Introduction to WinBUGS

Lin Zhang
Biostatistics, University of Minnesota

1 About WinBUGS and Its Installation

1.1 BUGS (Bayesian inference Using Gibbs Sampling)

BUGS is a programming language for Bayesian analysis using Markov chain Monte Carlo(MCMC) methods. The BUGS project develops into several different versions of software that are all very similar to each other:

- Classic BUGS
- **WinBUGS** – stable version, features a graphical user interface (GUI)
- OpenBUGS – open source, under development
- JAGS(Just Another Gibbs Sampler)

We mainly focus on WinBUGS in this file and most of the course and will also introduce JAGS implemented in R later.

1.2 Install WinBUGS

The software WinBUGS is freely available at MRC Biostatistics Unit website. To install on a 64-bit machine, you need to download a zipped version of the whole file structure, unzip it, and put the extracted folder **"WinBUGS14"** into Program Files or whereever you want.

You will also need to download the patch to upgrade to version 1.4.3 and the "immortality key" for unrestricted use.

To install the patch and the key, do the following steps:

1. Start WinBUGS by double clicking the executable file **"WinBUS14.exe"** in the folder where you saved the file. (Or you can create a shortcut on Desktop for each access.)
2. Click the menu **"File"** → **"Open..."**, find the patch file in ".txt" format in the folder where you saved it, and open it.
3. Click the meanu **"Tools"** → **"Decode"**. Click the button **"Decode All"**, and click "okay" for all pop-up windows. Now the patch is now installed and you can close the patch file.
4. Now install the immortality key as what you did to install the patch.

2 The BUGS language

2.1 Variable/Nodes

The BUGS/WinBUGS has its unique language. Comparing to R, the most fundamental difference is that it has two kinds of variables or called **"nodes"** in WinBUGS.

– Logical (deterministic) nodes: assign with **" <- "** (like in R): e.g.
x <- $mu + e$
y <- $x * x$

– Stochastic nodes: assign with **"∼"** (is distributed as): e.g
$x \sim$ dnorm(mu,tau)
$x \sim$ dbeta(a,b)

The distributions implemented in WinBUGS include:

Bernoulli, Binomial, Categorical, Negative Binomial, Poisson, Beta, Chi-squared, Exponential, Gamma, Normal, Pareto, Student-t, Uniform, Weibull, Multinomial, Multivariate normal, Multivariate student-t, Dirichlet, Wishart

For a complete list of distributions and their parameterization, please check **"Help → WinBUGS user manual → Distributions"**.

You can also specify **truncated** distribution using "**I**(**lower**, **upper**)", for example

- Truncated on two sides: $x \sim$ dnorm(0,1)$I(-1, 1)$
- Left truncated: $x \sim$ dnorm(0,1)$I(0,)$
- Right truncated: $x \sim$ dnorm(0,1)$I(, 0)$

2.2 Data

In WinBUGS, data can be represented in two formats or combination of the two formats. Unfortunately, WinBUGS cannot directly read data from external .csv or tab delimited .txt files.

– **List** (similar to the `list` in R)

```
list(
        xbar=22, N=3, T=5,
        x = c(8.0, 15.0, 22.0, 29.0, 36.0),
        Y = structure(
                        .Data=c(151,199,246,283,320,
                                145,199,NA,293,354,
                                153,200,244,286,324),
                        .Dim=c(3,5)
                       )
    )
```

Note that (1) there is **NO space** between "list" and "("; (2) the matrix Y is **filled by row** in WinBUGS; (3) use **NA** for missing data.

– **Rectangular data array** (similar to the `data frame` in R)

ID[ ]	Age[ ]	Gender[ ]
1	26	0
2	52	1
3	45	1
4	49	0

END

The array **must** end with an **"END"** statement, followed by at least one blank line.
Multi-dimensional arrays can be specified same way.

$Y[, 1]$	$Y[, 2]$	$Y[, 3]$	$Y[, 4]$	$Y[, 5]$
151	199	246	283	320
145	199	249	293	354
153	200	244	286	324

END

2.3 Indexing

Just like R, WinBUGS uses square brackets [ ] to index vectors, matrices, and high-dimensional arrays.

$x[1 : 5]$	first 5 elements in a vector x
$x[ ]$	all elements in vector x
y[2,]	2nd row in a matrix y
y[,]	all elements in matrix y
y[2, 2, 4]	one element in 3D array y

2.4 Functions

A list of functions defined in WinBUGS can be checked at **"Help → WinBUGS user manual → Model Specification → Logical nodes → Table I: Functions"**. Here I provide some of them

$abs(x)$	$\|x\|$
$sqrt(x)$	$x^{1/2}$
$pow(x, y)$	x^y
$step(x)$	1 if $x \geq 0$; 0 o.w.
$exp(x)$	$exp(x)$
$log(x)$	$ln(x)$
$logit(x)$	$ln(x/(1-x))$
$sin(x)$	$sine(x)$
$equals(x1, x2)$	1 if $x1 = x2$; 0 o.w.
$max(x1, x2)$	find the maximum
$sum(v)$	sum of all elements in a vector
$mean(v)$	mean of a vector
$sd(v)$	standard deviation
$ranked(v, s)$	find the s^{th} smallest number in a vector
$inprod(v1, v2)$	inner product of two vectors
$inverse(m)$	matrix inversion

2.5 Loop statement

The loop statement is just as in R:

for (variable **in** vector) { expression. }

Unfortunately, there is not **"if"** statement in WinBUGS. Use **"step"** function to do the trick.

tmp[1] $\sim$ dnorm(μ, 1)I(0,)

tmp[2] $\sim$ dnorm(μ, 1)I(,0)

$z[i]$ <- tmp[1]*step(y[i]) + tmp[2]*(1-step(y[i]))

3 Model Specification

Bayesian analysis in WinBUGS involves **3 steps**:

1. Coding step: **specify** model, data, and initial values in an **editor** window
2. Loading step: **load** model, data, and initial values using **interactive windows**
3. Sampling step: **run MCMC** and collect posterior samples using **interactive windows**

3.1 Coding in Editor

We can create a new script editor by clicking **"File → New"** or open an existing script file by clicking **"File → Open..."**. A WinBUGS script should include specification of three basic elements:

1. **Model**: This includes likelihood model and prior distribution for parameters. The code to specify a model should look like this:

```
model{
        # Likelihood: y[i] ~ N(theta, sigma2), i=1,...,n
        for(i in 1:n) {
            y[i] ~ dnorm(theta,prec.y)
        }
        prec.y <- 1/sigma2

        # Prior: theta ~ N(mu,tau2)
        theta ~ dnorm(mu,prec.theta)
        prec.theta <- 1/tau2
       }
```

This is a Normal-Normal model. Note the model is specified in the big braces after the **"model statement"**. In WinBUGS, we also use "#" for comments, but the normal distribution in WinBUGS has different parameters: mean and *precision* (inverse of variance).

2. **Data**: As detailed previously, it could be specified as a list or a rectangular matrix (with END statement) or a combination of them. Continue with the Normal-Normal model, we specify the data

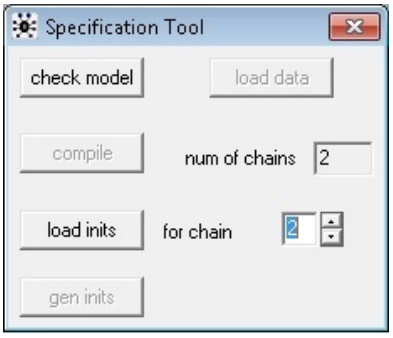
```
list(
    mu=2, sigma2=1, tau2=1, n=10,
    y=c(7.2, 5.0, 6.7, 4.6, 6.4, 5.5, 3.6, 6.5, 7.9, 6.6)
   )
```

3. **Initials**: Initial values are specified in the same format as Data; **alternatively,** they can be generated by interactive window. We will detail this in the next section. Note that if you run multiple chains, you need to specify the same number of sets of initial values.

3.2 Loading using the Interactive Window "Specification Tool"

Now that we have the three elements specified in the script, we can load the model, data, and initials into the WinBUGS program. Do the following steps:

1. Open **"Specificatoin Tool"** window: click the menu **"Model → Specification... "**.



2. Load/Check model: select the model statement in script, and click **"check model"** in Specification Tool window. At the bottom left of WinBUGS window, you will see "model is syntactically correct" if the model is correct.

3. Load model: select the data in script, and click **"load data"** in Specification Tool. The bottom of WinBUGS should display "data loaded".

4. Compile: specify the number of chains you want to run in the "num of chains" box, and click **"compile"**. The bottom will display "model compiled".

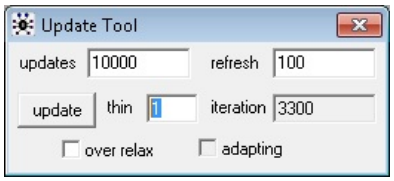
5. Load/Generate initials: select the initial values in script, and click **"load inits"** in Specification Tool; alternatively, click **"gen inits"** to let WinBUGS to generate initials. You should see at the bottom "model is initialized" or "initial values generated". **If you are to run multiple chains, change the chain index number (in the "for chain" box), and load/generate a different set of initials.**

Now that you are done with model specification, you can close the "Specification Tool" window.

3.3 Sampling using the Interactive Window "Sample Monitor Tool" and "Update Tool"

Before carrying out posterior sampling, we need to specify (1) the number of MCMC iterations to run, and (2) the nodes/parameters, of which we want to collect posterior samples . To do these, we need to open two windows:

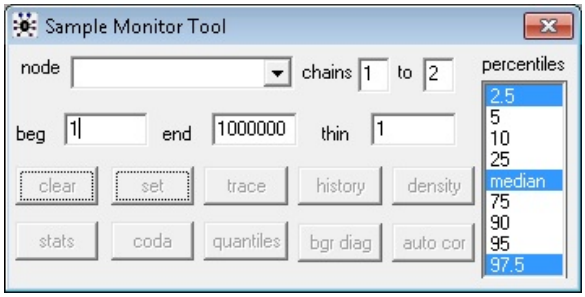
1. **"Update Tool"** window: open by menu **"Model → Update..."**.



   - update: number of MCMC iterations to run
   - refresh: blocks of iterations that runs at a time (you can stop MCMC after finishing the current block by clicking "update" buttom).
   - thin: the number of iterations to skip before collect the next sample (reduce autocorrelation).
   - iteration: current iteration number after thinning.

   After specifying "updates", "refresh", and "thin", you can click "update" to start MCMC iterations. However, at current stage, the posterior samples will not be saved. You can use this as **"burn-in"** period.

2. **"Sample Monitor Tool"** window: open by menu **"Inference → Samples..."**.



   We specify the nodes/parameters of which we will collect posterior samples. For a single parameter of interest, do the following steps:

   (a) Type the name of the parameter in the **"node"** box.
   (b) Click on "set".
   (c) Click on "Update" in the "Update Tool" window.

   If we have multiple unknown quantities that we want to generate samples for

   (a) Type the name of one parameter in the **"node"** box.
   (b) Click on "set".
   (c) Repeat step (a) & (b) for all unknown quantities
   (d) Type * in the **"node"** box to select all the nodes.
   (e) Click on "Update" in the "Update Tool" window.

4 Check Posterior Sampling Results

The "Sample Monitor Tool" window contains a bunch of tools to quickly check posterior sampling.

- trace: dynamic plot of value over iteration during MCMC sampling
- history: plot a complete trace
- density: plot smoothed kernel density estimate of posterior
- stats: summary statistics
- quantiles: plot running mean and 95
- bgr diag: Gelman-Rubin diagnostic statistic of convergence
  - Blue: width of 80
  - Green: width of 80
  - Red: R=pooled/within
- auto cor: plot autocorrelation function of the posterior samples

Each of the above will display plots/tables in separate windows. We can let them display in one log window:

1. Click on menu **"Options → Output options..."**.
2. Check **"log"** in the pop up window.
3. Click on menu **"Info → Open log"**, and the log window will pop up.

Now plots/tables will be added to the log window.

5 Saving Outputs from CODA files

Click on **"coda"** in the "Sample Monitor Tool" window, and two types of windows will pop up:

- **Output files**: each file contains the outputs for each chain, including iteration numbers and sampled values for all unknown quantities of interest
- **Description file**: indicating which lines of each output file corresponding to which parameters.

Now you can **save each CODA file** (including output and description) separately by clicking on **"File → Save As ..."**. Give a file name, choose **"Plain Text (.txt)** for the file type, and save. Now you can analyze the outputs by reading them into R.