Hints on using WinBUGS

1 Running a model in WinBUGS

1. Start WinBUGS by double clicking on the WinBUGS icon (or double click on the file `WinBUGS14.exe` in the `WinBUGS14` directory in `C:\Program Files`).

2. Open file containing model code as follows:

   - Point to File on the tool bar and click once with left mouse button (LMB);
   - Highlight Open option and click once with LMB;
   - Select appropriate directory and double click on file to open.

3. Select the Model menu as follows:

   - Point to Model on the tool bar and click once with LMB.
   - Highlight Specification option and click once with LMB.

4. Focus the window containing the model code by clicking the LMB once anywhere in window — the top panel of the window should then become highlighted in blue to indicate that the window is currently in focus.

5. Highlight the word `model` at the beginning of the code by dragging the mouse over the word whilst holding dowm the LMB.

6. Check model syntax by moving the mouse over `check model` box in the Specification Tool window and clicking once with LMB.

   - A message saying `model is syntactically correct` should appear in the bottom left of the WinBUGS program window.

7. Open data (the data can either be stored in a separate file, in which case open this file, or they may be stored in the same file as the model code, in which case click on the arrow as instructed to view the data).

8. Load data as follows:

   - Highlight the word `list` at the beginning of the data file.
   - Click once with LMB on `load data` box in Specification Tool window.
   - A message saying `data loaded` should appear in the bottom left of the WinBUGS program window.

9. If you wish to run more than one chain, then specify the number of chains in the white box labeled `num of chains` in the Specification Tool window (the default is 1 chain).

10. Compile the model by clicking once with the LMB on the `compile` box in the Specification Tool window.

- A message saying `model compiled` should appear in the bottom left of the WinBUGS program window.

11. Open inital values files (the initial values can either be stored in separate file(s), in which case open these files, or they may be stored in the same file as the model code, in which case click on the arrow as instructed to view the initial values).

12. Load initial values as follows:

    - Highlight the word `list` at the beginning of the first set of initial values.
    - Click once with LMB on `load inits` box in Specification Tool window.
    - A message saying `initial values loaded: model itialialized` should now appear in the bottom left of the WinBUGS program window.

    If you have specified more than 1 chain to be run, then you need to load separate initial value files for each chain (just repeat the above steps for each file).

13. Close the Specification Tool window by clicking once with LMB on X button in top right corner of window.

14. You are now ready to start running the simulation:

    - Before doing so, you may want to set some monitors to store the sampled values for selected parameters (see section **Monitoring parameter values** below).
    - To run the simulation, select the Update option from the Model menu.
    - Type the number of updates (iterations of the simulation) you require in the appropriate white box (default is 1000).
    - Click once on the `update` box:
      - The program will now start simulating values for each parameter in the model.
      - This may take a few seconds — the box marked `iteration` will tell you how many updates have currently been completed. The number of times this value is revised depends on the value you have set for `refresh` (see white box above `iteration`). The default is every 100 iterations, but you can ask the program to report more frequently by changing `refresh` to, say, 10 or 1.

15. When the updates are finished, the message `updates took *** s` will appear in the bottom left of the WinBUGS program window (where `***` is the number of seconds taking to complete the simulation).

16. If you set monitors for any parameters you can now view graphical and numerical summaries of the samples (see below).

17. To save any files created during your WinBUGS run, focus the window containing the information you want to save, and select the Save As option from the File menu.

18. To quit WinBUGS, select the Exit option from the File menu.

2 Monitoring parameter values

In order to check convergence and obtain posterior summaries of the model parameters, you first need to set *monitors* for each parameter of interest. This tells WinBUGS to store the values sampled for those parameters; otherwise, WinBUGS automatically discards the simulated values.

There are two types of monitors in WinBUGS:

1. *Sample* monitors

   - Setting a sample monitor tells WinBUGS to store *every* value it simulates for that parameter.
   - You will need to set sample monitors if you want to view trace plots of the samples to check convergence (see section **Checking convergence** below) and if you want to obtain posterior quantiles, for example, the posterior 95% Bayesian credible interval for that parameter.
   - To set a sample monitor:
     - Select Samples from the Inference menu.
     - Type the name of the parameter to be monitored in the white box marked `node`.
     - Click once with the LMB on the box marked `set`
     - Repeat for each parameter to be monitored.

2. *Summary* monitors

   - Setting a summary monitor tells WinBUGS to store the running mean and standard deviation for the parameter.
   - The values saved contain less information than saving each individual sample in the simulation, but require much less storage. This is an important consideration when running long simulations (1000's of iterations) and storing values for many variables.
   - We usually set *summary* monitors on the vector of area-specific random effects (relative risks) since we are primarily interested in obtaining the posterior mean as an estimate of the relative risk of disease in each area (these are the quantities we use for mapping). However, if we want to estimate a 95% interval for each relative risk, then we would need to set a *sample* monitor in order to store the necessary information for each parameter.
   - To set a summary monitor:
     - Select Summary from the Inference menu.
     - Type the name of the parameter to be monitored in the white box marked `node`.
     - Click once with the LMB on the box marked `set`
     - Repeat for each parameter to be monitored.

Note: you should not set a summary monitor until you are happy that convergence has been reached (see next section), since it is not possible to discard any of the pre-convergence ('burn-in') values from the running mean summary once it is set.

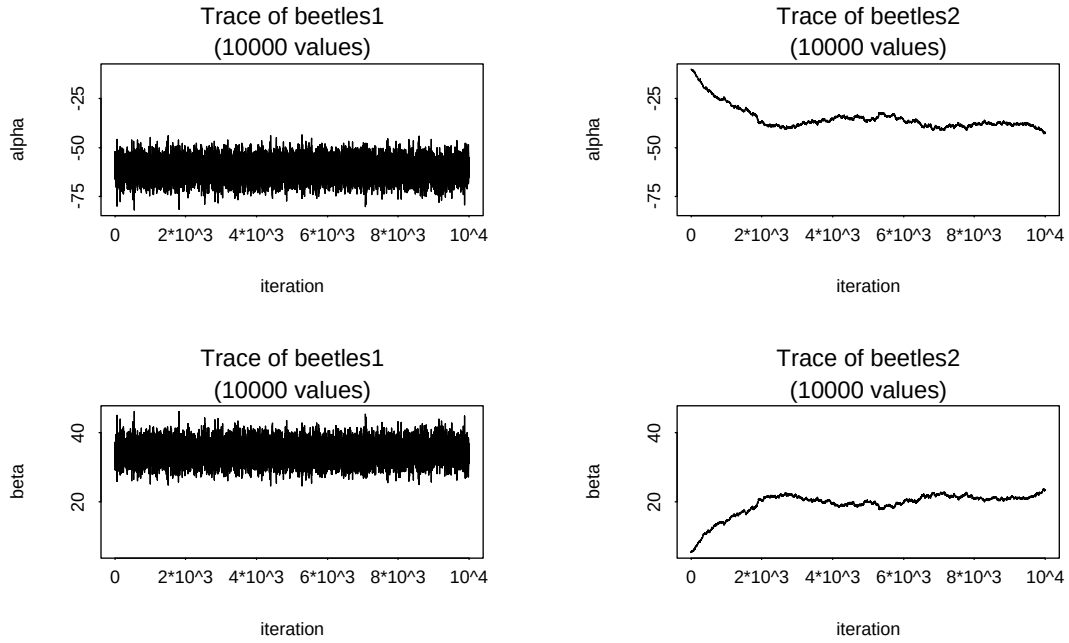
3 Checking convergence

Checking convergence requires considerable care. It is very difficult to say conclusively that a chain (simulation) has converged, only to diagnose when it definitely hasn't converged.

The following are practical guidelines for assessing convergence:

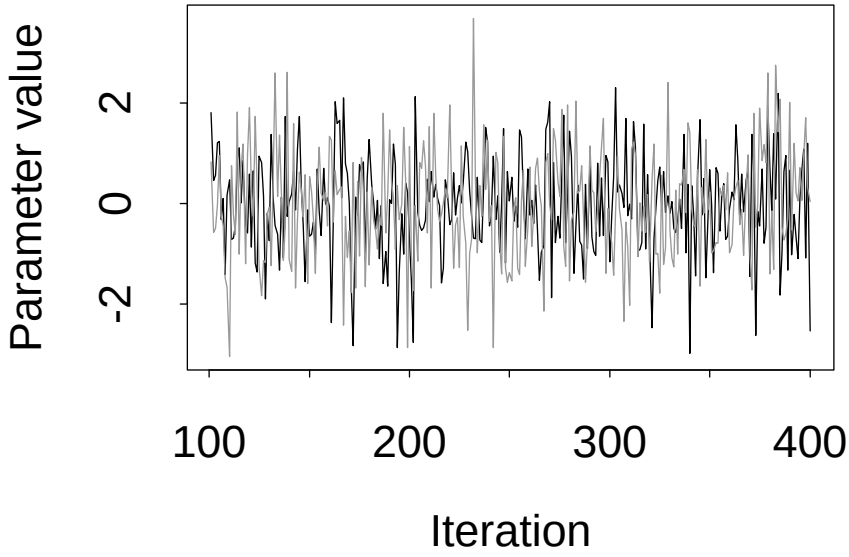
- For models with many parameters, it is inpractical to check convergence for every parameter, so just chose a random selection of relevant parameters to monitor
  - For example, rather than checking convergence for every element of a vector of random effects, just chose a random subset (say, the first 5 or 10).
- Examine trace plots of the sample values versus iteration to look for evidence of when the simulation appears to have stabilised:
  - To obtain 'live' trace plots for a parameter:
    * Select Samples from the Inference menu.
    * Type the name of the parameter in the white box marked `node`.
    * Click once with the LMB on the box marked `trace`: an empty graphics window will appear on screen.
    * Repeat for each parameter required.
    * Once you start running the simulations (using the Update Tool, trace plots for these parameters will appear 'live' in the graphics windows.
  - To obtain a trace plot showing the full history of the samples *for any parameter for which you have previously set a sample monitor and carried out some updates*:
    * Select Samples from the Inference menu.
    * Type the name of the parameter in the white box marked `node` (or select name from pull down list).
    * Click once with the LMB on the box marked `history`: a graphics window showing the sample trace will appear.
    * Repeat for each parameter required.
  - The following plots are examples of (i) chains for which convergence (in the pragmatic sense) looks reasonable (left hand side); (ii) chains which have clearly not reached convergence (right hand side):

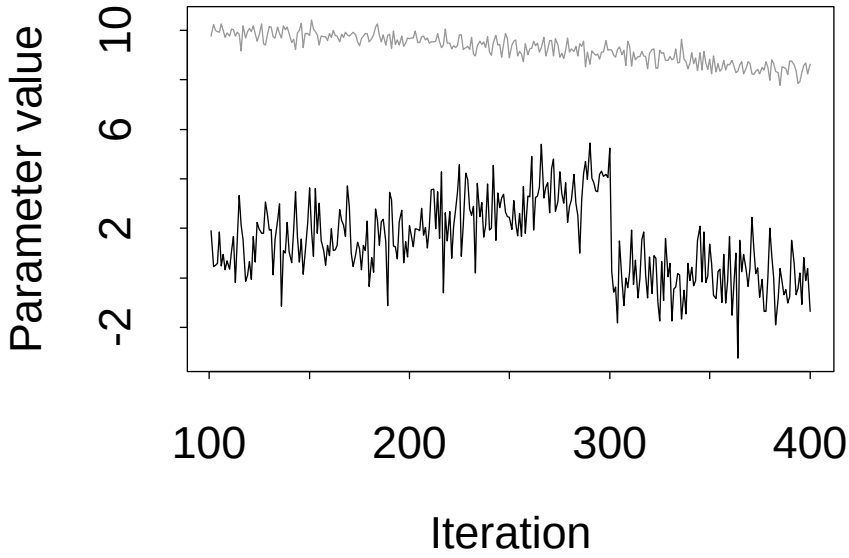
BEETLES



– If you are running more than 1 chain simultaneously, the trace and history plots will show each chain in a different colour.

  * In this case, we can be reasonably confident that convergence has been achieved if all the chains appear to be overlapping one another.
  * The following plots are examples of (i) multiple chains for which convergence looks reasonable (top); (i) multiple chains which have clearly not reached convergence (bottom):


Parameter value
-2
0
2
100
200
300
400
Iteration


Parameter value
-2
2
6
10
100
200
300
400
Iteration

4 How many iterations after convergence?

Once you are happy that convergence has been achieved, you will need to run the simulation for a further number of iterations to obtain samples that can be used for posterior inference. The more samples you save, the more accurate will be your posterior estimates.

One way to assess the accuracy of the posterior estimates is by calculating the Monte Carlo error for each parameter. This is an estimate of the difference between the mean of the sampled values (which we are using as our estimate of the posterior mean for each parameter) and the true posterior mean.

As a rule of thumb, the simulation should be run until the Monte Carlo error for each parameter of interest is less than about 5% of the sample standard deviation. The Monte Carlo error (MC error) and sample standard deviation (SD) are reported in the summary statistics table (see next section).

5 Obtaining summary statistics of the posterior distribution

To obtain summaries of the monitored samples, to be used for posterior inference:

- Select Samples from the Inference menu.
- Type the name of the parameter to be summarised in the white box marked `node` (or select name from the pull down list, or type `*` to select all monitored parameters).
- Type the iteration number which you want to start you summary from in the white box marked `beg`: this allows the pre-convergence 'burn-in' samples to be discarded.
- Click once with LMB on box marked `stats`: a table reporting various summary statistics based on the sampled values of the selected parameter will appear.

6 Plotting summaries of the posterior

`WinBUGS` version 1.4 includes options for producing various plots of posterior summary statistics. The plot options include:

1. **Kernel density plot:** plots an estimate of the shape of the (univariate) marginal posterior distribution of a parameter see on-line manual for further details (select User manual from the **Help** menu in `WinBUGS` and then take a look at the subsection on *Density plots* in the *WinBUGS Graphics* section).

2. **Box plots or caterpillar plots:** these plots show a side-by-side comparison of the posterior distributions of each element of a vector of parameters summarised either as a point estimate and 95% interval (caterpillar plot) or by the mean, interquartile range and 95% interval (box plot). Typically this vector will be a vector of random effects parameters. For example, suppose you have a vector of random effects called `p` in your model:

   - You should have already set a samples monitor on the appropriate vector (`p`) and carried out a suitable number of updates.
   - Then select **Compare** from the Inference menu.
   - Type the name of the parameter vector to be plotted (in this case `p`) in the white box marked `node`.
   - If you want to discard any pre-convergence burn-in samples before plotting, type the appropriate iteration number in the white box marked `beg`.
   - Click once with LMB on the button marked either `box plot` or `caterpillar` as required.

3. **Model fit:** this option produces a 'time series type plot and is suitable for plotting an ordered sequence of parameter estimates against corresponding values of a known variable, e.g. plotting posterior estimates of the fitted values of a growth curve against time. For example, in the rats model from the WinBUGS examples Vol I, you could use this option to produce a plot of the vector of 5 fitted values for the weight of each rat (`mu[i,]`) against age (`x`), as follows:

   - You should have already set a samples monitor on the appropriate vector (i.e. `mu`, the mean of the normal distribution assumed for the responses, `Y`) and carried out an appropriate number of updates.
   - Then select Compare from the **Inference** menu.
   - Type the name of the (stochastic) parameter vector to be plotted on the vertical axis in the white box marked `node` (e.g. `mu[1,]` to produce the plot for rat 1).
   - Type the name of the known (i.e. not stochastic) variable to be plotted on the horizontal axis in the white box marked `axis` (in this case, `x`, the name of the vector of ages at which each rat was measured).
   - An optional argument is to type the name of another known (i.e. not stochastic) variable in the white box marked `other` (for example, `Y[1,]` — this would plot the observed measurements for rat 1 as well as the fitted values `mu[1,]`).
   - If you want to discard any pre-convergence burn-in samples before plotting, type the appropriate iteration number in the white box marked `beg`.
   - Click once with LMB on the button marked model fit. The resulting plot shows the posterior median (solid red line) and posterior 95% interval (dashed blue line) for the values of node (in this case, the fitted values `mu[1,]` for rat

1) against the values of the variable specified in the axis box (in this case, `x`, the age of the rat at each measurement). The black dots show the values of the variable specified in the other box (in this case, `Y[1,]`, the observed weights for rat 1).

4. There are various options for customising all these plots (e.g. changing the order in which the elements of the vector are plotted, switching the x and y axis, etc.). To access these options, click on the window containing the plot to focus it, then place the mouse somewhere in the plot window and click once with the *right* mouse button. A menu will appear and you should select the Properties option. This will open another menu called Plot Properties which provides options for editing plot margins, axis labels and fonts etc. (these are generic options for all `WinBUGS` plots), plus some special options specific only to certain plots (click on the `Special` button at the bottom of the Plot Properties menu). See the on-line manual for further details (select User manual from the Help menu in `WinBUGS` and then take a look at the sections *Inference* (see sub-section on the *Compare* menu) and *WinBUGS Graphics*).

7 Some notes on the BUGS language

7.1 Basic syntax

- `<-` represents logical dependence, *e.g.* `m <- a + b*x`
- `~` represents stochastic dependence, *e.g.* `r ~ dunif(a,b)`
- Can use arrays and loops

```
for (i in 1:n){
  r[i] ~ dbin(p[i],n[i])
  p[i] ~ dunif(0,1)
  }
```

- Some functions can appear on left-hand-side of an expression, *e.g.*

```
logit(p[i])<- a + b*x[i] log(m[i])  <- c + d*y[i]
```

- `mean(p[])` to take mean of whole array, `mean(p[m:n])` to take mean of elements m to n. Also for `sum(p[])`.
- `dnorm(0,1)I(0,)` means the prior will be restricted to the range $(0, \infty)$.

7.2 Functions in the BUGS language

- `p <- step(x-.7)` = 1 if `x` ≥ 0.7, 0 otherwise. Hence monitoring `p` and recording its mean will give the probability that `x`≥ 0.7.
- `p <- equals(x,.7)` =1 if `x` = 0.7, 0 otherwise.
- `tau <- 1/pow(s,2)` sets $\tau = 1/s^2$.
- `s <- 1/ sqrt(tau)` sets $s = 1/\sqrt{\tau}$.
- `p[i,k] <- inprod(pi[], Lambda[i,,k])` sets $p_{ik} = \sum_j \pi_j \Lambda_{ijk}$.
- See 'Model Specification/Logical nodes' in manual for full syntax.

7.3 Some common Distributions

Expression	Distribution	Usage
`dbin`	binomial	`r ~ dbin(p,n)`
`dnorm`	normal	`x ~ dnorm(mu,tau)`
`dpois`	Poisson	`r ~ dpois(lambda)`
`dunif`	uniform	`x ~ dunif(a,b)`
`dgamma`	gamma	`x ~ dgamma(a,b)`

NB. The normal is parameterised in terms of its mean and *precision* = 1/ variance = $1/\text{sd}^2$.

See 'Model Specification/The BUGS language: stochastic nodes/Distributions' in manual for full syntax.

Functions cannot be used as arguments in distributions (you need to create new nodes).

7.4 The WinBUGS data formats

WinBUGS accepts data files in:

1. Rectangular format

```
 n[] r[]
 47  0
148 18
 ...
360 24
END
```

2. S-Plus format:

```
list(N=12,n = c(47,148,119,810,211,196,
          148,215,207,97,256,360),
     r = c(0,18,8,46,8,13,9,31,14,8,29,24))
```

Generally need a 'list' to data consist of mixture of scalars and vectors/arrays, or vectors/arrays of different lengths.

7.5 Calling WinBUGS from other software

- Scripts enable WinBUGS 1.4 to be called from other software
- Interfaces developed for R, Splus, SAS, Matlab
- See `www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml`
- The `R2winbugs` function for R is most developed - reads in data, writes script, monitors output etc. See `http://cran.r-project.org/src/contrib/Descriptions/R2WinBUGS.html`
- OpenBUGS site `http://mathstat.helsinki.fi/openbugs/` provides an open source version, including BRugs which works from within R