Introduction to Bayesian Inference: Practical Exercises

You will be using WinBUGS 1.4.3 for these practicals

All the data and other files you will need for the practicals are provided in a zip file, which you can download from
`http://www.bias-project.org.uk/WB2011Man/CourseData/WBintroData.zip`.
You should unzip this file and save the contents in the directory `C:\work\bugscourse`**.**

Solutions for all the exercises are also provided in the zip files.

Scripts have also been prepared which can be used to run most of the models as an alternative to using the menu click-and-point interface. You are advised to start by using the menu interface for the first practical, so that you understand the steps involved in setting up and running a model in WinBUGS. However, feel free to use the scripts to run models for the other practicals (or write your own scripts to do this) if you prefer. **If you have copied the data and program files to a directory other than** `C:\work\bugscourse` **you will need to edit all the path names of the files specified in each script.**

Further detailed instructions on using `WinBUGS` **can be found in the handout** *Hints on using WinBUGS*, **and in the on-line WinBUGS User Manual (see the Help menu in** `WinBUGS`**).**

Notes on scripts in WinBUGS 1.4.3

The standard way to control a WinBUGS model run is using the click-and-point menu interface. However, there is also a script language which enables all the menu options to be listed in a text file and run in batch mode. Scripts to run all of the models in the practical exercises are included in the data zip file you were provided with. Using scripts is generally much quicker than clicking-and-pointing. However, we recommend you start with the click-and-point approach so that you understand how WinBUGS works. The click-and-point approach is also better for debugging a model.

The section *Batch mode: Scripts* of the on-line User Manual in WinBUGS gives details of the script language.

To run a script, click on the window containing the script file to 'focus' it, and then select Script from the Model menu.

Notes on scripts:

- The model code, data and each set of initial values called by the script have to be stored in separate files
- Once the script has finished executing, the analysis is still 'live' and you can continue to use the WinBUGS menus interactively in the usual way.
- The script language currently has very limited error handling, so check that your model compiles correctly and that the data and initial values can all be loaded OK using the usual WinBUGS menu/GUI interface, before setting up a script to carry out a full analysis.

Notes on DIC tool in WinBUGS 1.4.3

- Remember that you should run sufficient burn-in iterations to allow the simulations to converge *before* you set the DIC tool, since you cannot discard burn-in values from the DIC calculations after the monitor has been set.
- There is an FAQ about DIC on the BUGS web site:
`http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/dicpage.shtml`

Notes on BUGS language

- A list of distributions and their syntax in the BUGS language is given in the on-line Help: `Help: User Manual: Distributions`.
- A list of functions and their syntax in the BUGS language is given in the on-line Help: `Help: User Manual: Model Specification: Logical Nodes`.
- Details of data formats accepted by WinBUGS are given in the on-line Help: `Help: User Manual: Model Specification: formatting of data`.
- Syntax for writing 'loops' and indexing vectors and arrays in the BUGS language is given in the on-line Help: `Help: User Manual: Model Specification: Arrays and indexing` and `Help: User Manual: Model Specification: repeated structures`.

Practical Class 1: Introduction to Monte Carlo using WinBUGS

A. Coin example from lecture 1

1. Start WinBUGS
2. Open `coins.odc` : this program will simulate throws of 10 balanced coins and record which give 8 or more heads (see lecture 1 slides 14 and 15).
3. First try running using the interactive interface.
   - Open up the Model : Specification window.
   - Highlight `model` and click on check model (this should also work without highlighting, provided `coins.odc` is the open window).
   - There is no data to load, so just click compile.
   - There is no real need to provide initial values as this is a Monte Carlo forward sampling from a known distribution, but the program still requires some initial values to be given. Click gen inits to generate some.
   - Open up the Inference : Samples window, type `Y` in the node window and then click set. This sets the monitor. Repeat for `P8`. Type `*` in the node window to indicate all monitored quantities.
   - Open up the Model : Update window and generate 1000 iterations.
   - Click trace to generate traces of the simulated values.
   - stats then gives summary statistics.
   - density plots distribution of `Y` and `P8`.
4. *Using the script language*: `WinBUGS` version 1.4.3 includes a facility for running models in batch mode using a script. The script to run the first model for the coins example in `WinBUGS` is in file `coins-script.odc`. Open this file and look at the commands to make sure you understand them (see section *Batch mode: Scripts* of the on-line User Manual in `WinBUGS` for more details). To run the script, click on the window containing the script file to focus it, and then select Script from the Model menu.

   See notes on scripts at the beginning of the Practical Exercise sheet for further details.
5. By editing the model file, find the probability that a clinical trial with 30 subjects, each with probability 0.7 of response, will show 15 or fewer responses.

B. Drug example from lecture 1

1. Open `drug-MC.odc` and carry out a `WinBUGS` run for this model, obtaining the results shown in the lectures (lecture 1 slides 24 to 25). You should be able to run it using the previous instructions (question A) and the short list given in the lectures (lecture 1 slide 18). If stuck, a script `drug-MC-script.odc` is available.

2. Edit the model code to specify a Uniform(0, 1) prior on the response rate `theta`, and re-run the analysis. (Note: the syntax for the uniform prior in WinBUGS is `dunif(a, b)` where `a` and `b` are the lower and upper bounds. The values of `a` and `b` can either be specified in the data file, or directly in the BUGS code (e.g. `a <- 1`), or just replace `a` and `b` by their values in the `dunif` statement.)

   - Plot the predictive distribution for the number of successes.
   - What is now the predictive probability that 15 or more patients will experience a positive response out of 20 new patients affected?

*C. Writing your own code (optional if you have time)

1. Generate 10000 observations from a Student t distribution from 4 degrees of freedom [WinBUGS code, `y ~ dt(0,1,4)`], and plot the density. Would you consider this a heavy -tailed distribution compared to the standard Normal distribution?

2. Find by simulation the expectation of the cube of a normal random variable with mean 1 and standard deviation 2 (remember the WinBUGS parameterisation of the normal is in terms of the precision = 1/variance, and y^3 is written as `pow(y,3)`).

Practical Class 2: Conjugate Bayesian inference using WinBUGS

A. Drug example from lecture 2

The `WinBUGS` code for fitting the beta-binomial model (lecture 2 slide 21) to the drug data can be found in file `drug-model.odc`. The data are in file `drug-data.odc`.

1. Open these files and carry out a `WinBUGS` run for this model
   (a) Check the model, load data, and compile the model, as you did for the models in practical 1.
   (b) You will need to specify some initial values for the unknown parameters in the model (`theta` and `y.pred`). You can either ask WinBUGS to generate these automatically (click `gen inits`) or you can load them from a file. An example set of initial values is given in file `drug-in.odc`; to load these, open the file and click `load inits`.
   (c) Before you start updating, set *sample* monitors for the drug response rate, `theta`, the predicted number of positive responses in 40 future patients, `y.pred`, and the indicator of whether 25 or more positive responses are predicted, `P.crit` (See section **Monitoring parameter values** of the hints handout).
   (d) Run 10000 iterations (or more or less if you wish) to obtain samples from the posterior distribution of the model parameters.
   (e) Produce summary statistics for all the variables you have monitored.
      - Provide a point and interval estimate for the treatment response rate
      - What is the probability that 25 or more patients will experience a positive response out of 40 new patients to be administered the drug?
   (f) Produce kernel density plots of the posterior distribution for `theta` and the predictive distribution for `y.pred`

   A script to run this model in `WinBUGS` is in file `drug-script.odc`, if you wish to use it.

2. Compare the model code (and data) with the code for the Monte Carlo analysis of the drug data that you carried out in practical 1. Make sure you understand the difference between the two analyses.

3. Edit the model code to specify a Uniform(0, 1) prior on the response rate `theta` (or equivalently, a Beta(1, 1) prior), and re-run the analysis.
   - How is the posterior estimate of `theta` affected?
   - How is the probability that 25 or more patients will experience a positive response out of 40 new patients affected?

B. THM data from lecture 2

From scratch, implement the THM example of inference on the mean of a Normal distribution (lecture 2 slides 26 to 30. (Note — you will need to specify values for both the THM observations, rather than just inputting their mean as the data, so just enter both values as 130).

1. First implement the model with the informative prior specified in the lecture notes (lecture 2 slide 27). The syntax for the normal distribution in WinBUGS is `dnorm(mu, tau)` where `mu` is the mean and `tau` is the *precision* (= 1/variance).
2. Now try using a non-informative prior on the unknown mean THM concentration (assume either a uniform prior with a wide range, or a normal prior with a large variance (small precision))
3. For each model, include statements in your WinBUGS code to:
   (a) obtain a sample from the predictive distribution of a future THM concentration in the zone
   (b) calculate the probability that the zone *mean* THM concentration exceeds 130 μg/l
   (c) calculate the probability that a future THM concentration measured in the zone exceeds 130 μg/l

*C. Disease risk in a small area (optional if you have time)

From scratch, implement the Poisson analysis of disease risk in a small area described at the end of Lecture 2 (slides 34 to 37).

Practical Class 3: Fitting Bayesian regression models in WinBUGS

A. Linear Regression: Small Area Estimation of Average Income

This question deals with the example of small area estimation of mean household income in Swedish municipalities, discussed in Lecture 4 (slides 6 to 16). The data, two sets of initial values and the basic model code are provided in files `income-dat.odc`, `income-in1.odc`, `income-in2.odc` and `income-model.odc`. Note that the data for this example are in the 'rectangular array' data format, rather than the 'list' format. The 284 rows correspond to areas, and the columns give each of the three variables (INCOME, AVAGE and RURAL). To load these data into `WinBUGS`, highlight the first row containing the column names and click on `Load data` at the appropriate stage (just clicking on `Load data` without highlighting the first row should also work, provided the data file is the focused window).

1. Open these files and carry out a `WinBUGS` run for this model. Unlike the `WinBUGS` analyses you have run so far (which involved either forward sampling or direct sampling from the closed-form posterior), for this analysis (and all subsequent practical exercises) you will need to carry out checks for convergence of the MCMC samples and discard any burn-in samples before making posterior inference. To do this, you should run 2 (or more) separate chains and use the Gelman-Rubin convergence diagnostic (lecture 3, slides 18 to 23) as follows:

   (a) Check and compile the model as before (see the instructions in section **Running a model in `WinBUGS`** of the hints handout), but remember to set the number of chains to be 2 (the default is 1) **before** you click *compile* (step 9 on the handout), and to load both sets of initial values (step 12 on the handout).

   (b) Set samples monitors on appropriate parameters — e.g. the regression coefficients and residual error variance. (see section **Monitoring parameter values** of the hints handout).

   (c) Run 10,000 iterations (updates), then look at history plots and autocorrelation plots of the sample traces and calculate the Gelman and Rubin convergence diagnostic (GR diag) for the parameters you have monitored.

      - After how many iterations do the simulations look like they have converged?

      (See section **Checking convergence** of the hints handout and lecture 3 slides 18 to 28).

   (d) Once you are happy with convergence,

      - Discard the burn-in iterations (by setting the `beg` option in the `Inference->samples` dialog box).
      - Then produce posterior summary statistics for the variables you have monitored.
      - Check the Monte Carlo standard error of each variable to assess the accuracy of your estimates (See section **Obtaining summary statistics of the poste-**

rior distribution of the hints handout, and lecture 3 slide 29 for a discussion of the MC error).

- How does the MC error change if you run more iterations? (Alternatively, look at how the MC error changes if you base your posterior summaries on fewer iterations — you can specify which iterations to summarise by setting the `beg` and `end` options in the *samples* tool in `WinBUGS`).

(e) Produce density plots of the posterior distributions of the regression coefficients, similar to that shown on lecture 4 slide 12.

2. Produce a plot of the posterior mean and 95% credible interval for the fitted values, `mu`, versus the covariate AVAGE, similar to that shown on lecture 4 slide 13. To do this:

   - you will need to have monitored the posterior samples of `mu` to produce this plot, so if you haven't already done this, set a monitor on `mu` now and run some further updates (say 5,000-10,000).
   - Then open the 'comparison tool' from the 'inference' menu, enter the relevant variable names ('node'=`mu`, 'axis'=`AVAGE` and 'other'=`INCOME`), then select the 'model fit' button. Don't forget to discard the burn-in samples by setting the value of 'beg' in the 'comparison tool' window before producing the plot.
   - You can edit the plot to add axis labels and change the margins by right-clicking on the plot and selecting 'properties'.
   - Note that the fitted line is not completely straight (unlike the plot shown in the lecture notes). Why do you think this is? (hint: the plot in the lecture notes on lecture 4 slide 13 is based on a model in which we had dropped the `RURAL` covariate).

3. Produce a box plot of the posterior estimates on mean income (i.e `mu`) for all the areas (see section **Plotting summaries of the posterior distribution** of the hints handout).

   - Don't forget to set the `beg` option in the plotting tool to discard the burn-in samples.
   - Edit the plot to order the box plots by rank of the posterior mean income (right click on the plot and select 'properties' then 'special').

4. *Classification of areas*: Rather than ranking the posterior mean estimates of income, as in the previous box plot, you can calculate the posterior distribution of the rank of each area. The `rank` function in WinBUGS takes a vector of values and a numerical index, so that the value associated with this index is ranked among all the others (see lecture 3 slides 33 to 35). Another way of classifying areas (e.g. to aid policy making) is to calculate the posterior probability that the mean income is below some threshold, such as the poverty line. Posterior probabilities can be calculated in WinBUGS using the `step` function (see lecture 3 slide 36). `step` takes a single numeric argument and returns 1 if the argument is ≥ 0 and zero otherwise.

   (a) Edit your model code to include the following statements to compute the posterior rank of the mean income in each area and the probability that the mean income is ≤ 1690 (the poverty threshold):

```
for(i in 1:284) {
   rankmu[i]<-rank(mu[1:284], i)
   p.poverty[i]<-step(1690-mu[i])
   AREA[i] <- i  # provides an area index for plotting
}
```

(b) Re-run your model and monitor `rankmu` and `p.poverty`. **Note: monitoring the ranks for each area causes WinBUGS to slow down considerably, since the rank calculation is quite a complex proceedure and must be executed at every iteration after the monitor is set**. To save you waiting for too long, carry out the necessary burn-in interations *before* you set a monitor on `rankmu`. Then just run a short number of further updates (say 500 or 1000). You would need to do more iterations than this for a real analysis, to ensure sufficient accuracy of your posterior estimates, but this will at least allow you to get a feel for how to calculate and summarise ranks of a set of parameters.

- Produce a box plot of the posterior distribution of the ranks.

(c) Alternatively, instead of including code to calculate the rank of each area in the bugs model, you could instead have just used the rank monitor tool in the `Inference` menu to monitor the rank of the fitted values, `mu`, directly. This is optimized to run much faster than explicitly calculating the rank in the model code, although you can't produce box plots of the ranks using the rank monitor tool.

(d) Next use the 'scatterplot' option from the `Inference->compare` tool to produce a plot of the posterior probability of being below the poverty line against area ID (enter `p.poverty` in the `node` box and `AREA` (the new index included in the above code) in the `axis` box (see lecture 3 slide 36 for an example)).

(e) Based on these plots, do you think that it is easy to separate areas with low income from the rest?

5. Set a monitor on DIC using the DIC tool at the bottom of the `Inference` menu. (If you still have a monitor set on `rankmu`, clear this before doing any further updates). Carry out further updates (say 5,000–10,000) and then view and record the value of DIC for this model. (See notes on using DIC tool at start of practical exercise sheets and lecture 4 slides 45 to 48).

6. Repeat question 1 above, but edit the model code so that you no longer centre the covariate `AVAGE` about its mean. What impact does this have on convergence?

7. Repeat question 1, but replacing the Normal error distribution for `INCOME` by a student t distribution on 4 df (see lecture 4 slide 14).

   - After completing the burn-in iterations, monitor DIC for this model and record its value after completing some further updates. Compare with the value of DIC obtained from the model with normal errors. Which model is preferred?
   - Compare the posterior estimates of the effect of `AVAGE` under the Normal and t models, and also compare the plots of the fitted values versus `AVAGE`. How do they differ?

*B. Non-linear regression, predictions and model comparison: Dugongs (optional if you have time)

In this practical you will work with the same data and models described in the Dugongs example shown in Lecture 4 (slides 33 to 35). The data, one set of initial values and a basic model file are provided in `dugongs-dat.odc`, `dugongs-in1.odc` and `dugongs-model.odc`.

1. Edit the model code to include statements to predict dugong length at ages 35 and 40 (lecture 4 slide 41).
2. Run the model, carrying out the usual checks for convergence and posterior sample size. Make sure you set monitors on the predictions that you have included in your model code.
3. Try producing the plot of these predictions similar to that shown on lecture 4 slide 43.
4. Try fitting a (rather inappropriate) linear regression model
   ```
           mu[i] <- alpha + beta*x[i]
   ```
   and use DIC to compare the fit of this model to the non-linear model. (See notes on using DIC tool at start of practical exercise sheets and lecture 4 slides 45 to 48). Note that there will be no `gamma` parameter in the linear model, but you can just leave it in the code as a prior distribution to avoid having to edit the initial values to remove the `gamma`'s.

*C. Logistic regression: Beetles (optional if you have time)

The data, some initial values and a basic model file are provided in `beetles-data.odc`, `beetles-in1.odc`, `beetles-in2.odc` and `beetles-model.odc`

1. Run the model to get the parameter estimates and plot shown on lecture 4 slide 29. Have a look at the history plots for the model parameters.
2. Try 'uncentering' the covariate and check the influence on the convergence of `beta`
3. Try different link functions: complementary log-log $[\log(-\log(1-p))]$ and probit $[\Phi(p)]$. These can be very sensitive to initial values! Try the two different ways of implementing the probit model as described in the lecture notes (lecture 4 slide 31).
4. Use DIC to compare the 3 alternative link functions for the beetles data.