SMSTC (2018/19)

Statistics

Lecture 18: MCMC Methods 3: WinBUGS

Valentin Popov
School of Mathematics and Statistics
University of St Andrews

www.smstc.ac.uk

Contents

18.1 Introduction

Recent years has seen a dramatic increase in the use of MCMC within the Bayesian statistical community. However, Bayesian analyses of statistical problems typically involves computer programs to perform the MCMC simulations. The writing of such bespoke code can often be difficult, particularly for non-programmers and researchers from different disciplines, such as biology, medicine, epidemiology etc. Despite this increase in the use of Bayesian methods, relatively few programs are readily (and freely) available for its implementation for general problems. There are many problems that need to be overcome, not least the fact that algorithms are generally problem-specific, with no automatic mechanism for choosing the best implementation procedure for a given problem. However, one program has overcome many of these problems, and is used throughout a wide variety of areas. This program is known by the acronym WinBUGS (Lunn *et al*,. 2000) and is freely available.

The original BUGS package (Bayesian inference Using Gibbs Sampler) was primarily developed for implementing a Bayesian approach within medical applications. However, since the model is specified by the user, it clearly is applicable in much wider fields. WinBUGS 1.4 is the latest incarnation that runs under Windows. The package provides a declarative S-Plus/R type language for specifying the statistical models and input parameters. The program then processes the model and data and sets up the sampling distributions required for the MCMC updates, using a variety of different moves. In addition, a stand-alone and menu-driven set of S-Plus/R functions (CODA) is supplied with WinBUGS to calculate convergence diagnostics and both graphical and statistical summaries of the simulated samples.

The package can be freely downloaded from the MRC Biostatistics Unit website:

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

The website contains detailed instructions for downloading and installing WinBUGS, as well as a plethora of examples using the package. Note that you also need to download a license key (this is also free). Note that WinBUGS will work without the license key, but only for the examples that are supplied.

There have been a number of development in recent years. These include `OpenBUGS` - an open source code; and `R2WinBUGS` which allows WinBUGS to be called from R, and the corresponding posterior distributions plotted etc. We will only focus on the basic WinBUGS package here – however, almost all of this information (with the exception of scripting) is directly applicable to OpenBUGS.

18.2 Getting started in WinBUGS

18.2.1 Setting up WinBUGS code

18–1. For WinBUGS code, you will need a file that contains:

i) the model specification;

ii) data values; and

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

18–2. 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 you can find your WinBUGS file.

ii) Once you find your file, either double-click on the file, or single-click on it and the `Open` - This will open your file into a WinBUGS window.

18–3. 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 your code (you 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 your model, and it will need to be corrected.

iii) If you wish to run multiple chains, then this needs to be specified here, since the number becomes fixed during the compilation process to come - Note that if you want to run the BGR convergence diagnostics you will need at least 3 replications.

18–4. Now that the model is correct, we need to read in the data and compile the code:

i) Highlight the word `list` corresponding to the data from our file and click on `load data` in the `Specifictaion 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.

18–5. The final input step is to read in the initial parameter values for the Markov chain:

i) Highlight the word `list` in your 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 you will need to 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.

18–6. Once the model has been compiled and the data read in, you are ready to start!!

18.2.2 Running WinBUGS code

18–1. You need to 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 you to specify the parameters that you wish to record. Note that you need to specify the parameters to be recorded prior to running any simulations.

18–2. 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.

ii) To run the simulations use the `Update Tool`. Input the number of iterations to be run (`update`) and how often to refresh the updates to write to the screen (`refresh`) - Once you press `update` this many iterations will run. 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 less than 5000 iterations.

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

18.2.3 Obtaining posterior estimates in WinBUGS

18–1. Once you have run your simulation, and the adaptation period is finished, you 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 that you entered earlier. Note that using the symbol `*` 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 you to thin the output. Setting `thin=10`, for example, means that the sample statistics are based on every 10th sample value.

18–2. 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`: Calculates estimates 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. This can be very useful for inputting into R to plot posterior distributions, for example.
- `quantiles`: Plots a running mean of the selected parameters, together with a symmetric 95% error bars.
- `bgr`: 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.

18.2.4 Using scripts in WinBUGS

An alternative, and complimentary, way of running WinBUGS[a] is to use a *script* rather than the Window pull-down menus, which can be quicker, particularly if running a number of simulations.

[a] The scripting language is substantially changed in OpenBUGS – see the OpenBUGS manual for details.

The script file essentially fills in the drop-down menu options within a set of commands in a single file. In order to use these scripts a minimum of 3 input files are needed, for example:

i) **"script.odc"**: The file with the list of script commands;

ii) **"model.odc"**: The model specification;

iii) **"data.odc"**: The data;

iv) **"inits.odc"**: Initial parameter values (optional - initial values can be generated).

To run a given script, click on `Model` in the toolbar, followed by `Script`. This will open a `Log` window specifying the commands that are being run (and possibly any error messages). Note that you can use text files, rather than native WinBUGS files, as long as they are .txt files.

The script file "script.odc" contains a list of commands corresponding to different pull-down menu options in WinBUGS. Further help and information on the use of scripts can be found under the `Help` toolbar and `User manual` option (or using the shortcut key F1). Then click on `Batch-mode: Scripts` in the user manual. Note that script commands can be combined with the usual Windows *point and click* version of WinBUGS. Essentially, the script commands are very useful in setting up the simulations, and then further exploration of the posterior and/or further simulations can be run using the pull-down menus in WinBUGS.

Comments

18–1. It is possible to combine both script commands and the use of the menu/dialog box interface.

18–2. The WinBUGS files for the model, data (and initial values) need to be placed in the WinBUGS root directory. The script file can be stored in an alternative directory - but it needs to be uppermost in the WinBUGS program when running the script command in WinBUGS.

Useful script commands

Typical commands to perform a Bayesian analysis include (equivalent pull-down menu):

- `display(`*option*`)`: Possible options are `log` and `window`. The entry `log` produces the output into a single window; `window` to separate windows. (`Options` → `Output options`).
- `check(`*file1*`)`: Checks the model given in *file1*. (`Model` → `Specification` → `check model`).
- `data(`*file2*`)`: Reads in the data given in *file2*. (`Model` → `Specification` → `load data`).
- `compile(`*no*`)`: Compiles the model using *no* chains. (`Model` → `Specification` (`num of chains =` *no*) → `compile`).
- `inits(`*no*`,`*file3*`)`: Specifies the initial parameter values in *file3* for chain number *no*. (`Model` → `Specification` → `load inits for chain` *no*).
- `gen.inits()`: Generates the initial parameter values for those not specified by `inits(`*no*`,`*file3*`)`. (`Model` → `Specification` → `gen inits`).
- `set(`*param*`)`: Specifies the parameter *param* to be monitored throughout the MCMC simulations. (`Inference` → `Samples` → `set`).
- `update(`*itns*`)`: Performs *itns* iterations of the MCMC algorithm. (`Model` → `Update` → `update`).

- `beg(begitn)`: Specifies *begitn* as the start of the sample values to be used to construct summary statistics. (`Inference` → `Samples` → `beg`).
- `end(enditn)`: Specifies *enditn* as the end of the sample values to be used to construct summary statistics. (`Inference` → `Samples` → `end`).
- `history(param)`: Produces the trace plot of *param*. (`Inference` → `Samples` → `history`).
- `density(param)`: Produces the marginal density of *param*. (`Inference` → `Samples` → `density`).
- `trace(param)`: Produces the dynamic trace plot of *param*. (`Inference` → `Samples` → `trace`).
- `stats(param)`: Produces the posterior summary statistics of *param*. (`Inference` → `Samples` → `stats`).
- `dic.set()`: Specifies that the DIC statistic is to be monitored (`Inference` → `DIC` → `set`). (See Section 20.4.1)

18.3 Example: Rats - Normal hierarchical model

We consider the example given within WinBUGS relating to the weights of rats. Details can be found via the `Help` menu, and `Example Vol I`. The data and model specification are presented before the associated WinBUGS code.

Data

The data correspond to the weight of 30 rats on 5 different days. We can display the data in the form given in Table 18.1.

	Day (x_j)				
Rat	8	15	22	29	36
1	151	199	246	283	320
2	145	199	249	293	354
3	147	214	263	312	328
4	155	200	237	272	297
⋮					
30	153	200	244	286	324

Table 18.1: The weights of the rats at each time.

18.3.1 Model 1

Let y_{ij} denote the weight of rat $i = 1, \ldots, 30$ at time $j = 1, \ldots, 5$. We assume a linear growth curve to the data with slope and intercept common to all individuals (we relax this assumption later) and assume that,

$$Y_{ij} \sim N(\alpha + \beta(x_j - \bar{x}), \sigma^2),$$

where α, β and σ^2 are parameters to be estimated; x_j denotes the jth time and $\bar{x} = 22$.

Note: Within Bayesian analyses it is usually important to normalise the data corresponding to the response variable and explanatory variable(s). This simply means we take the set of observed response data, subtract the sample mean from each value and divide by the sample standard

deviation. This is important in terms of the specification of the priors on the parameters for comparability and interpretability. However, for this rats example, for comparability with the example provided within the WinBUGS `Examples` we do not do this here.

Priors

We assume that we have no prior information on the parameters, and specify:

$$\alpha \sim N(0, 10^6); \qquad \beta \sim N(0, 10^6); \qquad \sigma^2 \sim \Gamma^{-1}(0.001, 0.001).$$

Note that within WinBUGS the Normal distribution is parameterised via the precision ($\tau = 1/\sigma^2$).

WinBUGS code

A WinBUGS code is provided in the file "rats.odc" that can be downloaded. This contains the model, data and initial values. The model is specified as:

```
model {

# Specify the likelihood:

    for(i in 1:N) {
        for(j in 1:T) {
            Y[i,j] ~ dnorm(mu[i,j],tau)
            mu[i,j] <- alpha + beta * (x[j] - xbar)
        }
    }

# Specify the priors:

    alpha ~ dnorm(0,0.000001)
    beta ~ dnorm(0,0.000001)

    tau ~ dgamma(0.001,0.001)
    sigma2 <- 1/tau
}
```

The data are given by:

```
list(x = c(8.0, 15.0, 22.0, 29.0, 36.0), xbar = 22, N = 30, T = 5,
    Y = structure(.Data = c(151, 199, 246, 283, 320,
                            145, 199, 249, 293, 354,
                            147, 214, 263, 312, 328,
                            155, 200, 237, 272, 297,
                            ........................
                            153, 200, 244, 286, 324),
                            .Dim = c(30,5)))
```

Finally, initial parameters can be read in using:

```
list(alpha = 0, beta  = 0, tau = 1)
```

Note that the initial parameter values need to be specified on the stochastic nodes within the WinBUGS code (i.e. the parameters that we specify a distribution on). Further, WinBUGS cannot generate an initial parameter value for a parameter with a $\Gamma(0.001, 0.001)$ prior specified, so that an initial value must be specified for the parameter `tau`.

Note that it is possible to combine the script commands with the standard WinBUGS menu format. For example, following the above script command, we could use the `Sample Monitor Tool` box in WinBUGS to give the trace plots for the parameters of interest.

Results

The simulations are run for 10,000 iterations. The corresponding trace plots of the parameters α, β, σ^2 and τ are provided in Figure 18.1 (thinned by 10 to reduce the file size of the figure).

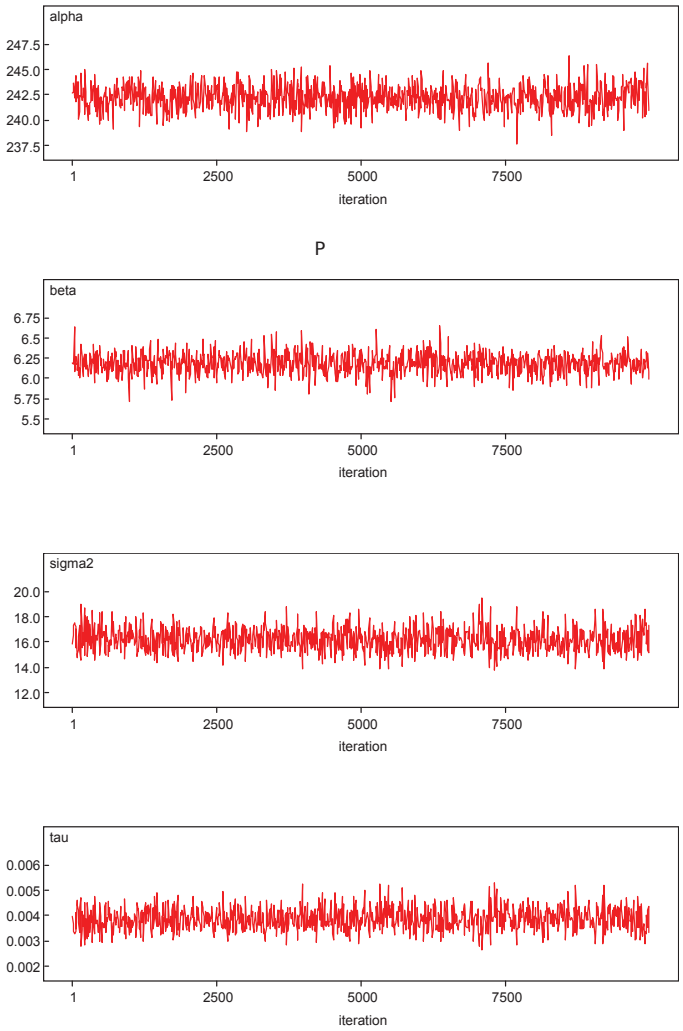

Figure 18.1: Raw trace plots for the parameters α, β, σ^2 and τ.

There is no real discernible burn-in from the plot, but still to be conservative, we assume a burn-in of 1,000 iterations. The corresponding (default) summary statistics are given by,

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	242.2	1.321	0.01373	239.6	242.2	244.8	1001	9000
beta	6.186	0.1338	0.00137	5.922	6.185	6.451	1001	9000
sigma2	16.22	0.9627	0.01011	14.5	16.18	18.23	1001	9000
tau	0.003842	4.521E-4	4.603E-6	0.00301	0.003821	0.004757	1001	9000

The first column provides the name of the parameters with the 2nd and 3rd columns the corresponding posterior (marginal) mean and standard deviation. The next column provides the Monte Carlo error. In this case these values all appear to be very small compared to the posterior mean (so that if the simulations are replicated we would expect only a very small deviation from the current values). The next 3 columns provide the 2.5%, 50% and 97.5% posterior quantiles, providing the posterior symmetric 95% credible interval and posterior median. The final 2 columns simply reports the range of iteration values used to calculate the posterior summary statistics. Marginal density plots can be obtained using the `density` command within the `Sample Monitor Tool` and are provided in Figure 18.2 - each of these show a unimodal posterior distribution).

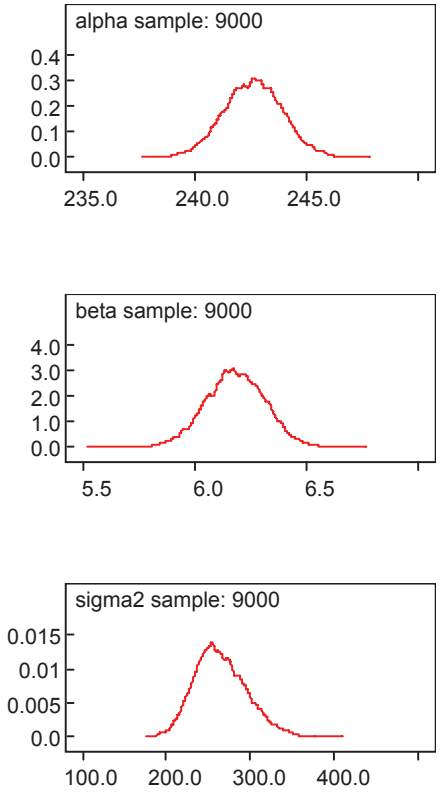

Figure 18.2: Marginal density plots for the parameters α, β and σ^2.

We note that α can be interpreted as the intercept term (after the weights have been centred) and β the slope of the linear regression. In particular, since β is clearly positive, there is strong evidence that there is a positive relationship between the weight of the rat and time, so that the weight of the rat increases with time (which we assume is linear in nature). In addition we have that the 95% symmetric credible interval for β does not contain the value of 0 which would correspond to no (linear) relationship between weight and time. Thus an *ad-hoc* interpretation of this output might conclude that there is evidence that time is important in this regression (note we return to this issue in §20.4 where we conduct a formal model selection procedure or hypothesis test).

Trace plots suggest that convergence of the Markov chains to the stationary distribution is very rapid. However, we run multiple chains from different starting points ($\alpha = 0, \beta = 0, \sigma^2 = 1$, $\alpha = -100, \beta = -10, \sigma^2 = 0.1$ and $\alpha = 100, \beta = 10, \sigma^2 = 10$). The corresponding BGR trace plots are provided in Figure 18.3. The BGR statistic is plotted in red and the (normalised) pooled and

mean 80% credible intervals width are green and blue (respectively). The red line should converge to 1 and the green and blue lines converge to stability. Clearly convergence is very rapid and using a burn-in of 1,000 iterations is clearly very conservative. Note that the actual $\hat{R}$ statistics can be obtained in WinBUGS by double-clicking on the given plot followed by a `ctrl-left-mouse-click`.

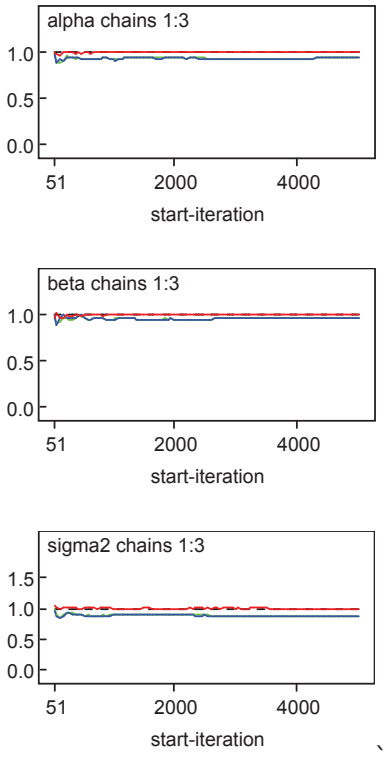

Figure 18.3: BGR convergence diagnostic plots for the parameters α, β and σ^2.

Finally, for comparison with the summary statistics above, below are the summary statistics obtained for one of the other Markov chains run:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	242.2	1.334	0.01416	239.6	242.2	244.8	1001	9000
beta	6.185	0.1334	0.00139	5.922	6.186	6.442	1001	9000
sigma2	16.22	0.9428	0.009412	14.49	16.18	18.2	1001	9000
tau	0.003841	4.441E-4	4.361E-6	0.00302	0.003821	0.004765	1001	9000

Clearly, these are very similar to those obtained in the initial Markov chain. Thus, there is no evidence of lack of convergence in this case.

We conduct a prior sensitivity analysis by rerunning the MCMC iterations in WinBUGS using different priors on each of the parameters. For example, we specify the following priors,

$$\alpha \sim N(0, 10^8); \qquad \beta \sim N(0, 10^8); \qquad \sigma \sim U[0, 100].$$

In WinBUGS these could be input using,

```
# Priors:
    alpha ~ dnorm(0,0.00000001)
    beta ~ dnorm(0,0.00000001)
    tau <- 1/sqrt(sigma)
    sigma ~ dunif(0,100)
```

Note that with this change in prior we need to also change the inputting of the initial parameter values as we need to specify initial values for the parameters which are defined to have a prior distribution (α, β and σ instead of τ).

```
# Input initial parameter values:
list(alpha = 0, beta = 0, sigma = 1)
```

Once more convergence is extremely swift and we obtain the corresponding posterior summary statistics:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	242.6	1.316	0.01381	240.1	242.6	245.2	1001	9000
beta	6.185	0.1337	0.001373	5.92	6.184	6.444	1001	9000
sigma2	16.26	0.9535	0.01123	14.53	16.19	18.25	1001	9000
tau	0.003822	4.443E-4	5.129E-6	0.003004	0.003813	0.004737	1001	9000

These are again very similar to the previous posterior summary statistics. Thus we would typically conclude that the posterior distribution is data-driven.

18.3.2 Model 2

Let y_{ij} denote the weight of rat $i = 1, \ldots, 30$ at time $j = 1, \ldots, 5$. We assume a linear growth curve to the data and assume that,

$$Y_{ij} \sim N(\alpha_i + \beta_i(x_j - \bar{x}), \sigma^2),$$

where α_i, β_i and σ^2 are parameters to be estimated; x_j denotes the jth time and $\bar{x} = 22$. So that the slope and intercept terms may differ between individuals.

Priors

We assume that we have no prior information on the parameters, and specify:

$$\begin{aligned} &\alpha_i|\mu_\alpha, \sigma^2_\alpha \sim N(\mu_\alpha, \sigma^2_\alpha); \qquad \mu_\alpha \sim N(0, 10^6); \qquad \sigma_\alpha \sim U[0, 100]; \\ &\beta_i|\mu_\beta, \sigma^2_\beta \sim N(\mu_\beta, \sigma^2_\beta); \qquad \mu_\beta \sim N(0, 10^6); \qquad \sigma_\beta \sim U[0, 100] \\ &\sigma^2 \sim \Gamma^{-1}(0.001, 0.001). \end{aligned}$$

Thus, we specify a hierarchical prior on the parameters α_i and β_i, for $i = 1, \ldots, 30$, (see Section 16.5.3).
Note that these are the same prior specifications as described within WinBUGS - our notation is slightly different. In particular they parameterise the Normal distribution via the precision ($\tau = 1/\sigma^2$).

WinBUGS code

The WinBUGS code is provided in the file "rats1.odc" that can be downloaded. This contains the model, data and initial values. The model is specified as:

```
model {

# Specify the likelihood:

    for(i in 1:N) {
        for(j in 1:T) {
            Y[i,j] ~ dnorm(mu[i,j],tau)
```

```
            mu[i,j] <- alpha[i] + beta[i] * (x[j] - xbar)
        }
        alpha[i] ~ dnorm(mu.alpha,tau.alpha)
        beta[i] ~ dnorm(mu.beta,tau.beta)
    }

# Specify the priors:

    mu.alpha ~ dnorm(0.0,1.0E-6)
    mu.beta ~ dnorm(0.0,1.0E-6)

    sigma.alpha ~ dunif(0,100)
    sigma.beta~ dunif(0,100)


    tau.alpha <- 1/(sigma.alpha*sigma.alpha)
    tau.beta <- 1/(sigma.beta*sigma.beta)

    tau ~ dgamma(0.001,0.001)
    sigma <- 1 / sqrt(tau)
}
```

The inputting of the data remains the same, but we now need to specify initial parameter values for each α_i, β_i (for $i = 1 \ldots, 30$), μ_α, μ_β, σ_α, σ_β and $\tau = 1/\sigma^2$ parameters. For example:

```
list(alpha = c(250,250,250,250,250,250,250,250,250,250,250,250,250,
250,250,250,250,250,250,250,250,250,250,250,250,250,250,250,250),
beta  = c(6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6),
mu.alpha = 150, mu.beta = 10, tau = 1, sigma.alpha = 1, sigma.beta = 1)
```