Analyzing Dominance Hierarchies with WinBUGS

Eldridge Adams; Dept. of Ecology & Evolutionary Biology; University of Connecticut;
Last updated: November 2009

This document shows how to use the freeware statistical package WinBUGS to analyze dominance hierarchies following the approach described by Adams (2005). A companion document describes how to use a free supplemental program, CodaReader, to carry out additional steps.

This may appear at first to be a long process, but after you've worked through an example or two, you will be able to carry out the routine steps rapidly. By working through this document and the manual for CodaReader, you will be able to complete all the steps necessary to estimate ranks under the Bradley-Terry model, given data on wins and losses for particular individuals, and to report the degree of certainty for particular inferences. For a broader introduction to Bayesian methods or the use of WinBUGS, several resources are available. If you are new to WinBUGS, I recommend that, after downloading the program, you go through the Tutorial available through its "Help" menu. (The tutorial is part of the User Manual.) New books on Bayesian analysis appear every year, and several of these present WinBUGS code. In particular, McCarthy (2007) presents an introduction to Bayesian analysis for ecologists, with examples implemented in WinBUGS.

Contents:

==========

I. Obtain a copy of WinBUGS

WinBUGS is a free software package (for Microsoft Windows operating systems) that can be used to estimate a wide variety of Bayesian statistical models (Spiegelhalter et al. 2003). Several versions exist, some of which are called OpenBUGS – the open source version. However, the

instructions listed here will work for any of the versions with at most minor differences. OpenBugs can be obtained from the following web site:

http://mathstat.helsinki.fi/openbugs/Home.html

Once you have downloaded, unzipped, and installed the WinBUGS files, you will be ready to carry out the next steps. Notice that once the program is launched, a User Manual is available via the Help menu. This manual includes a tutorial illustrating the use of WinBUGS, and much more.

II. Use WinBUGS to estimate dominance abilities

A. Load the example file, or enter your own data

The sample file "cockroach.odc" -- distributed with this manual -- contains the necessary code for estimating the Bradley-Terry model of dominance, and illustrates the data format. If you want to work with a different data set, you can open and modify this file as described below (section II.A.3), or open a new file and type (or cut and paste) to enter model and your data. The data in the sample file are from Bell and Gorton's (1979) paper on dominance hierarchies in cockroaches, which is discussed by Adams (2005).

Start the program, then select "Open" from the "File" menu. Navigate to the folder containing the file "cockroach.odc," and open the file. The contents are shown below. Comments and labels are shown in boldface: these can be omitted, but including them will not alter the outcome.

The distribution model and priors:

```
model {
   # The Bradley-Terry model
   for (i in 1:dyads) {
      n[i] <- win1[i] + win2[i]
      win1[i] ~ dbin(p[i],n[i])
      logit(p[i]) <- d[ind1[i]] - d[ind2[i]]
   }
   # The next few lines specify the priors
   for (i in 1:individuals) {
      d[i] ~ dnorm(0,tau)
   }
  tau <-  pow(sigma,-2)
  sigma ~ dunif(0,1000)
}
```

...continued on next page

Initial values:

Initial values for the first chain
list(d=c(-14,-14,-14,-14,-14),sigma=0.1)
Initial values for the second chain
list(d=c(14,14,14,14,14),sigma=10)

The data
list(individuals = 5, dyads = 9)

ind1[ ]	ind2[ ]	win1[ ]	win2[ ]
1	2	9	10
1	3	12	2
1	4	6	3
1	5	27	2
2	3	9	5
2	4	12	3
2	5	12	3
3	5	2	0
4	5	2	4

END
*

The cockroach.odc file has three sections:

(a) The distribution model and priors

At the top of the file is a section preceded by the word "model" and contained within a pair of braces { }. This implements the Bradley-Terry model of dominance hierarchies as described by Adams (2005), with one modification. Whereas the code shown in Appendix 1 of Adams (2005) identifies a focal individual, whose dominance score is fixed at 0, here all individuals are treated equally. This avoids the need to evaluate whether the inferences are affected by the assignment of a particular individual as the focal individual. Adams (2005) describes the structure and rationale of the model. The last several lines specify the prior probability distributions for the dominance abilities.

(b) The data

Bell and Gorton's (1979) table of wins and losses is shown below. This table shows, for example, that individuals **a** and **b** met in 19 contests (9 + 10), and that individual **a** prevailed in 9 of these encounters, while individual **b** prevailed in the other 10.

	Loser :				
Winner ↓	**a**	**b**	**c**	**d**	**e**
a	--	9	12	6	27
b	10	--	9	12	12
c	2	5	--	0	2
d	3	3	0	--	2
e	2	3	0	4	--

In the file cockroach.odc, this table has been converted to a rectangular format useable in WinBUGS. At the top is a row identifying four arrays:

ind1[ ] ind2[ ] win1[ ] win2[ ]

Each subsequent row lists data for one pair of individuals. There are four numbers in each row: the index for two different individuals (ind1 and ind2), the number of wins for the first individual (win1), and the number of wins for the second individual (win2). For example, because the individuals in the table above are listed in the order **a**, **b**, **c**, **d**, and **e**, individual **b** is second in the list and individual **d** is fourth in the list, so they are given indices **2** and **4** respectively. For this pair of individuals, the table above shows that individual **b** was the winner in 12 encounters and individual **d** was the winner in 3 encounters. Therefore, the row containing the data for this dyad reads as follows:

2 4 12 3

There is an additional list specifying the number of individuals and the number of dyads for which there are observations:

list(individuals = 5, dyads = 9)

(c) Initial values for two Markov chains

To estimate the values of parameters, WinBUGS uses Markov chain Monte Carlo (MCMC) methods to draw samples from the join posterior probability distribution of all the parameters included in the model. The chains must be given initial starting values. Although WinBUGS can generate these automatically, fewer problems are encountered if the user includes appropriate starting values in the code. For reasons explained below, it is very helpful to run two (or more) chains from different starting values, so initial values are listed in the cockroach file for two chains.

3. Entering other data sets

To enter your own data set, the cockroach.odc file can be modified as follows.

(a) Convert your data on wins and losses to the rectangular format shown in the cockroach.odc example. There should be a row for each unique pair of individuals for which at least one encounter was observed. (In other words, once you have entered a row for the dyad composed of individual 1 and individual 2, do not include a second row with the same two individuals in reverse order.) Give each individual an integer index. The integers should be consecutive starting with 1 (e.g., 1, 2, 3, 4, 5). The row for each pair of individuals must contain four integers in the following order, separated by spaces or tabs:

the index of the first individual
the index of the second individual
the number of encounters in which the first individual won and the second individual lost
the number of encounters in which the second individual won and the first individual lost

Do not include a row for any pair of individuals for which there are no observations (e.g., individuals **c** and **d** in the cockroach example).

As in the example, after the listing for the last pair of individuals, there is an additional line with the word "END." There must be at least one line after that within the odc file or a bug ensues, so its helpful to get in the habit of typing something (e.g., an asterisk) in the line after "END."

(b) Find the following line:

list(individuals = 5, dyads = 9)

Change the two numbers to match your data set. The first number is the number of individuals, which should be equal to the largest index used in step (a). The second number is the number of dyads for which there are observations, which should be equal to the number of rows in your table of data (not including the header and the final word "END"). From the tabulated cockroach data, you can see that there are five individuals and nine unique pairs of individuals for which at least one encounter was observed.

(c) For the lists of initial values, change the number of elements in the list of initial values of d to match the number of individuals in your data set. For the cockroach example, there are five individuals, so there are five elements in the list. If there are 12 individuals in your data set, your lists should have 12 initial values of d, e.g.:

Initial values for the first chain
list(d=c(-14,-14,-14,-14,-14,-14, -14,-14,-14,-14,-14,-14),sigma=0.1)
Initial values for the second chain
list(d=c(14,14,14,14,14,14, 14,14,14,14,14,14),sigma=10)

The initial values can differ from those listed above, but they should contrast sharply between the two chains. It is sometimes necessary to restrict allowable values of d[i] to values greater than around -15 and less than around 15 (see the note on Trap 66 below), so I have chosen initial values towards the limits of this range. Initial values of sigma must be positive numbers. Select "Save As" from the File menu to save your odc file with an appropriate name.

B. Use WinBUGS to obtain parameter estimates

The file containing your model, data, and initial values should still be open.

1. Go to the **Model** menu and select **Specification**. The *Specification Tool* opens. Position this tool conveniently.

2. Double-click on the word "**model**" at the beginning of your model specification. That word, and only that word, will then be highlighted. Click "**check model**" in the *Specification Tool*. (If a dialog box appears with the message "the new model will replace the old one" click "OK" to close the dialog box.) Confirm that the status line (at the lower left of the main WinBUGS window) reads "model is syntactically correct." If not, something has been typed incorrectly.

3. Load the data. There are *two* lists of data to load: the list containing the numbers of individuals and dyads, and the list of wins and losses in rectangular format. The steps are described here for the cockroach example, but are essentially identical for larger data sets.

Find the line reading as follows:

list(individuals = 5, dyads = 9)

Double-click on the word "**list**." When "list" is highlighted, click "**load data**" in the *Specification Tool*.

The status line should now read "data loaded."

Now highlight the entire line "ind1[ ] ind2[ ] win1[ ] win2[ ]". To highlight the line, insert the cursor at the start of the line and, while holding down the left button of the mouse, drag the cursor to the end of the line.

Click "**load data**" in the *Specification Tool* again.

Once again, confirm that the status line reads "data loaded."

4. In the *Specification Tool*, find the box to the right of the label "num of chains." Change this number to 2. Then click "**compile**" in the *Specification Tool*. Confirm that the status line reads "model compiled."

5. Load the initial values. Double-click on the word "**list**" right before the first list of initial values. When the word "list" is highlighted, click "**load inits**" on the *Specification Tool*. Confirm that the status line reads "initial values loaded but this or another chain contain uninitialized variables." In the *Specification Tool*, the number to the right of the label "for chain" has advanced to 2.

Now double-click on the word "**list**" right before the second list of initial values. When the word "list" is highlighted, click "**load inits**" on the *Specification Tool*. Confirm that the status line reads "model is initialized."

6. Go to the **Model** menu and select **Update**. The *Update Tool* opens. Position this tool conveniently.

7. Go to the **Inference** menu and select **Samples**. The *Sample Monitor Tool* opens. Position this tool conveniently.

8. In the node box of the *Sample Monitor Tool*, type "d" (without the quotation marks). As soon as you do so, the "**set**" box in the *Sample Monitor Tool* is activated. Click on "**set**." [Optional: you can then type "sigma" into the node box, and click "set" again.]

9. Type "*" (without the quotation marks) in the node box of the *Sample Monitor Tool*. Several boxes in the *Sample Monitor Tool* are then activated. Click in the "**trace**" box in the *Sample Monitor Tool*.

10. Select the length of the discarded burn-in. The earlier values generated for each Markov chain are often atypical, and should be excluded. The "**beg"** box of the *Sample Monitor Tool* shows the number of the 1st step for which information will be retained. The default value is 1 (= no burn-in). For the vast majority of models, you should type in a greater value to discard the early samples. For this example, change the value in the beg box to 10001, which causes the first 10,000 samples to be omitted. The posterior summary will use only values beginning with the 10,001st. There is no need to change the value in the "end" box for this example.

11. In the "**updates**" box of the *Update Tool*, enter the total number of steps desired for the Markov chain. This is equal to the number of steps in the burn-in plus the number of samples desired from the posterior probability distribution after the burn-in. The greater the number chosen, the more time will be required for computations and the greater will be the precision of the estimates obtained. For this example, a posterior sample of 50,000 is sufficient (and fast), so total number of desired steps is 10,000 + 50,000 = 60,000. Change the value in the "updates" box of the Update Tool to 60000.

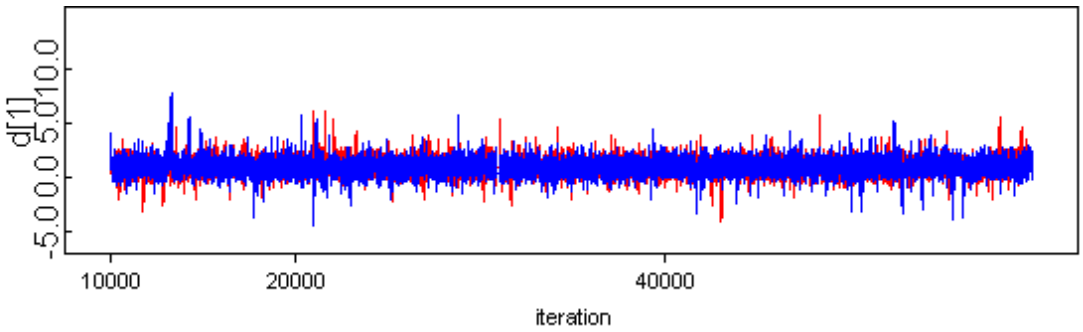
12. Click on "**update**" in the *Update Tool*. The simulation begins! While the simulation is running, the number of iterations completed keeps rising in the "**iteration**" box of the *Update Tool*. When the last sample is drawn, the status line changes to "60000 updates took 37 s" (or however long the simulation ran).

III. Evaluate the output

A. Tools for determining whether the Markov chains have converged. After the simulation is complete, use diagnostic tools to assess whether the two Markov chains (the lists of samples from the posterior distribution) are of sufficient length and are sampling the same distribution. Well behaved chains converge from their initial values to the posterior distribution, and generate enough values to produce a representative sample. The output is not reliable if the chains have remained in the region of their initial values, or if each chain draws samples from a restricted part of the posterior distribution. Comparison of two or more chains starting from different initial values is one of the principal tools for detecting these problems. Follow these steps to generate several diagnostic graphs. These should be examined to determine if anything has gone awry. Some examples of well behaved and problematic output are shown.

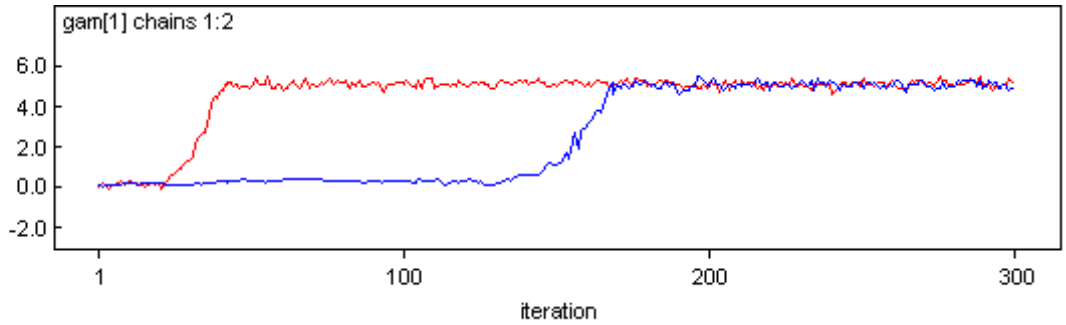
1. Click on "**history**" in the *Sample Monitor Tool* to view the sequence of values generated by the two Markov chains. The values generated by the two Markov chains are shown in different colors. They should appear well mixed – moving up and down through the same vertical range. Also, you should see no indication of a non-stationary distribution; that is, no sign that the mean or variance is changing from early samples to later ones.

The example below (for one of the d[i] values) looks great: the values shown in blue and red appear to come from very similar distributions. The mean and the degree of variation around the mean appear to be the same for both chains, and do not appear to change as the simulation proceeds. There is cause for concern if the values in the blue chain remain above (or below) those of the red chain for long stretches, or if the mean for either chain is clearly different for (say) the first 10,000 steps and the last 10,000 steps.

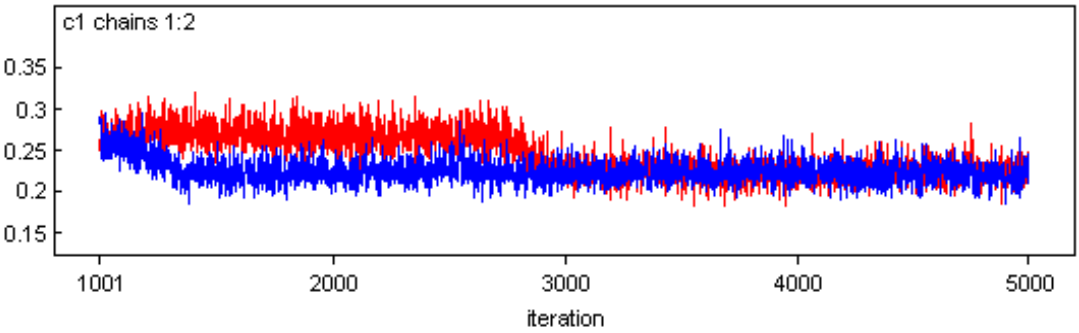


The Bradley-Terry model is well behaved in WinBUGS, so I've had a hard time generating examples in which the chains do not converge. But here are a few examples of output from some other WinBUGS models (following parameters with different names) that illustrate the sort of problems that you should watch out for.

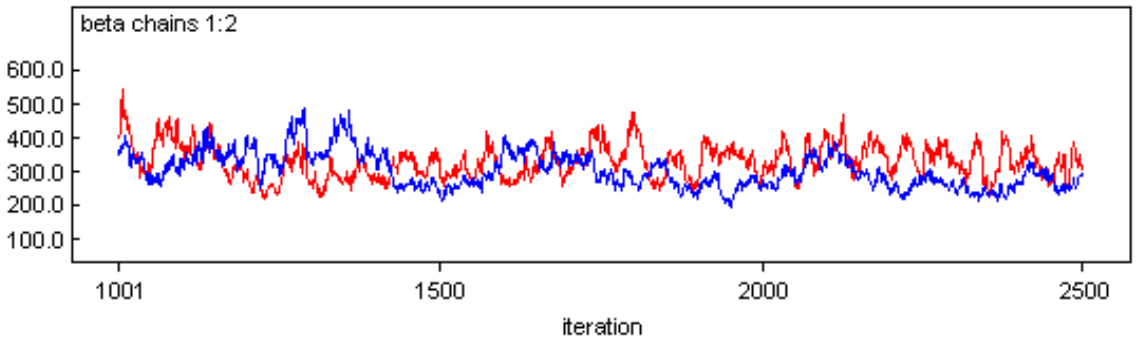
The first troublesome example shows the history of a variable called "gam[1]" for the first 300 iterations of two Markov chains. You can see that both chains stay for awhile at values around 0, before rising to values around 5. The early values should be part of the discarded burn-in. In this case, the discarded burn-in must be *at least* 170; there is little risk in making it much larger.



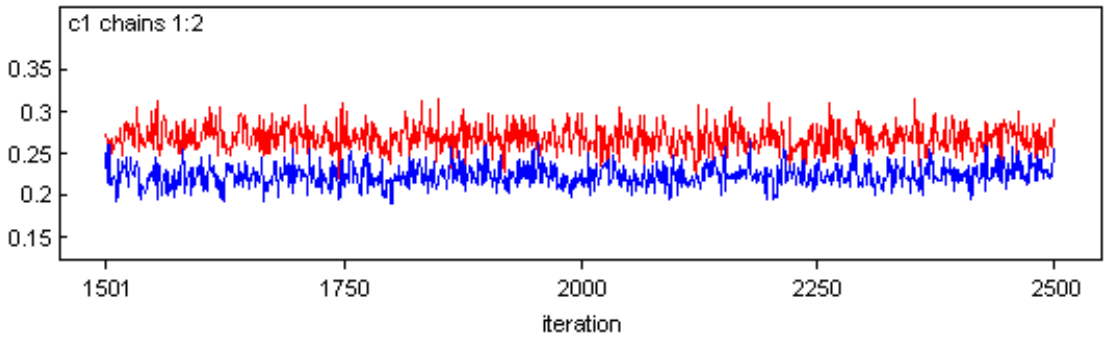
The graph below (from a different model) shows the history for two chains, after a burn-in of 1000. This plot suggests that the chains take awhile to move from their initial values towards the center of the posterior distribution, and therefore that the burn-in must be longer (at least 3000 iterations, but better to make it substantially longer and then examine the history plot again).



The next plot (below) shows the history of a variable for two chains that are mixing, but not as well as in the cockroach example. The red chain stays above or below the blue chain for 100 steps or more. This is unlikely to cause problems as long as the chains are sufficiently long (e.g., tens of thousands of steps instead of the 1500 steps shown below). However, if you see output that looks like this, you should examine the graph of the Gelman-Rubin statistic (described below) to see if there is evidence of non-convergence.

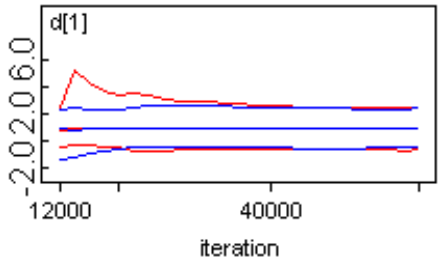


Finally, the two chains in the next graph are not mixing well, to put it mildly. The mean values of the two chains are clearly different; they may be stuck on two different peaks in posterior space. The statistical summaries for this run should not be used as a description of posterior probabilities. You are unlikely to encounter an example that behaves this badly for the Bradley-Terry model of dominance hierarchies.

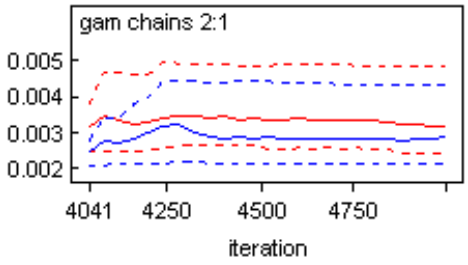


2. Click on the "**quantiles**" button in the *Sample Monitor Tool*. Summaries for the two chains are shown in different colors. For each chain, the central solid line shows the running mean; this line is bracketed by lines showing the running estimates of the 95% credibility interval. We hope to see plots with two desirable properties. First, all of the lines should become nearly horizontal before the last iteration, indicating that the estimates of the mean and credibility interval are no longer changing much as additional samples are gathered. Second, the blue and red lines should stabilize at very similar levels. (In fact they may overlap so much that they cannot be distinguished towards the right side of the plot.) This indicates that the two chains are generating statistically similar samples, which is expected if they have converged to the posterior probability distribution. Lack of convergence is indicated, for example, by running means that remain separated for the two chains.

The example below shows the desired characteristics for one of the d[i] values, but you should examine the quantiles plots for all parameters. Towards the right end of the plot, all of the lines are nearly horizontal, and there is no substantial difference between the two chains (blue and red) for either the mean or the 95% credibility intervals.

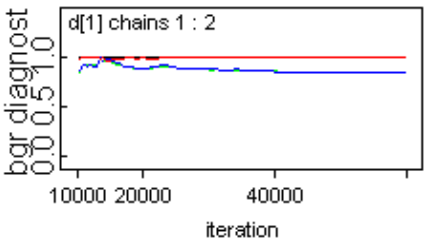


As an example with "red-flags," consider the quantiles plot shown below, which indicates that the two chains are not mixing well. The means and the 95% credibility intervals for these chains are clearly different all the way to the end of the simulation.

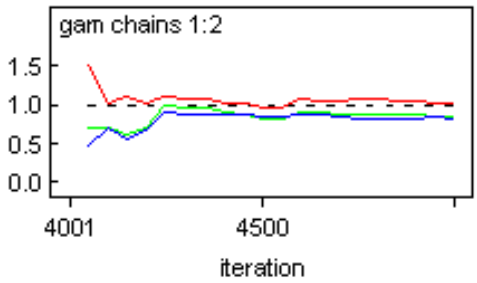


3. Click on the **"brg diag**" button in the *Sample Monitor Tool*. A graph is generated for each monitored parameter. *This may take several minutes.* The red line shows the plot of the Gelman-Rubin convergence statistic against iteration number. This line should become horizontal or very nearly so at a value close to 1.0. Values much greater than 1.0 indicate that the variance for the two chains pooled together is greater than the average within-chain variance, which happens when the chains are drawing dissimilar sets of samples. Gelman (1996) indicates that values less than about 1.1 or 1.2 are acceptable. If the red line does not stabilize to a value that low, then you need to either run the chains for more iterations (easy), or alter the model to improve convergence (more difficult). You should also look at the green and blue lines. These show the width of the central intervals for the pooled and within-chain variances respectively. These lines should stabilize (become horizontal).

In the plot below, the red, blue and green lines all become horizontal, so we're in good shape. (The green line is hard to see because it overlaps with the blue line.)



By contrast, the next graph is troublesome. The Gelman-Rubin statistic (the top line) comes acceptably close to 1.0, but the three plotted values have not stabilized. A much larger number of samples must be generated, or the model should be re-formulated.



If your output passes inspection, advance to section IV.

B. Troubleshooting

1. What to do if the output does not pass inspection

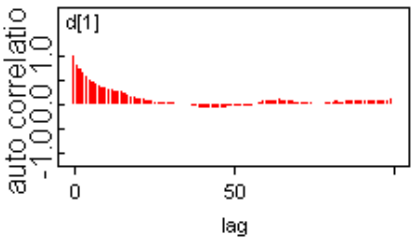
Possible solutions are to:

(a) Increase the length of the burn-in. If the mean of the early samples shown in the history plot differs from the mean of later samples, discard a greater number of early samples by increasing the value in the "**beg**" box of the *Sample Monitor Tool*. (If you increase this

value, you should also increase the value in the "**updates**" box of the *Update Tool* by at least the same amount, or else the size of the posterior sample will be diminished.)

(b) Increase the length of the chains after the burn-in. If the two chains snake up and down slowly on the history plot, if the means of the two chains appears to differ, or if the Gelman-Rubin statistic indicates lack of convergence, you should increase the number of samples substantially, then check again for problems. Enter the additional number of samples in the "**updates**" box of the *Update Tool*, then click the "update" button.

(c) If successive values are autocorrelated, it may be helpful to thin the chains by retaining only 1 out of every 10 values, or one out of every 50 values. There are two ways to detect autocorrelation. First, inspect the history plots. If the graphs drift slowly up and down, then successive values are correlated. Second, in the *Sample Monitor Tool*, click on the "**auto cor**" button. The graph below, for example, shows that estimates of d[1] are autocorrelated out to a lag of about 30 steps. This is acceptable, as long as the chains are of sufficient length, but then the number of values stored in memory may start to strain the computer's resources, and the CODA file (to be generated below) will be unnecessarily long. Therefore, the chains can be thinned by entering "30" (without the quotation marks) in the "**thin**" box of the *Update Tool*.



(d) If the run terminates early with the message "**undefined real result**," there are two possible remedies. The first is to change the initial starting values to numbers that are less extreme (closer to 0). The second is to restrict the possible values of the dominance indices (as described for Trap 66, below).

(e) If you run into other problems, contact me.

2. The infamous "Trap 66"

Some runs may terminate early generating a window with the message "**TRAP 66 (postcondition violated)**," followed by a long list of not-very-helpful information. As far as I can tell, this trap is triggered whenever the algorithm governing the Markov chains is attempting to move to new parameter values which have very low likelihoods and, after some number of attempts, cannot find its way back to a region of greater likelihood. This occurs in dominance hierarchy models when one individual won all of its encounters, or lost all of its encounters. The trap can be avoided by restricting the possible values of the dominance indices, d[i].

This requires only one change. Find the line that reads:

d[i] ~ dnorm(0,tau)

Change it by appending I(-15,15) to the end as follows:

d[i] ~ dnorm(0,tau) I(-15,15)

This restricts the possible values of d[i] so that they may be no smaller than -15 and no larger than 15. (Values other than 15 can be selected, but 15 seems to work well in practice. More extreme values still allow Trap 66, and less extreme values trigger censoring of parameter estimates more often than necessary.)

IV. Generate summaries of the results

A. Estimates of dominance abilities

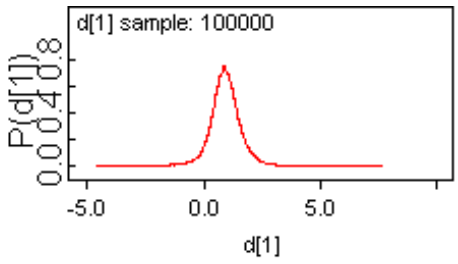
The next items generate graphs or quantitative summaries of the results of the simulation.

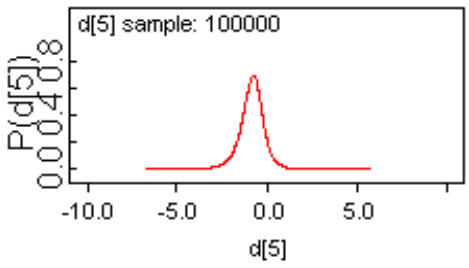
1. Click on "**stats**" in the *Sample Monitor Tool* to see the mean, standard deviation, and median of the posterior probability distributions for each node, as well as the limits of the 95% credibility intervals. "MC error" gives you an estimate of the magnitude of the error resulting from sampling. If you run a longer chain, the magnitude of the MC error will decline. "**start**" indicates the first value used to generate the statistical summary (1 + the length of the burn-in) and "**sample**" indicates how many samples after the burn-in were generated, summed across the chains.

Results for the cockroach analysis:

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
d[1]	0.9304	0.7617	0.01492	-0.5483	0.9093	2.456	10001	100000
d[2]	0.7093	0.7617	0.01493	-0.7803	0.6894	2.233	10001	100000
d[3]	-0.1635	0.7954	0.01482	-1.761	-0.1653	1.392	10001	100000
d[4]	-0.5709	0.7999	0.0146	-2.241	-0.5446	0.9263	10001	100000
d[5]	-0.9099	0.7827	0.01468	-2.546	-0.8779	0.5351	10001	100000

2. Click on "**density**" in the *Sample Monitor Tool* to see a graph of the posterior probability distribution for each variable that you decided to trace. If the variables are continuous (as in this case), the plot is smoothed. Below are the density plots for the first and fifth cockroaches:





The posterior mean of the dominance ability of the first individual is 0.93, and the 95% credibility interval ranges from -0.55 to 0.91. There appears to be substantial overlap between the 95% credibility intervals for any two of these individuals, but examination of these summaries is not a good way to determine the degree of evidence that one individual outranks another. To evaluate the strength of the evidence that, say, individual 1 outranks individual 4, or other hypotheses about dominance ranks, output from WinBUGS can be saved and analyzed using program "CodaReader," written by myself specifically for this purpose.

B. Inferring dominance rank: save a CODA file for further analysis with CodaReader

By themselves, the summaries of dominance abilities (the d[i] values) generated by WinBUGS are of limited usefulness. It is the *differences* in d[i] values that determine the probability of victory, and additional steps are needed to summarize the degree of evidence for particular orderings of dominance abilities. The procedure is to use WinBUGS to save some of the results in a "CODA" file, then to use a second program called CodaReader to carry out the remaining steps. CodaReader is a free, stand-alone application for Windows operating systems. Its use is described in a separate pdf file (the CodaManual.pdf).

To save the file in the right format, you need to re-run the WinBUGS model with two modifications.

1. Run only a single chain. Before you compile the model (that is, after loading the model and the data), change the value for "num of chains" in the *Specification Tool* to 1. Then click the "compile" button. You will then load only one set of initial values—either one will do. Because you are running only one chain, some of the diagnostic tools described above are not available (e.g., the brg diag button, which generates the Gelman-Rubins statistic). Therefore, the single-chain run should be carried out only after you have run two chains and have determined that the chains are of sufficient length.

2. Monitor the variable "d," and nothing else. You may have elected to monitor sigma in earlier runs. For this final run, type in "d" (without the quotation marks) in the "node" box of the *Sample Monitor Tool*, then click the "set" button, then type an asterisk in the "node" box. Carry out the other steps as before.

When the updating has finished, click on the "**coda**" button in the *Sample Monitor Tool*. This will create two windows. The "**CODA index**" window is not needed; it can be closed. The "**CODAchain 1**" window has the information that should be saved. Select "**Save As**" from the "**File**" menu. Navigate to the folder where you want to store the file, give it a name, and select "Plain Text (*.txt)" from the "Save as type:" drop down menu.

See CodaManual.pdf for a description of the next steps.

References

Adams, E.S. 2005. Bayesian analysis of dominance hierarchies. Anim. Behav., 69, 1191-1201.

Bell, W.J. & Gorton, R.E., Jr. 1978. Informational analysis of agonistic behaviour and dominance hierarchy formation in a cockroach, *Nauphoeta cinerea*. Behaviour, 67, 217-235.

Gelman A. 1996. Inference and monitoring convergence. In: *Practical Markov Chain Monte Carlo* (Gilks W., Richardson S. and Spiegelhalter D., eds). New York: Chapman & Hall; 131-143.

McCarthy, M.A. 2007. *Bayesian Methods For Ecologists*. Cambridge, U.K.: Cambridge University Press.

Spiegelhalter, D. J., Best, N. G. & Carlin, B. P. 2002. Bayesian measures of model complexity and fit. J. R. Statist. Soc. B, 64, 1-34.

Spiegelhalter, D., Thomas, A., Best, N. & Lunn, D. 2003. *WinBUGS User Manual*, Version 1.4. Cambridge, U.K.: MRC Biostatistics Unit.