a single exception to this rule, and that is the transformation of the response by some

function such as the `log()` or `abs()`. So to conduct an analysis of a log-transformed response, you may write `y <- log(y)` and then `y ~ dnorm(mu, tau)`. Beware of inadvertently defining quantities multiple times when erroneously putting them within a loop that they don't belong.

10. *logit function*: In more complex models, I have fairly often experienced problems when using WinBUGS' own logit function, for instance, with achieving convergence (Actually, problems may arise even with fairly simple models.). Therefore, it is often better to specify that transformation explicitly by `logit.p[i] <- log(p[i] / (1 – p[i]))`, `p[i] <- exp(logit.p[i]) / (1 + exp(logit.p[i]))` or `p[i] <- 1 / (1 + exp(-logit.p[i]))`.
11. *Stabilized logit*: To avoid numerical overflow or underflow, you may "stabilize" the logit function by excluding extreme values (B. Wintle, pers. comm.). Here's a sketch of how to do that. The Gibbs sampling will typically get slower, but at least WinBUGS will be less likely to crash:

```
logit(psi.lim[i]) <- lpsi.lim[i]
lpsi.lim[i] <- min(999, max(-999, lpsi[i]))
lpsi[i] <- alpha.occ + beta.occ * something[i]
```

12. *Truncated normals*: Similarly, in log- or logit-normal mixtures (which we see when introducing a normal random effect into the linear predictor of a Poisson or binomial generalized linear model), you may want to truncate the zero-mean normal distribution, e.g., at ±20 (Kéry and Royle, 2009).
13. *Tinn-R special*: Tinn-R is a popular R editor. Users of Tinn-R 2.0 (or newer) may have problems writing the text file containing the BUGS model description with the `sink()` function; Tinn-R adds to that file some gibberish that will cause WinBUGS to crash. You must then use an alternative way of writing the model file. As an example, here is a workaround for the BUGS description of the model of the mean in chapter 5 that should be compatible with Tinn-R (thanks to Wesley Hochachka):

```
modelFilename = 'model.txt'
cat("
model of the mean {

# Priors
 population.mean ~ dunif(0,5000)
 precision <- 1 / population.variance
 population.variance <- population.sd * population.sd
 population.sd ~ dunif(0,100)
```

```
# Likelihood
 for(i in 1:nobs){
    mass[i] ~ dnorm(population.mean, precision)
 }
}
",fill=TRUE, file=modelFilename)
```

An alternative solution is given by Jérôme Guélat: The "R send" functions available in Tinn-R allow sending commands into R. However the "(echo=TRUE)" versions of these functions should not be used when sending the `sink()` function and its content into R. For example: one should use "R send: selection" instead of "R send: selection (echo=TRUE)".

14. *Trial runs first*: Run very short chains first, e.g., of length 12 with a burnin of 2, just to confirm that there are no coding or other errors. Only when you are satisfied that your code works and your model does what it should, increase the chain length to get a production run.
15. *Use of native WinBUGS*: A key feature of this book is that we run WinBUGS exclusively from within program R. I believe that this is much more efficient than running native WinBUGS. However, with some complex models and/or large sets, WinBUGS will be extremely slow. This may be the one exception where it is perhaps more efficient to run WinBUGS natively. You may still prepare the analysis in R as shown in this book, but only request WinBUGS to run very short Markov chains. When you set the option DEBUG = TRUE in the function `bugs()`, then WinBUGS stays open after the requested number of iterations have been conducted. Then, you can request more iterations to be executed directly in WinBUGS (i.e., using the Update Tool; see Chapter 4). You can then incrementally increase the total chain length and monitor convergence as you go. Once convergence has been achieved, do the required additional number of iterations and save them into coda files. You must do this later, since when exiting WinBUGS, the `bugs()` function only imports back into R the (small) number of iterations that you originally requested. When you have your valuable samples of your complex model's posterior distribution in coda files, use facilities provided by R packages boa or coda to import them into R and process them (e.g., compute Brooks–Gelman–Rubin convergence tests or posterior summaries for inference about the parameters).
16. *Be flexible in your modeling*: Try out different priors, e.g., for parameters representing probabilities try a uniform(0,1), a flat normal for the logit transform, or a Beta(1,1). Sometimes, and for no obvious reason, one may work while another doesn't. Similarly, WinBUGS is very sensitive to changes in the parameterization of a model. Sometimes, one way of

writing the model may work and the other doesn't, for unknown reasons, or one works much faster than the other (which, usually, is mostly an issue for more complex models than most of those featured in this book).

17. *Scale continuous covariates*: Scaling continuous covariates, so that their range does not extend too far away on either side of zero, can greatly improve mixing of the chains and often only make convergence possible (see, e.g., Section 11.4. in this book).
18. *WinBUGS hangs after compiling*: Try a restart, and if that doesn't work, find better starting values (this is an important tip from the manual).
19. *Debugging a WinBUGS analysis 1*: If something went wrong, you need to attentively read through the entire WinBUGS log file from the top to identify the first thing that went wrong. Other errors may follow, but they may not be the actual cause of the failure.
20. *Debugging a WinBUGS analysis 2*: When something doesn't work, the simplest and best advice (see also Gelman and Hill, 2007) is to go back to a simpler version of the same model, or to a similar model, that did work, and then incrementally increase the complexity of that model until you arrive at the desired model. That is, from less complex models *sneak up* on the model you want. Indeed, when using WinBUGS, you learn to always start from the simplest version of a problem and gradually build in more complexity until you are at the level of complexity that you require.
21. *DIC problems*: Surprisingly, sometimes when getting a trap (including one with the very informative title "NIL dereference (read)"), setting the argument `DIC = FALSE` in the `bugs()` function has helped.
22. *R2WinBUGS chokes*: Sometimes the R object created by R2WinBUGS is too big for one's computer. Then, use BOA or CODA to read in the coda files directly, and use their facilities to produce your inference in this way (e.g., convergence diagnostics and posterior summaries).
23. *Identifiability/estimability of parameters*: To see whether two or more parameters are difficult to estimate separately, you can plot the values of their Markov chains against each other.
24. *Check of model adequacy*: Do residual analysis, posterior predictive checks, and cross-validation to see whether your model appears to be an adequate representation of the main features in the data.
25. *Predictions*: They are a very important part of inference: i.e., the estimation of unobserved or future data. One particularly useful way to examine predictions is to estimate what a response would look like for a chosen combination of values of the explanatory variables. The generation and examination of such predicted values is an important

method to understand complex models (for instance, to see what a particular interaction means) and also needed to illustrate the results of an analysis, e.g., as a figure in a paper.

26. *Sensitivity analysis for priors*: Consider assessing prior sensitivity, i.e., repeat your analyses, or those for key models, with different prior specifications and see whether your inference is robust in this respect. If it is not, then not all is lost, but you must report on that in the methods section of your paper.

27. *Have a healthy distrust in your solutions*: Always inspect your inference, e.g., plot predictions against observed values for quantities that can be observed, to make sure that the WinBUGS solution is sensible. Also watch out for unexplained differences in parameter estimates between neighboring models, e.g., those that differ by only one covariate or some other rather minor model characteristic. This can be an indication that something went wrong or that there are estimability problems with the model for your data set.

28. *NAs and NaNs*: When dealing with data in multidimensional arrays, a very useful R package is 'reshape'. The newer versions the reshape package in R 2.9 use an NaN to fill in NAs. This makes WinBUGS very unhappy—you must have NA, not NaN. In general, this is probably good to know about BUGS and newer versions of other packages may be doing the same thing. So, if you use the melt/cast functions in reshape to organize data, then you will need to update your code in the newer R versions by adding "fill=NA_real_". Example: Ymat=cast (data.melt, SppCode~JulianDate~GridCellID, fun.aggregate=mean, fill=NA_real_) (tipp from Beth Gardner).

29. *Long Windows addresses*: WinBUGS doesn't like too long Windows addresses (C:\My harddisk\Important stuff\Less important stuff\ …) for its working directory. Hence, you should not bury your WinBUGS analyses too much down in a tree hierarchy.

30. *VISTA problems*: Windows VISTA has caused all sorts of "challenges" in workshops taught using this book—be prepared! One problem was that the default BUGS directory is not the same as that stated in section 3.4 of the book.