An Introduction to WinBUGS

* PIMS Collaborative Research Group
* Summer School on Bayesian Modeling and Computation

14th July, 2008

Sebastien Haneuse, PhD
Group Health Center for Health Studies
Department of Biostatistics, University of Washington
## Outline

- Goals and Scope
- Preliminaries
- The WinBUGS Software
- Navigating the WinBUGS Menu System
- Additional Topics
Goals and Scope

Bayesian analyses

- Formulate the scientific question
  - understanding the association between two variables
  - prediction of some future event

- Identify a corresponding parameterization
  - ‘translation’ of the scientific question into statistical terms and quantities

- Specify prior distributions for the unknown parameters
  - translation of substantive knowledge
  - identification of ‘non-informative’ priors
  - establish several options towards a sensitivity analyses
Goals and Scope

Bayesian analyses (cont)

- Characterize linkage between the parameters and the observed data
  - specification of the likelihood

- Apply Bayes Theorem
  - posterior $\propto$ prior $\times$ likelihood
  - turn the Bayesian ‘handle’
  - examine features of the posterior

- Interpret the results and refine the scientific question
Goals and Scope

Where does WinBUGS fit in?

- We’ve seen that there are many steps in a Bayesian analysis
- WinBUGS facilitates turning the Bayesian handle
  - computational tool
  - via Markov Chain Monte Carlo (MCMC)
- Cuts out much of the hard work for a very general class of models
  - model: likelihood/prior combinations
- Sole purpose is to implement MCMC-based analyses
  - perform tasks with a few lines of code that would take (potentially) hundreds of lines of ‘home-grown’ code
  - highly efficient, both in terms of model specification but also in the MCMC implementation
Goals and Scope

Caveats

- Underlying WinBUGS is a *language*
  - even within a single language, there are often many ways of ‘saying’ the same thing!
  - many idiosyncrasies

- Cannot accommodate all analyses
  - OpenBUGS is an open source version

- WinBUGS User Manual:
  
  Beware: MCMC sampling can be dangerous!
Goals and Scope

Scope

- A good way to learn, and implement your own analyses within WinBUGS, is to build on work by others
  - hundreds of examples out there
  - WinBUGS manual and website
  - various books

- A reasonable strategy is to adapt existing code to your specific setting

- The main purpose here is to provide an overview of WinBUGS functionality and syntax
  - understanding models and adapting them to your own purposes
Goals and Scope

Scope (cont)

- Aimed at a new WinBUGS user
  - assume some familiarity with the Bayesian philosophy and theory
  - underlying premise of using draws from a distribution as a means to approximate its features

- Not a tutorial on the intricacies of Bayesian reasoning or MCMC
  - model and prior specification
  - convergence, optimal sampling strategies, etc
  - plenty of good texts for this

- An introductory WinBUGS book is on the way
Outline

- Goals and Scope

- Preliminaries
  - Lung cancer data from Ohio
  - Gibbs sampling algorithm

- The WinBUGS Software

- Navigating the WinBUGS Menu System

- Additional Topics
## Preliminaries

### Lung cancer data from Ohio

- Population estimates and lung cancer death counts in 1988
  - 88 counties
  - 2,220,177 individuals
  - 5,533 deaths

- Covariates include
  - gender
  - race (white vs. non-white)
  - age (10-year bands)

- National Center for Health Statistics Compressed Mortality file
  - Waller et al. (1997)
Preliminaries

Scientific interest

- Estimate the effects of gender and race on lung cancer mortality
  * marginal associations

- To assess their full joint effect, we might be interested in a potential interaction
  * consider a modeling framework
Preliminaries

Data

- Four unique covariate patterns
  - indexed using \( j \), where \( j = 1, \ldots, 4 \)

- \( N_{kj} \) denotes population size in the \( k^{th} \) county with the \( j^{th} \) covariate pattern

- \( Y_{kj} \) corresponding number of lung cancer deaths

- Dataframe in R

```r
> ohio
  county sex race pattern   N  Y
[1,] 39001  0  0     1 2631  9
[2,] 39001  1  0     2 2883  1
[3,] 39001  0  1     3  16  0
[4,] 39001  1  1     4  23  0
[5,] 39003  0  0     1 9315 31
[6,] 39003  1  0     2 12095 19
...```
Likelihood/Model

- Observed data: \( Y_{kj} \sim \text{Binomial}(N_{kj}, \pi_{kj}) \)

- Logistic disease model
  \[
  \text{logit} \, \pi_{kj} = \beta_0 + \beta_S S_j + \beta_R R_j + \beta_{SR} S_j \times R_j + V_k
  \]

  * \( R = 0/1: \) white vs. non-white
  * \( S = 0/1: \) male vs. female
  * \( V_k, \) county-specific random effect

- Assume the \( V_k \) are normally distributed across the 88 counties
  \[
  V_k \sim \text{Normal}(0, \sigma^2)
  \]
Preliminaries

Prior specification

- In the absence of substantive scientific input, adopt vague, non-informative priors

- Regression coefficients

  \[ \beta \sim \text{Normal}(0, 1e^6) \]

- Random effect variance

  * often parameterized in terms of the precision

  \[ \tau = \frac{1}{\sigma^2} \sim \text{Gamma}(0.5, 0.01) \]
  
  * mainly to exploit conjugacy

- Consider the *induced* prior for \( \sigma \)

  * median: 0.21
  * \((2.5^{\text{th}}, 97.5^{\text{th}})\) quantiles: \((0.06, 4.51)\)
A WinBUGS Session
A WinBUGS Session

Components/Windows

- Code file
  - model specification
  - data
  - initial values

- WinBUGS tools
  - specification tool
  - update tool
  - sample monitor tool

- Convergence diagnostics
  - time series plot
  - Brooks-Gelman-Rubin statistic
A WinBUGS Session

Components/Windows (cont)

- Posterior summary statistics

<table>
<thead>
<tr>
<th>node</th>
<th>mean</th>
<th>sd</th>
<th>MC error</th>
<th>2.5%</th>
<th>median</th>
<th>97.5%</th>
<th>start</th>
<th>sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta0</td>
<td>-5.631</td>
<td>0.0267</td>
<td>5.1E-4</td>
<td>-5.686</td>
<td>-5.63</td>
<td>-5.581</td>
<td>1000</td>
<td>20002</td>
</tr>
<tr>
<td>betaR</td>
<td>0.3147</td>
<td>0.05165</td>
<td>5.546E-4</td>
<td>0.2137</td>
<td>0.3147</td>
<td>0.4155</td>
<td>1000</td>
<td>20002</td>
</tr>
<tr>
<td>betaS</td>
<td>-0.9393</td>
<td>0.03023</td>
<td>3.176E-4</td>
<td>-0.9992</td>
<td>-0.9391</td>
<td>-0.8811</td>
<td>1000</td>
<td>20002</td>
</tr>
<tr>
<td>betaSR</td>
<td>-0.2305</td>
<td>0.09374</td>
<td>9.413E-4</td>
<td>-0.4185</td>
<td>-0.2291</td>
<td>-0.04894</td>
<td>1000</td>
<td>20002</td>
</tr>
<tr>
<td>sigma</td>
<td>0.1328</td>
<td>0.02858</td>
<td>8.246E-4</td>
<td>0.08105</td>
<td>0.1316</td>
<td>0.1927</td>
<td>1000</td>
<td>20002</td>
</tr>
<tr>
<td>theta</td>
<td>0.7976</td>
<td>0.07464</td>
<td>7.446E-4</td>
<td>0.658</td>
<td>0.7953</td>
<td>0.9522</td>
<td>1000</td>
<td>20002</td>
</tr>
</tbody>
</table>

- Based on 2 chains of 10,000 reps each
  - burnin of 1,000
- These are just the default statistics
  - could extract the samples themselves using ‘coda’
Gibbs sampling algorithm

**Gibbs sampler**

- Goal is to generate samples from the posterior distribution

\[ \pi(\theta | Y) \propto L(Y | \theta) \times \pi(\theta) \]

- In many problems, \( \theta \) consists of several parameters, often with very different properties
  
  * probabilities, means, variances, log-odds ratios, …
  * may be defined on \([0,1], \mathbb{R} \) and \( \mathbb{R}^+ \), …

- Can be challenging to generate draws from the full joint posterior distribution

- Suppose \( \theta \) can be divided into \( d \) components or subvectors
  
  * \( \theta = (\theta_1, \ldots, \theta_d) \)
Gibbs sampling algorithm

Gibbs sampler (cont)

- Gibbs sampling scheme is defined in terms of these subvectors
  - cycle the \(d\) subvectors, drawing each subset conditional on the value of the other \((d-1)\) subvectors
  - for the \(j^{th}\) subvector, generate a random draw from the conditional posterior
    \[
    \pi(\theta_j \mid \mathbf{Y}, \theta_{\neg j})
    \]
    where \(\theta_{\neg j} = (\theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_d)\)

- It is often the case that subvectors can be identified that yield full conditionals which are convenient to draw samples from
  - bivariate normal (see below)
  - hierarchical models
Gibbs sampling algorithm

Simple example

- Bivariate normal distribution with unknown mean

\[
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix} \sim \text{Normal} \left( \begin{pmatrix}
\theta_1 \\
\theta_2
\end{pmatrix}, \begin{pmatrix}
1 & \rho \\
\rho & 1
\end{pmatrix} \right)
\]

* assume \( \rho \) is known

- Single observation \( y = (y_1, y_2) \)

- With a uniform prior on \((\theta_1, \theta_2)\), the posterior is

\[
\begin{pmatrix}
\theta_1 \\
\theta_2
\end{pmatrix} \mid y \sim \text{Normal} \left( \begin{pmatrix}
y_1 \\
y_2
\end{pmatrix}, \begin{pmatrix}
1 & \rho \\
\rho & 1
\end{pmatrix} \right)
\]
Gibbs sampling algorithm

Simple example (cont)

- Algorithms exist for sampling directly from a multivariate normal distribution
- Consider an application of Gibbs sampling
- Require two full conditionals:
  \[
  \theta_1 | y, \theta_2 \sim \text{Normal}(y_1 + \rho(\theta_2 - y_2), 1 - \rho^2) \\
  \theta_2 | y, \theta_1 \sim \text{Normal}(y_2 + \rho(\theta_1 - y_1), 1 - \rho^2)
  \]
- In the following \( y = (0, 0) \) and \( \rho = 0.6 \)
Gibbs sampling algorithm

**Single run**

- Start at \((\theta_1^{(0)}, \theta_2^{(0)}) = (3, 3)\)
- 2 complete iterations
Gibbs sampling algorithm

Four runs

- Dispersed starting points
  - first 10 complete iterations for each
Gibbs sampling algorithm

Four runs

- Dispersed starting points
  - first 10 complete iterations for each

\[
\begin{align*}
\theta_1 & \quad \theta_2 \\
-3 & \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 3 \\
-3 & \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 3
\end{align*}
\]
Gibbs sampling algorithm

Four runs

- 2,500 iterations for each run
Notes

- Each iteration now consists of $d$ steps
  - can be inefficient, especially for highly correlated parameters
  - block-updating can sometimes be exploited and WinBUGS has some functionality

- If a full conditional is not of a convenient form, use a Metropolis update
  - Metropolis-within-Gibbs
  - chain no longer guaranteed to move with each iteration
Outline

- Goals and Scope
- Preliminaries
- The WinBUGS Software
  - Graphical models
  - WinBUGS syntax
  - Data
- Navigating the WinBUGS Menu System
- Additional Topics
The WinBUGS Software

**BUGS**

- **Bayesian inference Using Gibbs Sampling**
- WinBUGS runs under Windows 95/98/NT
  - provides a graphical interface to the BUGS language
  - jointly developed by Biostatistics Unit of the Medical Research Council, Cambridge and Imperial College School of Medicine, St. Mary’s, London
- Freely available for download from the WinBUGS website:
  
  http://www.mrc-bsu.cam.ac.uk/bugs/
  
  - key is required to unlock the educational version
- Versions of BUGS for a variety of other platforms
  - e.g., several approaches to running WinBUGS on a Mac
The WinBUGS Software

WinBUGS

- With WinBUGS you both make and use software
- Each analysis results in the creation of a new software system
  - you provide the plan
  - WinBUGS provides the building materials and tools
- Two user interfaces for the plan (i.e., model specification)
  - GUI
  - script language
- Planning/structure of the software built on principles of graphical models
  - graphical representations of Bayesian models
  - framework for calculating full conditional distributions, required for Gibbs sampling
WinBUGS Software

WinBUGS

- Modular design written in Component Pascal
  - consists of over 200 modules
  - each has a well-defined interface
  - consistency checks, across modules, are performed at compilation

- OpenBUGS permits development of new modules

http://mathstat.helsinki.fi/openbugs/
Graphical representations of models

Graphs as a formal language

- A graph is a set of *vertices* and *edges*
  - vertices are also called *nodes*

- The structure of a graph can be used to represent a wide range of systems

- Additional complexity can be incorporated by assigning attributes to the nodes and edges
Graphical representations of models

Bayesian systems

- Bayesian systems can be represented via graphs
  - structure of the graph is a representation of the joint posterior probability distribution

- Nodes represent
  - parameters
  - data

- Edges represent relationships
  - dependencies among data/parameters
  - all are ‘directed’

- Useful to distinguish ‘child’ and ‘parent’ nodes

- A good introduction to graphical models can be found in Fryback et al. (2001).
Linear regression with a single predictor

Mathematical representation

• Consider a simple linear regression

Stage I: Likelihood

\[ Y_i \sim_{iid} \text{Normal}(\mu_i, \tau^{-1}), \quad i = 1, \ldots, N \]
\[ \mu_i = \beta_0 + \beta_X X_i \]

Stage II: Priors

\[ \beta_0 \sim \text{Normal}(0, 1e^6) \]
\[ \beta_X \sim \text{Normal}(0, 1e^6) \]
\[ \tau \sim \text{Gamma}(0.5, 0.01) \]

• \( \tau \) is the precision (i.e., inverse variance)
  * interest may also lie in the standard deviation, \( \sigma = \frac{1}{\sqrt{\tau}} \)
Linear regression with a single predictor

Graphical representation
Linear regression with a single predictor

WinBUGS representation

```winbugs
model
{
    ## Model and likelihood
    for(i in 1:N)
    {
        Y[i] ~ dnorm(mu[i], tau)
        mu[i] <- beta0 + betaX*X[i]
    }

    ## Priors
    beta0 ~ dnorm(0.0, 1.0E-6)
    betaX ~ dnorm(0.0, 1.0E-6)
    tau ~ dgamma(0.5, 0.01)

    ## Additional parameter of interest
    sigma <- 1 / sqrt(tau)
}
```
Building blocks

Structures

- Three types of nodes
  - stochastic
  - logical
  - constant

- Two types of edges
  - single arrow
  - double arrow

- Plate
  - permits the concise representation of repetition
  - convenient for structured models
Building blocks

Stochastic nodes

- Stochastic nodes are data/parameters that are assigned a distribution
  - e.g., $Y_i$, $\beta_0$, $\beta_X$, and $\tau$
  - distribution $\rightarrow$ likelihood or prior

- Appear once on the LHS of the mathematical representation
  - accompanied by a ‘$\sim$’

- Appear once in the graphical representation
  - as an oval

- Appear once on the LHS in the WinBUGS representation
  - again, accompanied by a ‘$\sim$’
### Building blocks

<table>
<thead>
<tr>
<th>outline</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stochastic nodes (cont)</strong></td>
</tr>
<tr>
<td>- Each stochastic node in each representation requires</td>
</tr>
<tr>
<td>- a name</td>
</tr>
<tr>
<td>- a distribution</td>
</tr>
<tr>
<td>- specification of parameters for the distribution</td>
</tr>
<tr>
<td>- In a graph, these are referred to as ‘attributes’</td>
</tr>
</tbody>
</table>
Building blocks

Stochastic node $Y_i$
Building blocks

Stochastic node $\tau$

![Graphical Model](image)

Outline
- Goals and Scope
- Preliminaries
  - Lung cancer data
  - Gibbs sampling
- The WinBUGS Software
  - Graphical models
  - WinBUGS syntax
  - Data
- Navigating WinBUGS
  - Specification Tool
  - Update & Sample Monitor
  - Tools
  - Convergence and analysis
- Additional Topics
  - Errors and traps
  - Miscellaneous notes
Logical nodes

- Logical nodes are deterministic functions of other parameters/data
  - e.g., $\mu_i, \sigma$

- Appear once on the LHS of the mathematical representation
  - accompanied by an ‘=’

- Appear once in the graphical representation
  - as an oval, although the ‘attributes’ are different (see below)

- Appear once on the LHS in the WinBUGS representation
  - accompanied by a ‘<-’
Logical node $\mu_i$

Building blocks
Building blocks

Logical node $\sigma$

```
SimpleDoodle
name: sigma  type: logical  link: identity
value: 1 / sqrt(tau)
```

Diagram showing nodes and connections:

- $X[i] \rightarrow \mu[i] \rightarrow Y[i] \rightarrow \sigma$
- $\tau \rightarrow Y[i]$
- $\beta0 \rightarrow \mu[i] \rightarrow Y[i]$
- $\beta X \rightarrow \mu[i] \rightarrow Y[i]$

Outline
- Goals and Scope
- Preliminaries
  - Lung cancer data
  - Gibbs sampling
- The WinBUGS Software
  - Graphical models
    - WinBUGS syntax
    - Data
- Navigating WinBUGS
  - Specification Tool
  - Update & Sample Monitor
  - Tools
    - Convergence and analysis
- Additional Topics
  - Errors and traps
  - Miscellaneous notes
Building blocks

Logical nodes (nodes)

- Logical nodes require specification of
  - a name
  - a value

- ‘link’ option is to facilitate specification of generalized linear models
  - e.g., $\log$, $\logit$

- Logical node ‘deviance’ is automatically created and monitored
  - $-2 \times \log(\text{likelihood})$
  - contributes to the calculation of the Deviance Information Criterion (DIC);
  - developed as a measure of model comparison and criticism
Building blocks

Constants

- Constants are just that
  - include data such as $X_i$, $N$
  - fixed parameters, such as the variances in the normal priors

- Can either be incorporated as nodes (rectangles), or as part of the data

  - Have no ‘parents’ in the graph
    - not updated in the MCMC scheme
## Building blocks

### Edges

- Edges represent dependencies between nodes
  - nature of the dependency depends on the characteristics of the child node

- Single arrow denotes a stochastic dependence
  - child node is stochastic
  - parental node is a parameter for the *probability distribution* associated with the child node

- Double arrow denotes a deterministic dependence
  - child node is logical
  - parental node directly determines (at least in part) the *value* of the child node
Ohio lung cancer example

Three representations

- Consider the three representations of the Ohio lung data model
- Data consist of over 2 million records
  - aggregated (in R) according to county, sex and race
- County-specific index: $k = 1, \ldots, K$
  - $K = 88$
- Covariate (sex/race)-specific index: $j = 1, \ldots, J$
  - $J = 4$
Ohio lung cancer example

Mathematical representation

Stage I: Likelihood

\[ Y_{kj} \sim_{iid} \text{Binomial}(N_{kj}, \pi_{kj}), \ k = 1, \ldots, K, \ j = 1, \ldots, J \]

\[ \pi_{kj} = \beta_0 + \beta_S S_j + \beta_R R_j + \beta_{SR} S_j \ast R_j + V_k \]

\[ \theta = \exp\{\beta_{SR}\} \]

\[ V_k \sim \text{Normal}(0, \tau^{-1}) \]

\[ \sigma = \frac{1}{\sqrt{\tau}} \]

Stage II: Priors

\[ (\beta_0, \beta_S, \beta_R, \beta_{SR}) \sim_{iid} \text{Normal}(0, 1e^6) \]

\[ \tau \sim \text{Gamma}(0.5, 0.01) \]
Ohio lung cancer example

Graphical representation: Stochastic node $Y_{kj}$
Ohio lung cancer example

Graphical representation: Logical node $\pi_{kj}$
Ohio lung cancer example

WinBUGS representation

```plaintext
model
{
    ## Model and likelihood
    for(k in 1:K){
        V[k] ~ dnorm(0, tau)
        for(j in 1:J){
            Y[k,j] ~ dbin(p[k,j], N[k,j])
            logit(p[k,j]) <- beta0 + betaS*sex[j] + betaR*race[j]
            + betaSR*sex[j]*race[j] + V[k]
        }
    }

    ## Priors
    beta0 ~ dnorm(0.0, 1.0E-6)
    betaS ~ dnorm(0.0, 1.0E-6)
    betaR ~ dnorm(0.0, 1.0E-6)
    betaSR ~ dnorm(0.0, 1.0E-6)
    tau ~ dgamma(0.5, 0.01)

    ## Additional parameters of interest
    theta <- exp(betaSR)
    sigma <- 1 / sqrt(tau)
}
```
WinBUGS and graphical models

Doodles

- In WinBUGS, graphical representations of Bayesian systems are referred to as *Doodles*

- DoodleBUGS is a built-in utility for creating/editing Doodles
  * accessed via Doodle Menu

- The utility is well-documented in the WinBUGS manual
  * creation and manipulation of Doodles
WinBUGS and graphical models

Moving beyond graphical models

- Understanding graphical models is helpful in terms of understanding how WinBUGS works
  - calculate the full conditional distribution
  - examine each factor in the full conditional and classify its’ functional form
  - combine functional forms for each factor
  - choose suitable MCMC simulation algorithm

- For simple model, also provides a convenient means of communication

- Quickly become a cumbersome as models get more and more complex
WinBUGS language and syntax

WinBUGS script language

- Script, text-based language provides a second interface for model specification

- WinBUGS has functionality to convert a doodle into a script

- With experience, it becomes more convenient to write your own code
WinBUGS language and syntax

Building blocks

- Each representation consists of three main types of building blocks
- Nodes
  - stochastic/logical/constants
- Edges
  - stochastic/deterministic dependencies or relations
- Plates
  - characterize repetition via the use of indices
WinBUGS language and syntax

**Distributions**

- Stochastic nodes are characterized by a name and distribution
  - parameters associated with the distribution

- Assignment is via a ‘twiddle’
  \[
  Y \sim \text{dnorm}(\mu, \tau)
  \]

- Generally appear once and only once on the LHS of an expression
  - exception is when one performs a data transformation within WinBUGS (see later on)
Parameters that index distributions must be explicit nodes

* cannot be functional expressions
* a check of the syntax returns an error

```plaintext
model{
    ### Likelihood
    for(i in 1:N) {
        Y[i] ~ dnorm(beta0 + betaX * X[i], tau)
    }
    ### Priors
    beta0 ~ dnorm(0.0, 1.0E-6)
    betaX ~ dnorm(0.0, 1.0E-6)
    tau ~ dgamma(0.5, 0.01)
}
```

expected a comma
WinBUGS language and syntax

Distributions (cont)

- Broad range of distributional choices
  
  - Univariate
    * Bernoulli, binomial, categorical, negative binomial, poisson
    * beta, $\chi^2$, double exponential, exponential, gamma, generalized gamma, log-normal, logistic, normal, pareto, Student-t, uniform, Weibull
  
  - Multivariate
    * multinomial
    * Dirichlet, multivariate normal, multivariate Student-t, Wishart
## Distributions (cont)

- Should consult the WinBUGS manual for the specific parameterization
  - for example, the normal distribution is parameterized in terms of the *precision*, as opposed to the variance:

\[
Y \sim \text{dnorm}(\mu, \tau)
\]

has the density

\[
f_Y(y) = \sqrt{\frac{\tau}{2\pi}} \exp\left\{-\frac{\tau}{2}(y - \mu)^2\right\}
\]

- Additional caveats and restrictions mainly apply to multivariate distributions
  - e.g., restrictions on their use as priors

\[
\frac{\tau}{2}(y - \mu)^2
\]
WinBUGS language and syntax

<table>
<thead>
<tr>
<th>Functional expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Logical nodes are characterized by a name and a value</td>
</tr>
<tr>
<td>* value is determined by a logical expression, represented as a function of the parent nodes</td>
</tr>
<tr>
<td>• Assignment is via a left-pointing arrow</td>
</tr>
<tr>
<td>* for example,</td>
</tr>
<tr>
<td>mu[i] &lt;- beta0 + betaX[i]</td>
</tr>
<tr>
<td>sigma &lt;- 1 / sqrt(tau)</td>
</tr>
<tr>
<td>* similar to R</td>
</tr>
<tr>
<td>• Only appear once on the LHS of an expression</td>
</tr>
<tr>
<td>* data transformation exception (again, see later on)</td>
</tr>
</tbody>
</table>
Functional expressions (cont)

- Functional arguments can be
  - expressions, e
    - \( \exp(\beta) \), \( \exp(\beta_0 + \beta X) \)
    - \( \log(e) \)
    - \( \min(e_1, e_2) \)
    - \( \text{pow}(e_1, e_2) = e_1^{e_2} \)
  - vector-valued nodes, v
    - \( \text{mean()} \)
    - \( \text{sd}(v) \)
    - \( \text{inprod}(v_1, v_2) \)
  - scalar-values, s
    - \( \text{ranked}(v, s) \): returns the \( s^{th} \) smallest component of \( v \)
WinBUGS language and syntax

Functional expressions (cont)

- Extensive list of functions
  - details in the WinBUGS manual

- Generally, functions live on the RHS of the ‘<-’

- Certain functions can serve as link functions on the LHS
  - as in a generalized linear model
  - for example, the following are equivalent representations

```plaintext
logit(pi) <- beta0
pi <- exp(beta0) / (1 + exp(beta0))
```

- others include log, cloglog, and probit,
WinBUGS language and syntax

Functional expressions (cont)

- Logical nodes cannot be given data or initial values
  - they are determined, as functions of other quantities

- A consequence of this is that one cannot model observed data as a sum of two random variables

```r
model{
  Y <- U + V
  U ~ dnorm(muU, tauU)
  V ~ dnorm(muV, tauV)
}
```

```r
## data
list(Y = ...)
```
Data in WinBUGS

Data format

- Generally follows the S-Plus and R conventions
  - loaded as part of a list()
  - assignment of names via an ‘=’ symbol

- Vectors use the ‘c()’ format

```r
list(N = 6,
     Y = c(10.3, 4, 5, 7.4, 12.3, 11.9, 15.6),
     X = c(0, 0, 0, 1, 1, 1))
```
Data in WinBUGS

Data format (cont)

- Arrays use the `structure()` format

```r
list(N = 6,
    dMat = structure(.Data = c(10.3, 0,
                              4.5, 0,
                              7.4, 0,
                              12.3, 1,
                              11.9, 1,
                              15.6, 1),
    .Dim = c(6,2)))
```

- Notice how WinBUGS reads the vector into the matrix by filling the right-most index first
  - fills in the $6 \times 2$ matrix by *rows*
  - S-Plus and R fill it in by *columns*
Data in WinBUGS

Data format (cont)

- Data in arrays (or vectors) can also be represented in a rectangular format

\[
\begin{array}{cc}
Y[ ] & X[ ] \\
10.3 & 0 \\
4.5 & 0 \\
7.4 & 0 \\
12.3 & 1 \\
11.9 & 1 \\
15.6 & 1 \\
\end{array}
\]

END

* each column is headed by the vector name, each explicitly ending with brackets

- File must end with an ‘END’ statement
Data in WinBUGS

Data format (cont)

- Multi-dimensional arrays can be defined by explicit indexing

```
  dMat[,1]  dMat[,2]
   10.3     0
    4.5     0
    7.4     0
   12.3     1
   11.9     1
   15.6     1

  END
```

- First index position for the array name should always be empty
  * including vectors (previous slide)
## Data format (cont)

- Can use a mixture of the `list()` and rectangular formats
  - matter of convenience
  - see the Rats example in the WinBUGS manual

- Missing data are represented as `NA`

- Various resources exist for converting data from other packages into a format conducive to WinBUGS
  - e.g., S-plus, R, SAS, Excel
  - WinBUGS website has a comprehensive list under
    ```bash
    /weblinks/webresource.shtml
    ```
Unbalanced data

- Repeated measures or longitudinal studies often have differing numbers of observations per cluster
  - community-based study of breast cancer screening
  - different radiologists interpret varying numbers of mammograms over the course of a year

- Resulting data form a ‘ragged array’
  - cluster 1: 13.2 16.5
  - cluster 2: 12.1
  - cluster 3: 15.5 9.9 6.6
  - ...

- Several ways to handle this
  - 1. fill in the ‘missing’ data
  - 2. nested indexing
Data in WinBUGS

**Unbalanced data: Option 1**

- Suppose there are 100 clusters, the largest of which is of size 10
  
  ```
  model{
    for(i in 1:N){
      for(j in 1:nMax){
        Y[i,j] ~ dnorm(mu[i], tau)
      }
      mu[i] ~ dnorm(0, 1.0E-6)
    }
    tau ~ dgamma(0.5, 0.01)
  }
  ```

  ```
  ## data
  list(N = 100, nMax = 10,
       Y = structure(.Data = c(13.2, 16.5, NA, NA, ...),
                     .Dim=c(100, 10)))
  ```

- Can be slow
  
  * missing are treated as unknown parameters
  * require updating in the MCMC scheme
Data in WinBUGS

### Unbalanced data: Option 2

- An alternative is to consider nested indexing

- For each observation provide an index for its cluster membership

  \[
  \begin{array}{ll}
  Y & \text{cluster} \\
  13.2 & 1 \\
  16.5 & 1 \\
  12.1 & 2 \\
  15.5 & 3 \\
  9.9 & 3 \\
  6.6 & 3 \\
  \ldots & \ldots \\
  \end{array}
  \]

- Only required to loop through the observed data
  - rather than the $100 \times 10$ observed + unobserved elements
Unbalanced data: Option 2

- Suppose there are 567 observations from the 100 clusters

```r
model{
  for(j in 1:nTot){
    Y[j] ~ dnorm(mu[cluster[j]], tau)
  }
  for(i in 1:N){
    mu[i] ~ dnorm(0, 1.0E-6)
  }
  tau ~ dgamma(0.5, 0.01)
}
```

```r
## data
list(N = 100, nTot = 567, 
    Y = c(13.2, 16.5, 12.1, ...),
    cluster = c(1, 1, 2, ...))
```
## Data in WinBUGS

### Data transformation

- Data manipulations are best done prior to using WinBUGS.

- Might be interested in comparing models with varying transformations of the outcome variable:

  ```
  for(i in 1:N){
    z[i] <- sqrt(y[i])
    z[i] ~ dnorm(mu, tau)
  }
  ```

  - WinBUGS has built-in checks to permit this type of declaration.

- May be useful to place the transformation at the beginning of the script:

  - separate transformation from main model specification.
Outline

- Goals and Scope
- Preliminaries
- The WinBUGS Software
  - Navigating the WinBUGS Menu System
    * Specification Tool
    * Update & Sample Monitor Tools
    * Convergence and analysis
- Additional Topics
Navigating WinBUGS

WinBUGS menu system

- Extensive menu system designed to facilitate and compliment the analysis

- Two key pull-down menu lists are
  - Model Menu
  - Inference Menu

- We’ll focus on these

- Others that may be of immediate interest include:
  - Info Menu; status/error log, node information
  - Options Menu; how output is returned, blocking and updating options
Navigating WinBUGS

Model Menu

- Provides access to the Specification Tool window
  - WinBUGS reads in model, data and starting values
  - compiles the code to ‘create’ new software

WinBUGS syntax

```plaintext
## Model specification
model {
  ## Model and likelihood
  for(k in 1 K) {
    Y[k] ~ dbin(n[k], N[k])
    logit(n[k]) <- beta0 + betaA* race[k] + betaR* sex[k]* race[k]
  }

  ## Priors
  beta0 ~ dnorm(0.0, 1.0E-5)
  betaA ~ dnorm(0.0, 1.0E-5)
  betaA2 ~ dnorm(0.0, 1.0E-5)
  betaS ~ dnorm(0.0, 1.0E-5)
  betaR ~ dnorm(0.0, 1.0E-6)
  betaSR ~ dnorm(0.0, 1.0E-6)
}
```
Navigating WinBUGS

 Specification Tool

- Position the cursor anywhere within the curly \{\} brackets and press ‘check model’
  - message bar reports on any syntactic problems
Navigating WinBUGS

Specification Tool

- To enter in data, highlight any part of the word ‘list’ and press ‘load data’
  - Highlighting beyond the word ‘list’ generates an error

![WinBUGS Interface](image)
Navigating WinBUGS

Specification Tool

- Enter the number of chains you want to run and press ‘compile’
  * builds the structures required to carry out the Gibbs sampling
  * model is checked for completeness and consistency
Navigating WinBUGS

Specification Tool

- Highlight the word ‘list’ and press ‘load inits’ to establish starting values for each chain
  - ‘gen inits’ will use the prior distribution (or an approximation) for any parameters not already provided
Navigating WinBUGS

Specification Tool

- Once all the components have been correctly loaded, the message ‘model is initialized’ should appear.
Navigating WinBUGS

Inference Menu

- After initialization, we need to tell WinBUGS which parameters we want to monitor
  * ‘Samples...’ provides access to the ‘Sample Monitor Tool’ window

![Image of WinBUGS interface showing model specification and sample monitor tool]

In the WinBUGS interface, you can see the model specification panel with the model code:

```plaintext
# Model specification
model
{
  # Model for(k)
  Y[k] ~ dnorm(beta[0] + beta[1]*age[k], tau)
  beta[0] ~ dnorm(0, 1)
  beta[1] ~ dnorm(0, 1)
  tau ~ dunif(0, 1000)
}
```

To monitor specific parameters, you can use the 'Samples...' option in the 'Sample Monitor Tool' window. This allows you to select which parameters you want to monitor during the MCMC sampling process.
Navigating WinBUGS

Sample Monitor Tool

- Enter the name of each node and press ‘set’
  - if a node is an array, all components will be monitored

![Sample Monitor Tool](image)

```
## Model specification
model
{
## Priors
for(k in 1:8)
Y[k] ~ dnorm(y[k],tau)
betaR[k] ~ dnorm(0,tauBetaR)

## Regression
for(k in 1:8)
log(beta[k]) <- beta0 + betaR[k] * age[k]
Y[k] <- exp(beta[k])
}
```
Navigating WinBUGS

Sample Monitor Tool

- Check which nodes have been entered
Navigating WinBUGS

Model Menu

- ‘Update...’ becomes active once the model has been compiled and initialized
  * provides access to the ‘Update Tool’ window
Navigating WinBUGS

Update Tool

- Enter the number of MCMC updates and press ‘update’
  - ‘thin’ = \( k \) indicates that every \( k^{th} \) sample is stored

BEGIN WINBUGS CODE

```
## Model specification
model
{
  ## Model and likelihood
  for(k in 1:K){
    Y[k] ~ dbin(p[k], N[k])
    logit(p[k]) <- beta0 + beta1*equals(age[k], 9) + beta2*equals(age[k], 10)
    betaR*race[k] + beta
  }
  ## Priors
  beta0 ~ dnorm(0, 10^2)
  beta1 ~ dnorm(0, 10^2)
  beta2 ~ dnorm(0, 10^2)

  update: 11000
  refresh: 100
  iteration: 8800
  thin: 1
  over relax: False
  adapting: False
```

END WINBUGS CODE
Navigating WinBUGS

Update Tool

- Progress is reported in the ‘iteration’ box
  - ‘refresh’ lets you control how often the screen is updated
  - WinBUGS lets you know how long it took

```
## Model specification
model{
  ## Model and likelihood
  for(k in 1:K){
    Y[k] ~ dbin(p[k], N[k])
    logit(p[k]) <- beta0 + betaA1*equals(age[k], 9) + betaA2*equals(age[k], 10)
    betaR*race[k] + betaR
  }

  ## Priors
  beta0 ~ dnorm(0, sigma2)
  betaA1 ~ dnorm(0, sigma2)
  betaA2 ~ dnorm(0, sigma2)
}
```

Update Tool

- Progress is reported in the ‘iteration’ box
- ‘refresh’ lets you control how often the screen is updated
- WinBUGS lets you know how long it took

Updates took 12 s
Navigating WinBUGS

Convergence and analysis

- ‘Sample Monitor Tool’ window provides functionality for assessing convergence of the MCMC scheme and analysis of the posterior samples
  - all nodes are updated
  - only those entered prior to updating are stored

- Choose a specific node by entering the name or by accessing the drop-down list
  - take all nodes by entering a ‘*’
Navigating WinBUGS

Convergence diagnostics

- ‘history’ generates a time series-type plot
  - chains separated by color

- Others include
  - ‘trace’ provides a dynamic version
  - ‘quantiles’ plots the running mean, with running 95% confidence intervals
  - ‘auto cor’ plots the autocorrelation function out to lag-50
Navigating WinBUGS

Convergence diagnostics

- ‘bgr diag’ calculates the Gelman-Rubin convergence statistic
  - Brooks and Gelman (1998) modification

![Gelman Rubin statistic graph](image)
Navigating WinBUGS

Convergence diagnostics

- Consideration of the posterior central 80% interval for a given parameter
  - *green line*: width of central 80% interval of the *pooled* runs
  - *blue line*: average width of the 80% intervals *within* the individual runs
  - *red line*: their ratio $R = \text{pooled}/\text{within}$

- Original statistic based on the posterior variance

- For plotting purposes, pooled and within interval widths are normalized to have a maximum of 1

- Looking for
  - $R > 1$ at the outset indicates suitably over-dispersed starting points
  - stability in the pooled and within interval estimates
  - convergence to $R = 1$
Navigating WinBUGS

WinBUGS graphics

- Note that the plot two slides ago was modified from what was automatically generated by WinBUGS
  - ‘right-click’ to open up a ‘Plot Properties’ window

![Gelman Rubin statistic plot](image)
Navigating WinBUGS

Posterior summaries

- Pressing ‘stat’ returns posterior summary statistics
  - pooled over the selected chains
  - ‘MC error’ is the Monte Carlo standard error of the mean

<table>
<thead>
<tr>
<th>node</th>
<th>mean</th>
<th>sd</th>
<th>MC error 2.5%</th>
<th>median</th>
<th>97.5%</th>
<th>start</th>
<th>sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta0</td>
<td>-5.97</td>
<td>0.0262</td>
<td>4.338E-4</td>
<td>-5.97</td>
<td>-5.921</td>
<td>1</td>
<td>22000</td>
</tr>
<tr>
<td>betaA1</td>
<td>0.5552</td>
<td>0.03142</td>
<td>4.866E-4</td>
<td>0.5343</td>
<td>0.5951</td>
<td>1</td>
<td>22000</td>
</tr>
<tr>
<td>betaA2</td>
<td>0.567</td>
<td>0.03716</td>
<td>4.629E-4</td>
<td>0.6158</td>
<td>0.6668</td>
<td>1</td>
<td>22000</td>
</tr>
<tr>
<td>betaR</td>
<td>0.3496</td>
<td>0.05039</td>
<td>5.238E-4</td>
<td>0.2494</td>
<td>0.3502</td>
<td>1</td>
<td>22000</td>
</tr>
<tr>
<td>betaS</td>
<td>-0.9759</td>
<td>0.03650</td>
<td>3.327E-4</td>
<td>-1.04</td>
<td>-0.9796</td>
<td>1</td>
<td>22000</td>
</tr>
</tbody>
</table>

- ‘density’ returns a smoothed kernel density estimate for continuous nodes and a histogram for discrete nodes
  - not particularly good looking though
  - may want a little more graphical sophistication for journal articles or presentations
Navigating WinBUGS

CODA

- Move beyond the diagnostic and graphical options of WinBUGS
- Press to ‘coda’ dump out an ascii representation of the samples
  * coda: Convergence Diagnostics and Output Analysis
CODA

- Generates separate files for each chain, as well as a file indicating the indices for each node
  - save each as a ‘.txt’ file from the File Menu

- Exploit software specifically written to assess convergence and generate interesting figures
  - e.g., coda, an S-Plus and R package written by Martyn Plummer and colleagues

Outline
- Goals and Scope
- Preliminaries
  - Lung cancer data
  - Gibbs sampling
- The WinBUGS Software
  - Graphical models
  - WinBUGS syntax
  - Data
- Navigating WinBUGS
  - Specification Tool
  - Update & Sample Monitor
  - Tools
- Convergence and analysis
- Additional Topics
  - Errors and traps
  - Miscellaneous notes
Outline

- Goals and Scope
- Preliminaries
- The WinBUGS Software
- Navigating the WinBUGS Menu System

Additional Topics

* Errors and traps
* Miscellaneous notes
Errors and traps

Error messages

- If an error is encountered, a message is displayed at the bottom of the WinBUGS window
  - if it can, WinBUGS will place the cursor somewhere after the offending location
  - change the preferences to make the cursor more visible
  - also recorded in a log which can be accessed via the Info Menu

- WinBUGS error and trap messages are fairly notorious
  - some are self-explanatory
  - many are not

- Worth taking into context the breadth of models that WinBUGS can accommodate
  - given the flexibility of WinBUGS, it’s challenging thinking up all the ways in which an error can be generated
Errors and traps

Error messages (cont)

- Errors can be the result of improper syntax
  - Use of an ‘=’ sign

![WinBUGS GUI](image-url)
Errors and traps

Error messages (cont)

- Others become evident at compilation

- For example, consider attempting to model observed data as a sum of two random variables
  * syntactically OK

```
model{
  // Likelihood
  for(i in 1:N){
    Y[i] ~ U[i] + V[i]
    U[i] ~ dnorm(muU, tauU)
    V[i] ~ dnorm(muV, tauV)
  }

  // Priors
  muU ~ dnorm(0, 1.0E-6)
  tauU ~ dgamma(0.5, 0.01)
  muV ~ dnorm(0, 1.0E-6)
  tauV ~ dgamma(0.5, 0.01)
}
```

```
## Data
list(N = 5, Y = c(1, 2, 3, 4, 5))
```
Errors and traps

Error messages (cont)

• Error at compilation

![WinBUGS interface with code and error message](image)
Errors and traps

Error messages (cont)

- Other, similar errors include
  - ‘undefined variable’; variable is in the data but not defined in the model
  - note that WinBUGS is case-sensitive
  - ‘index out of range’

- Limitations and restrictions
  - ‘logical expression too complex’; too many parameters/constants or too many operators
  - linear predictors can take advantage of the `inprod(,)` function

- WinBUGS manual has greater detail

- Debugging can often take longer than actually performing the MCMC.
  - phenomenon not restricted to WinBUGS though!
## Errors and traps

### Trap messages

- Traps that occur during the MCMC process can be particularly difficult to diagnose
  - *‘undefined real result’* indicates numerical overflow
  - can happen if all the initial values are equal

- Can sometimes restart the simulation by pressing the ‘update’ button twice.

- If this doesn’t work, possible solutions include
  - better initial values
  - more informative priors
  - re-parameterization to improve orthogonality
  - standardization of covariates to have mean 0 and standard deviation 1
Miscellaneous notes

Specifying new distributions

- In some instances one might need to specify a distribution not yet implemented
  - e.g., a truncated distribution

- ‘ones’ trick for specifying a new sampling distribution

```r
C <- 10000  # ensure all p[i]’s < 1
for(i in 1:N){
  ones ~ dbern(pi[i])
  p[i] <- L[i] / C
  ones[i] <- 1
}
```

  - `L[i]` is the likelihood contribution for the new sampling distribution
  - function of observed data and parameters of interest

- WinBUGS manual also has tricks for specifying a new prior distribution as well as a discrete prior on a set of values
Miscellaneous notes

Compound documents

- WinBUGS allows documents to be embedded in a *compound document*

- Convenient way to
  - save a doodle, the code and data
  - store output, including posterior summaries and graphics

- All in a single document
  - helpful for keeping a track of analyses

- The use of ‘folds’ is helpful in making the document concise
  - ‘hide’ sections of the document (including graphics), to be revealed when needed.
  - WinBUGS manual has documentation
Gibbs sampling algorithm

Choice of sampling method

- The Gibbs sampling algorithm is the underlying framework for MCMC within WinBUGS
  - cycling through a series of full conditional posterior distributions

- When WinBUGS can identify a full conditional as being a known distribution, it will sample directly
  - modules implemented for a range of distributions

- When the full conditional is not of a known form, WinBUGS employs a Metropolis step
  - Metropolis-within-Gibbs
Gibbs sampling algorithm

Choice of sampling method (cont)

- Towards this, WinBUGS has a variety of MCMC methods at its disposal
  - vary according to characteristics of the parameter, and its’ posterior distribution

- Continuous $\theta$
  - **Distribution**
    - Conjugate
    - Log-concave
    - Restricted range
    - Unrestricted range
  - **Method**
    - Direct sampling
    - Derivative-free adaptive rejection sampling
    - Slice sampling
    - Current-point Metropolis

- Discrete $\theta$
  - **Distribution**
    - Finite upper bound
    - Shifted Poisson
  - **Method**
    - Inversion
    - Direct sampling
Gibbs sampling algorithm

Choice of sampling method (cont)

- WinBUGS tries increasingly complex methods, until one works
  - if it is unable to identify a method, an error message is returned
  - introduction to the WinBUGS manual has more detail

* introduction to the WinBUGS manual has more detail
Miscellaneous notes

Software resources

● Add-ons specifically written for WinBUGS
  * GeoBUGS for spatial analyses
  * PKBUGS for pharmacokinetic analyses

● Packages written for other software programs
  * facilitate exporting data into a WinBUGS format
  * convergence and analysis (e.g., coda, boa)
  * setting up simulations/batches

● WinBUGS website
### Miscellaneous notes

#### Mac OS X

- It is possible to directly run WinBUGS on a Mac
  - OS X 10.4 or higher
  - Intel-based architecture

- Requires installation of
  - X11 windowing system
  - Darwine; a pre-compiled version is available at [http://www.kronenberg.org/darwine/](http://www.kronenberg.org/darwine/)

- Download the zip archive installation of WinBUGS

- More detail is available at [http://idiom.ucsd.edu/~rlevy/winbugsonmacosx.pdf](http://idiom.ucsd.edu/~rlevy/winbugsonmacosx.pdf)
WinBUGS
UBC Summer School

Outline
Goals and Scope
Preliminaries
Lung cancer data
Gibbs sampling
The WinBUGS Software
Graphical models
WinBUGS syntax
Data
Navigating WinBUGS
Specification Tool
Update & Sample Monitor Tools
Convergence and analysis
Additional Topics
Errors and traps
Miscellaneous notes

Miscellaneous notes

Mac OS X screenshot
**Miscellaneous notes**

**Additional references and resources**

- Good introductory papers

- WinBUGS manual is very comprehensive and worth reading
  - doesn’t provide guidance on Bayesian analyses

- WinBUGS website has links to a huge number of resources
  - additional worked examples
  - slides for other short courses/presentations/classes
  - software