Computer Session: `WinBUGS`

Bayesian model fitting

B.J.T.Morgan and R.S.McCrea

Spring Term 2011

This handout is in three parts: notes on using WinBUGS, and two WinBUGS program listings for the Practical Class, which are available on the lecture course folder. Note that the handout makes extensive use of notes by Steve Brooks, Ruth King and Olivier Gimenez.

1 Introduction

Despite the widespread use of MCMC within the Bayesian statistical community, surprisingly few programs are freely available for its implementation. This is partly because of the fact that algorithms are generally fairly problem-specific (particularly in the case of Metropolis Hastings, for example) and there is no automatic mechanism for choosing the best implementation procedure for any particular problem. However, one program has overcome some of these problems and is widely used by many statistical practitioners. The program is known by the acronym BUGS (Bayesian inference Using Gibbs Sampling). The WinBUGS package is able to implement MCMC methodology for a wide variety of problems, including generalised linear mixed models with hierarchical, crossed, spatial or temporal random effects, latent variable models, frailty models, measurement errors in responses and covariates, censored data, constrained estimation and missing data problems.

The package provides a declarative Splus/R-type language for specifying statistical models. The program then processes the model and data and sets up the sampling distributions required for Gibbs sampling. Finally, appropriate sampling algorithms are implemented to simulate values of the unknown quantities in the model.

Overall the package is extremely easy to use and is capable of implementing the Gibbs sampler algorithm for a wide range of problems. It is also freely available from the WinBUGS website at

`http://www.mrc-bsu.cam.ac.uk/work/bugs/`

which has led to it being an extremely popular tool, widely used amongst practicing statisticians.

2 Getting started with WinBUGs

1. For WinBUGS code, a file(s) is needed that contains:

   i) the model specification;

   ii) data values;

   iii) initial parameter values (note that the initial parameter values can be generated within WinBUGS).

2. Given a WinBUGS code, the first step is to read it into the WinBUGS package:

   i) Click on the `File` button in the tool bar and then `Open` - This will open up a pop-up window, from which one can find a WinBUGS file.

   ii) Find the file, then seither double-click on the file, or single-click on it and then `Open` - This will load open the file into a WinBUGS window.

3. The next step is to read the model into WinBUGS:

   i) Click on the `Model` button on the toolbar and then `Specification` - A `Specification Tool` pop-up menu will appear.

   ii) Highlight the model specification part of the code (one need only highlight the beginning of the model specification), and click `check model` in the `Specification Tool` window - The phrase `model is syntactically correct` should appear in the bottom left of the WinBUGS program. If not, there is an error in the specification of the model, and it will need to be corrected.

   iii) To run multiple chains this needs to be specified here, since the number becomes fixed during the compilation process to come - Note that to

run the Brooks-Gelman-Rubin (BGR) convergence diagnostics at least 3 replications are needed.

4. Now that the model is correct, we need to read in the data:

   i) Highlight the word `list` corresponding to the data from our file and click on `load data` in the `Specification Tool` window - The phrase `data loaded` should appear in the bottom left corner of the WinBUGS window; else an error message will appear.

   ii) Click `compile` in `Specification Tool` - The phrase `model compiled` should now appear in the bottom left corner; else an error message will appear stating why there is a problem with the code.

5. The final input step is to read in the initial parameter values for the Markov chain:

   i) Highlight the word `list` in the file that corresponds to the initial parameter values and then click on `load inits` - The phrase `model is initialized` should appear in the bottom left corner.

   ii) Alternatively, it is also possible to allow WinBUGS to generate starting values for the parameters. To do this click on `gen inits` in the `Specification Tool` - If WinBUGS can generate all the parameter values the phrase `initial values generated` will appear in the bottom left corner. However, note that WinBUGS is not always capable of selecting suitable starting values. An error message will appear in the bottom left corner, specifying the parameters which could not be sampled. In this case repeat the above procedure of reading in the model and supply the initial parameter values. In some cases we may wish to read in some initial parameter values and allow WinBUGS to generate the rest. This can be done by simply following the above procedure to read in the initial parameter values, where the `list` in the file contains only some parameter values. Then following reading in these values, click on `gen inits` to generate the remaining parameter values.

6. Once the model has been compiled and the data read in, we are ready to start.

3 Running WinBUGs code

1. Pull down a few more pop-up menus from the toolbar:

   i) Click on `Update` from the pull-down `Model` menu - The `Update Tool` is used to run the simulations.

   ii) Click on `Samples` from the `Inference` pull-down menu - The `Sample Monitor Tool` window allows the specification of the parameters to be recorded. Note that it is necessary to specify the parameters to be recorded prior to running any simulations.

2. Now to use these to run the simulations and monitor the parameters:

   i) In the `Sample Monitor Tool` enter the names of the parameters individually, pressing `set` after each parameter - Note that for vectors, entering the vector name ensures that all elements of the parameter vector are recorded. For the dippers example above, the parameters to be set are p and phi.

   ii) To run the simulations use the `Update Tool`, inputting the number of iterations to be run (`update`) and how often to refresh the updates to write to the screen (`refresh`) - Press `update` to then run this many iterations. Clicking on `update` whilst the MCMC algorithm is running will pause the simulation. Click on `update` again, and the MCMC simulations will continue for as many iterations again. Whilst the `adapting` box in the `Update Tool` window is checked, the simulation is going through an automatic adaptation procedure and no summary statistics are available during this period. This is usually fairly short and usually less than 5000 iterations.

   iii) Note that it is often useful to initially run very short simulations to check the length of time that they take before running a large number of iterations.

4 Obtaining posterior estimates in WinBUGS

1. Once the simulations are run and the adaptation period is finished, we can begin to monitor the statistics of interest:

i) Highlight the relevant parameter in the `node` box of the `Sample Monitor Tool` window - Click on the arrow to the right of the `node` box to get a scrollable list of the parameters entered in earlier. Note that using the symbol $\star$ in the `node` box represents all specified parameters.

ii) The values for `beg` and `end` specify the start and end of the sample values to be used to construct summary statistics - The `thin` box allows thinning of the output. Setting `thin=10`, for example, means that the sample statistics are based on every 10th sample value. The values for `beg` and `end` specify the start and end of the sample values to be used to construct summary statistics.

2. The `Sample Monitor Tool` allows a variety of sample statistics to be calculated:

   - `trace`: Provides a dynamic trace plot of recent parameter values which updates continuously during the simulation.
   - `history`: Provides a static trace plot of parameter values for the entire simulation.
   - `density`: Produces plots of the posterior marginal densities of the selected parameters.
   - `stats`: Provides a table of posterior means, variances and other summary statistics.
   - `coda`: Dumps out a list of monitored parameter values for input into alternative packages for calculating posterior summary statistics.
   - `quantiles`: Plots out the running mean of the selected parameters and symmetric 95% credible interval.
   - `bgr diag`: Calculates the Brooks, Gelman and Rubin convergence diagnostic if multiple chains have been run. Plots settling to a value close to 1 indicates convergence.
   - `auto cor`: Plots the autocorrelation function for the selected parameters. This measures the between-iteration dependence sampled values.

5 WinBUGS example - dippers

To illustrate the WinBUGS package we fit model C/C to the dippers dataset described earlier in the lecture course. The WinBUGS codes have been annotated for

clarity. This is done by simply using the # sign which acts as a line comment.

The code is read into WinBUGS and compiled as described above. Recall that before we perform any simulations, we need to set the parameters that are to be monitored. This is done via the `Sample Monitor Tool`, where the parameters in this case are the survival rate `phi` and recapture rate `p`. The simulations are initially run for 10000 iterations, using the `Update Tool`. This took approximately 2 seconds. The corresponding trace plot of the full MCMC simulation is given using the `history` button within the `Sample Monitor Tool` and is given in Figure 4. The trace plots suggest that convergence is very rapid, essentially within only a few iterations. However, in order to ensure convergence has been achieved we use a (very) conservative burn-in of 1000 iterations.

Several checks can be made regarding convergence. The autocorrelation plots for the survival and recapture probabilities in Figure 5 shows that the dependence between iterations in the sampled values is negligible. In addition, convergence seems to occur for the two parameters as the values for the Brooks, Gelman and Rubin convergence diagnostic for both the survival and recapture probabilities are less than 1.2. The plots for this diagnostic can be produced in WinBUGS using the `bgr diag` button within the `Sample Monitor Tool`, see Figure **??**. The numerical values can be obtained by selecting the plot followed by a ctrl-left-mouse-click (for further details, see the *The Inference Menu* in the *WinBUGS User Manual* which you can obtain by pressing `F1`).

6 Exercise

In the Practical Class, experiment with changing models, by copying the files to your own file-space, and then suitably moving the #.

```
model {
# Define priors for phi and p (beta distributions)
# The symbol ~ means "distributed as"
phi ~ dbeta(1,1)
p ~ dbeta(1,1)
# Define the likelihood:
    for (i in 1 : ni){
        m[i, 1 : (nj + 1)] ~ dmulti(q[i, ], r[i])
    }
# Calculate the no. of birds released each year (see input data)
    for (i in 1 : ni){
        r[i] <- sum(m[i, ])
    }
# Calculate the cell probabilities
    for (i in 1 : ni){
# Calculate the diagonal elements
        q[i, i] <- p*phi
# Calculate remaining terms above diagonal
        for(j in (i+1) : nj ){
            for(k in i : (j -1)){
                lq[i, j, k] <- log(phi*(1-p))
            }
# Probabilities in table
            q[i,j] <- p*phi *exp(sum(lq[i,j,i:(j-1)]))
        }
# Zero probabilities in lower triangle of table
        for(j in 1 : (i - 1)){
            q[i, j] <- 0
        }
# Probability of an animal not being seen again
        q[i, nj + 1] <- 1 - sum(q[i,1:nj])
    }
}
```

Figure 1: Example WinBUGS code of the model specification for the dippers dataset. Note that `phi` denotes the survival rate, `p` the recapture rate, `ni` the number years of release; `nj` the number of recapture events; `m` the cell entries of the m-array (number of first recaptures).

```
# Data Set: Dippers list(ni=6,nj=6,m=structure(.Data=c(
        11,2,0,0,0,0,9,
        0,24,1,0,0,0,35,
        0,0,34,2,0,0,42,
        0,0,0,45,1,2,32,
        0,0,0,0,51,0,37,
        0,0,0,0,0,52,46
        ), .Dim=c(6,7)))
```

Figure 2: Example WinBUGS code of the data for the dippers dataset. Note that `ni` denotes the number years of release; `nj` the number of recapture events; `m` the cell entries of the m-array with the final column detailing the number of birds that are released in the given year and not observed again.

```
# Initial values list(phi=.5,p=.5)
```

Figure 3: Example WinBUGS code for initial starting values for the dippers dataset, where `phi` denotes the survival rate and `p` the recapture rate.

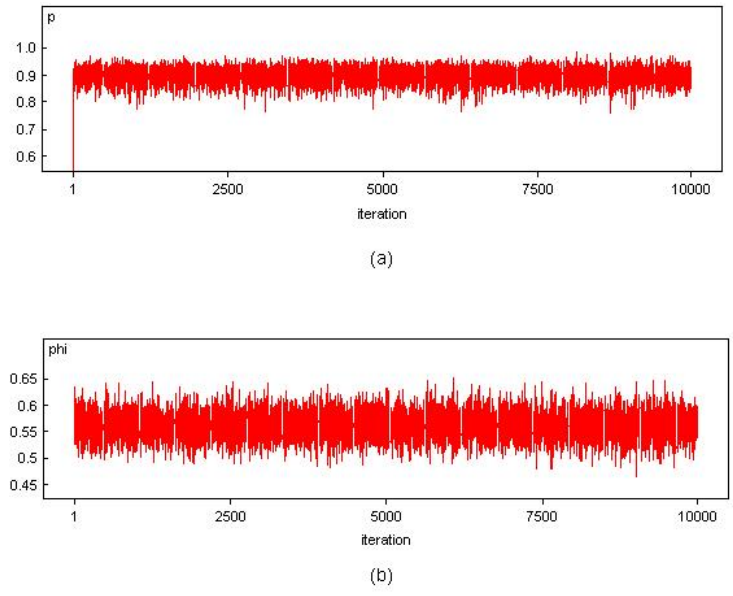

Figure 4: The trace plot of the parameters (a) `p` and (b) `phi` produced in WinBUGS using the `history` button within the `Sample Monitor Tool`.

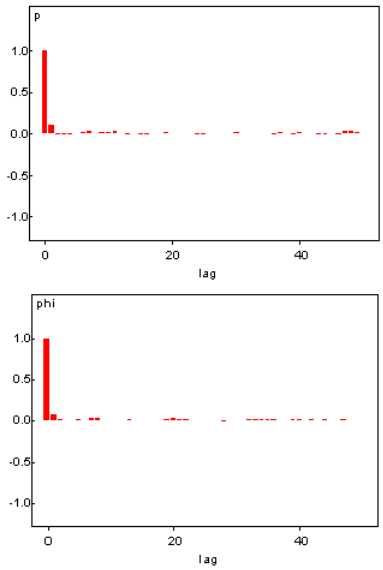

Figure 5: The autocorrelation plot of the parameters `p` (top panel) and `phi` (bottom panel) produced in WinBUGS using the `auto cor` button within the `Sample Monitor Tool`.

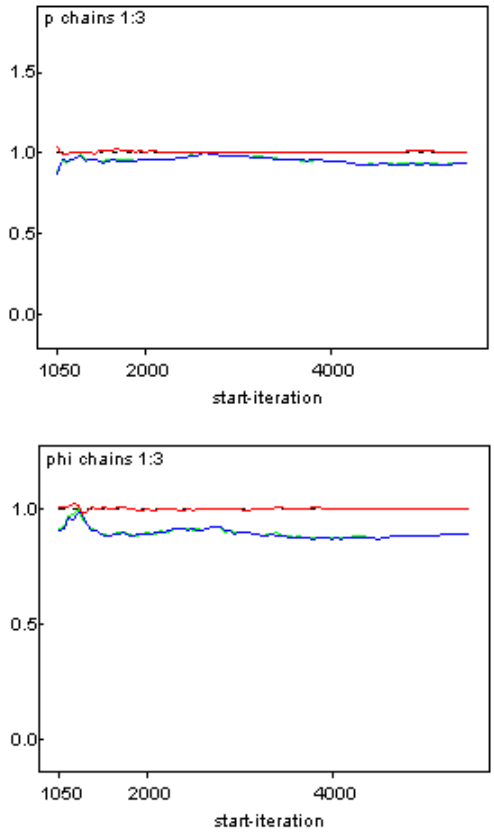

Figure 6: The Brooks, Gelman and Rubin convergence diagnostic plot of the parameters `p` (top panel) and `phi` (bottom panel) produced in WinBUGS using the `bgr diag` button within the `Sample Monitor Tool`.