Chapter 7

Introduction to WinBUGS

WinBUGS is the Windows version of the **B**ayesian analysis **U**sing the **G**ibbs **S**ampler software developed by the UK Medical Research Council and the Imperial College of Science, Technology and Medicine, London.

At the time of writing, WinBUGS is freely available at

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

BUGS Background

The BUGS project began in 1989 in the Biostatistics Unit of the Medical Research Council, U.K.

- "Classic" BUGS
  - Thomas et al. (1992).
  - Batch mode operation.
- WinBUGS
  - Developed by MRC for Windows operating system in late 1990's.
  - Current version (May 2006) is 1.4, with upgrade to version 1.4.1.
- OpenBUGS
  - Open source version of WinBUGS for running on Windows and Linux, as well as inside the R statistical package (BRugs).

BUGS and R

- CODA
  - R package for convergence diagnosis (Best et al., 1995).
  - Other similar R packages available, e.g., BOA.
- R2WinBUGS
  - R package for executing WinBUGS from R (Sturtz et al., 2005).
  - Uses the WinBUGS scripting language (the WinBUGS interface appears on the desktop).
- BRugs
  - R package for running OpenBUGs components from within R.

WinBUGS Structure

Essentially, the WinBUGS program is simply a syntactical representation of the model, in which the distributional form of the data and parameters are specified. (It does not require (in most cases) knowing the formulae of density functions).

For example, $y \sim Binomial(n, p)$

is written in WinBUGS as

```
y ~ dbin(p,n)
```

and $y_i \sim Binomial(n_i, p_i)\,,\quad i = 1,\ldots M$

is written in WinBUGS as

```
for(i in 1:M) {y[i] ~ dbin(p[i],n[i])}
```

WinBUGS Structure

NOTE: The interpretation of WinBUGS code is unlike that of other programming languages such as R.

In R: $y = y+1$ makes perfect sense.

In WinBUGS: y <- $y+1$ is nonsensical, because a datum (or parameter) can not equal itself plus unity.

If you can write the model down on paper, then you should be able to code it up in WinBUGS.

CAUTION: There is no guarantee that WinBUGS will "work".

WinBUGS: Practical 1

- Start up WinBUGS.
- Click File->New to open up a BUGS window.
- Type in the code:

```
model NormalPrior
{
	mu ~ dnorm(0,1)
}
```

- Click Model->Specification
  - Click "check model". Bottom left of screen should say "Model is syntactically correct (else it will provide an error message and the cursor will be positioned at the error.)
  - This example contains no data (so ignore the "load data" step, for now).
  - Click "compile".
  - Click "gen inits".
- WinBUGS is now ready to generate the MCMC sample.

WinBUGS: Practical 1

- Click Model->Update to open the Update Tool window.
  - Click “update”.
  - You’ve just generated 1000 samples from a Markov chain with a standard normal stationary distribution!
  - These first thousand samples have not been saved, which is good practice because the chain needs to burn in.
- Click Inference->Samples to start the Sample Monitor.
  - Type “mu” in the node box and click on “set”.
- Go to Update Tool and click “update”.
- Go to Sample Monitor Tool
  - Several choices of summary plots and statistics can now be selected.

WinBUGS: Practical 2

Repeat Practical 1, but with a non-normal distribution.

To see the choices of distribution:

- Click Help > User Manual.
- Scroll down and click on Contents.
- Follow the links:

  Model Specification > The BUGS language: stochastic nodes > Distributions

WinBUGS: Practical 3

The previous two examples did not include any data.

Here, we will assume that we observe a single observation y~N(μ,1), where μ has a standard normal prior distribution. (In this case $\mu|y$ ~ N(0.5y,0.5))

```
model NormalPrior
{
  mu ~ dnorm(0,1)
  y ~ dnorm(mu,1)
}
list(y=  )
```

Insert a number here

After checking the model syntax, use the mouse to highlight the word "list", and click "load data" on the Specification Tool window. Then, proceed as before.

WinBUGS: Practical 4

Recall, the IID Normal example with known variance of the data:

```
model IIDNormal
{
  for(i in 1:10) { y[i] ~ dnorm(mu,1) }
  mu ~ dnorm(0,1)
}
list(y=c(1.64,1.10,1.33,0.27,0.61,
         0.25,-0.02,-0.08,0.43,-0.53 ))
```

WinBUGS example

More generally, in the case of ten (say), iid observations:

```
model IIDNormal
{
  for(i in 1:10) { y[i] ~ dnorm(mu,1) }
  mu ~ dnorm(0,1)
}
list(y=c(1.64,1.10,1.33,0.27,0.61,
         0.25,-0.02,-0.08,0.43,-0.53 ))
```


Moving along…

Now, let's drop the assumption of known variance, and instead we shall assume that the data are IID Normal with unknown mean μ and unknown variance σ^2.

If an informative prior on σ^2 is to be specified then an inverse gamma* distribution will typically be used. This corresponds to a gamma distribution on $1/\sigma^2$.

If a "non-informative" prior on σ^2 is desired then it can be approximated by specifying a highly dispersed gamma distribution on $1/\sigma^2$.

*The gamma distribution is a generalization of the χ^2 .

Moving along…

Note: For normal densities, Bayesian's typically work with $1/\sigma^2$ (precision) rather than σ^2 (variance). In WinBUGS, the normal density is specified as dnorm(mean,precision).

IID Normal, μ and σ^2 unknown

```
model IIDNormal2
{
  for(i in 1:10) { y[i] ~ dnorm(mu,prec) }
  var <- 1/prec
#Add priors
  mu ~ dnorm(0,1)
  prec ~ dgamma(0.001,0.001)  #Disperse gamma
}
#Data
list( y=c( 1.64,1.10,1.33,0.27,0.61,0.25,
          -0.02,-0.08,0.43,-0.53 )  )
#Inits
list( mu=0, prec=1)
```

WinBUGS syntax

In WinBUGS, the tilde sign ~ means “distributed as”. It is used to:

- Specify the distribution of the data.
- Specify the prior distributions.
- Values to the left of a ~ are called “stochastic”.

The left arrow <- corresponds to the “equals” sign. It is used in calculations, such as **`var<-1/prec`**

- Values to the left of a <- are called “logical”.

Linear regression: Lines example

```
model
{
  for(i in 1:N){
     Y[i] ~ dnorm(mu[i], tau)
     mu[i] <- alpha + beta*(x[i] - mean(x[]))
               }
  sigma <- 1/sqrt(tau)
  alpha ~ dnorm(0, 1.0E-6)
  beta ~ dnorm(0, 1.0E-6)
  tau ~ dgamma(1.0E-3, 1.0E-3)
}
list(x=c(1,2,3,4,5), Y=c(1,3,3,3,5), N=5 )
list(alpha = 0, beta = 0, tau = 1)
```