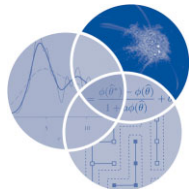
Overview



WinBUGS: a tutorial

Anastasia Lykou[1] and Ioannis Ntzoufras[2,*]

The reinvention of Markov chain Monte Carlo (MCMC) methods and their implementation within the Bayesian framework in the early 1990s has established the Bayesian approach as one of the standard methods within the applied quantitative sciences. Their extensive use in complex real life problems has lead to the increased demand for a friendly and easily accessible software, which implements Bayesian models by exploiting the possibilities provided by MCMC algorithms. WinBUGS is the software that covers this increased need. It is the Windows version of BUGS (Bayesian inference using Gibbs sampling) package appeared in the mid-1990s. It is a free and a relatively easy tool that estimates the posterior distribution of any parameter of interest in complicated Bayesian models. In this article, we present an overview of the basic features of WinBUGS, including information for the model and prior specification, the code and its compilation, and the analysis and the interpretation of the MCMC output. Some simple examples and the Bayesian implementation of the Lasso are illustrated in detail. © 2011 John Wiley & Sons, Inc. *WIREs Comp Stat* 2011 3 385–396 DOI: 10.1002/wics.176

HISTORICAL BACKGROUND

The BUGS (Bayesian inference using Gibbs sampling) project was introduced in 1989 by the research group of David Spiegelhalter in the MRC Biostatistics Unit as a software that uses Markov chain Monte Carlo (MCMC) methods in complex Bayesian methods. BUGS was initially a software for DOS and Linux operating systems limited in specialized algorithms. The project started expanding after it moved to the Imperial College in 1996 headed by Nicky Best, and the first version of WinBUGS for Windows became available in 1997. In the following years, WinBUGS was improved and extended by considering more complicated model structures. Later when Jon Wakefield and Dave Lunn joined the group, WinBUGS capabilities improved impressively. The latest version of WinBUGS (1.4.3) was developed jointly by Imperial College and the School of Medicine at St Mary's, London. More details can be found in Ref 1 and in the web page of OpenBUGS (http://www.openbugs.info).

WinBUGS has become widely popular over the last years as it can estimate the posterior distributions of the parameters of interest in a variety of models using MCMC. It only requires to specify the model code in which the model likelihood and the prior distribution are defined. The programming syntax and requirements are similar to the ones in R and Splus. For those who are not familiar with programming, there is also DOODLE, a graphical interface in which we can define our model via drawing the corresponding directed acyclic graph (DAG). Moreover, WinBUGS is freely available and this is one of the reasons that contributed in its growing popularity. A detailed manual,[2] as well as three volumes of examples[3–5] that provide important help and insight to the WinBUGS user, is available in the BUGS project website (www.mrc-bsu.cam.ac.uk). A comprehensive introduction in Bayesian modeling using WinBUGS is also offered by Ntzoufras,[6] in which emphasis is given on model building, implementation using WinBUGS, and the interpretation and analysis of the posterior results.

The purpose of this article is to provide a comprehensive short tutorial by summarizing the most important features of WinBUGS. It provides information on how to code and compile a model through two simple examples and a more detailed one that illustrates the implementation of Bayesian Lasso method in WinBUGS. The remainder of this article is organized as follows. Section '*Getting Started with Winbugs*' contains a brief description of the menu bar and detailed illustration of the model specification, the

*Correspondence to: ntzoufra@aueb.gr

[1]Department of Mathematics and Statistics, Lancaster University, Lancaster, UK

[2]Department of Statistics, Athens University of Economics and Business, Athens, Greece

DOI: 10.1002/wics.176

 © 2011 John Wiley & Sons, Inc.

data and the initial values definition using two simple examples. Section '*Running an MCMC in Winbugs and Obtaining Posterior Summaries*' provides a condensed list of the actions needed to run the MCMC algorithm for a model and how to obtain and analyze the corresponding output. An example of implementing the Bayesian Lasso in WinBUGS can be found in the Section '*An Illustrating Example—Implementing Bayesian Lasso in Winbugs*'. The article closes with a short discussion concerning the future potentials of WinBUGS followed by a short conclusion.

GETTING STARTED WITH WINBUGS

The Menu Bar

The latest version of WinBUGS (1.4.3) as well as installation instructions and the free key for unrestricted use can be found on the software's website: www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml. Once the installation process has been completed, WinBUGS can be assessed by double-clicking its shortcut. All main operations are available in a menu bar, which is similar to the ones found at any windows-based program.

The basic operations in the menu bar are the `File`, `Window`, and `Help` menus. The `File` menu handles the usual file actions that are available in any windows-based software (e.g., `Open, Close, Save, Print`), the `Window` manages active windows in WinBUGS and the `Help` provides access to the detailed manual and examples of WinBUGS which are very useful for the user.

The `Tools`, `Edit`, `Attributes`, and `Text` menus refer to editing facilities of the documents in WinBUGS. To be more specific, the `Edit` menu allows the user to do the usual editing actions of any word-processing software (e.g., `Copy, Cut, Paste`), while the `Tools` menu manages more specialized actions available for WinBUGS compound documents (`insert dates, encode, and decode algorithms`). The WinBUGS user can change the color, the font type, and size of the text in a compound document using the `Attributes` menu, while he/she can find or replace a text, insert a ruler, a paragraph, or blank spaces (amongst other operations) in the `Text` menu.

The most substantial WinBUGS operations are included in the `Model` and `Inference` menus. They include all the commands and actions related to running the MCMC algorithm and analyzing its output in order to obtain posterior results.

The `Specification` tool, under the `Model` menu, is used to compile the model and initialize the MCMC algorithm; the `Update` tool generates random observations from the posterior distribution and the `Monitor Met` tool checks the acceptance rate of the Metropolis-Hasting algorithm (in cases it is used). The last set of generated values can be saved using the `Save State` tool. This set of values can be used as initial values in cases that we want to rerun the MCMC algorithm from the point that a previous MCMC run stopped. The `Inference` menu provides information about the posterior distribution of the model parameters. This information is based on the analysis of the MCMC output. Under this menu, the most frequently used tool is the `Samples` tool, which provides basic descriptive summaries and graphical representation for specific attributes of the MCMC output and the corresponding estimated posterior distribution.

The `Info` and `Options` menus are offering some auxiliary tools for MCMC analysis. From the former menu, we can open or clear a log window (where WinBUGS results and figures are printed) or we can extract the current parameter values using the `Node info` tool. From the latter menu, we can change some options concerning the output, the blocking and the generation algorithm used to update each parameter.

Finally, the `Map` and the `Doodle` menus are offering more specialized tools for the user. The `Map` menu corresponds to the GeoBUGS add-in module and it can be used for spatial modeling and mapping. The `Doodle` menu is used to construct the DAG that describes the conditional dependencies of the Bayesian model we wish to fit. This tool is essential for those who are not familiar with programming as the model code can be generated from this graphical representation of the model. Nevertheless, it assumes very good knowledge and understanding of Bayesian models and how they can be represented in terms of conditional distributions and hierarchies.

How to Code a Model

WinBUGS uses its own type of input and output files that are called compound documents and are saved with the odc suffix. The first step in WinBUGS is to specify the model, which includes the likelihood function for the observed sample and the prior information for the parameters. The model code is written in a compound file within the syntax

```
model { ..... } .
```

There are three categories for the model parameters (or nodes): the constant, the stochastic, or random and the logical. The constant nodes are fixed values while the stochastic are random variables, such as the data and the model parameters. The stochastic nodes follow a distribution, which can be

 © 2011 John Wiley & Sons, Inc.

specified by using commands similar with the ones used in R and Splus packages. The logical nodes are mathematical expressions of other (constant or stochastic) components.

Example 1

Let's assume that a part of a Bayesian model includes a normally distributed variable $X \sim \text{Normal}(\mu, \sigma^2 = 1/\tau)$, with known mean $\mu = 0$ and unknown precision τ. If the uncertainty about the precision is expressed by considering the Gamma prior distribution $\tau \sim \text{Gamma}(0.01, 0.01)$, this can be expressed in WinBUGS as follows:

```
model{
        mu <- 0
        a<-0.01
        b<-0.01
        X ~ dnorm( mu, tau )
        tau ~ dgamma( a, b )
        sigma2 <- 1/tau   # sigma2:
             variance of the normal
                distribution
}
```

In the above syntax, the sign # is used to add comments, ~ is used to specify that a random node follows a distribution and <- is used to define assignments for constant and logical nodes. Nodes `mu, a, b` are constants here, `X` and `tau` are stochastic components and `sigma2` is a logical component. The commands `dnorm( mu, tau)` and `dgamma(a,b)` are used to specify the normal (with mean `mu` and variance `1/tau`) and the gamma (with mean `a/b`) distributions respectively. A list with all the distributions available in WinBUGS can be found at the software's manual[2] and in Ref 6, p. 90–91. Each component/node must be uniquely defined in the WinBUGS syntax.

Vectors, matrices, and arrays can be represented in WinBUGS in a similar way as in R/Splus packages. In the example above, the random variable X is a scalar. If X is a n-dimensional random vector this is denoted in WinBUGS by `X[]`. Syntax `X[i]` refers to the i-th element of vector $\boldsymbol{X}$ for $i = 1, \ldots, n$, whereas, `X[i:j]` extracts a vector with components the i-th up to the j-th element of $\boldsymbol{X}$. If M is a matrix this is specified in WinBUGS by `M[]`. If M is a $n \times p$ matrix then `M[i,j]` corresponds the m_{ij} element of M for any $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, p\}$. The entire i-th row can be extracted by typing `M[i,]` while the j-th column by `M[,j]`. The sub-matrix that contains all elements m_{ij} with $i_1 \leq i \leq i_2$ and $j_1 \leq j \leq j_2$ can be extracted using the syntax `M[i1:i2,j1:j2]`. A three-dimensional array A is denoted by `A[ , , ]` and the item `A[i,j,k]` correspond to the A_{ijk} element. Arrays of lower dimensions or parts of the initial array can be derived using syntax similar to the one used for the matrices; further details can be found in Section 3.3.2 of Ref 6.

Calculations among vectors, matrices, and arrays cannot be performed directly in WinBUGS. The following calculation $x^T y$ between the vectors x and y can be performed using the function `inprod( x[], y[] )` and the multiplication between two matrices A and B of dimension $n \times k$ and $k \times m$ can be performed using the `inprod` for all the rows and columns. Thus, the use of the function `for` is required to define a loop among the rows and the columns, and the multiplication is performed as follows:

```
for (i in 1:n){
    for (j in 1:m){
              C[i,j] <- inprod
                     (A[i,],B[,j])
    }
}
```

A set of built-in functions are available within WinBUGS including arithmetic functions such as the absolute value (`abs`), the exponential (`exp`), the natural logarithm (`log`), the square root (`sqrt`), and the statistical functions such as the sum (`sum`) the standard deviation (`sd`) and the rank (`rank`). A list with the functions that can be used in WinBUGS can be found on the software's manual.[2]

Model Specification

Suppose that n realizations are available for the response variable $\boldsymbol{y}$, which follows a distribution with parameter vector $\boldsymbol{\theta}$. Assume that the parameter $\boldsymbol{\theta}$ is related with the explanatory variables $X_1, X_2, \ldots, X_p$ through a link function h and a parameter vector $\boldsymbol{\beta}$. The prior information for $\boldsymbol{\beta}$ is expressed by a known distribution. This model is specified in WinBUGS with the following syntax.

```
# Likelihood
for (i in 1:n){
    y[i] ~ distribution.name(theta)
}
# Link function
theta <-  [function of beta
         and X's]
# prior distribution
beta ~ distribution.name( ... )
```

If the parameters $\boldsymbol{\beta}, \boldsymbol{\theta}$ have dimension higher than one, a `for` loop is needed to specify them.

 © 2011 John Wiley & Sons, Inc.

Example 2

Consider a normal linear regression model with the response vector $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ following the normal distribution and x_{i1}, x_{i2} the realizations of 2 explanatory variables X_1, X_2 for $i = 1, \ldots, n$ individuals. The Bayesian model is formulated as

$$Y_i \sim N(\mu_i, \sigma^2) \quad \text{for} \quad i = 1, \ldots, p$$

and the parameter μ_i is a linear combination of the explanatory variables,

$$\mu_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}.$$

The Normal distribution is chosen to express the prior information for $\boldsymbol{\beta}$ and the Gamma distribution for the precision $\tau = \sigma^{-2}$. Hence,

$$\beta_i \sim \text{Normal}(0, 10^4) \quad \text{for} \quad i = 0, 1, 2,$$
$$\tau \sim \text{Gamma}(0.01, 0.01).$$

The syntax of the above regression model follows.

```
model{
   # definition of the likelihood
            function
   for (i in 1:n){
       y[i] ~ dnorm( mu[i], tau )
       mu[i] <- beta0 + beta1*
            x1[i] + beta2 * x2[i]
    }
   # prior distributions
   beta0 ~ dnorm( 0.0 , 1.0E-4 )
   beta1 ~ dnorm( 0.0 , 1.0E-4 )
   beta2 ~ dnorm( 0.0 , 1.0E-4 )
   tau   ~ dgamma( 0.01, 0.01 )
   # other parameters of interest
   s2 <- 1/tau
   s  <- sqrt(s2)
}
```

This syntax can be written in a more general way in order to cover the case with p explanatory variables. This can be achieved by introducing the parameter vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ and the $n \times (p+1)$ data matrix $\boldsymbol{X} = (\mathbf{1}_n, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_p)$, where $\mathbf{1}_n$ is a vector of length n with all elements equal to one and $\boldsymbol{X}_j$ $(j = 1, \ldots, p)$ is the vector of length n, with elements the observed values of covariate X_j. The syntax can be now written as follows:

```
model{
    # definition of the likelihood
              function
    for (i in 1:n){
        y[i] ~ dnorm( mu[i], tau )
        mu[i] <- inprod(X[i,]) ,
                    beta[] )
    }
    # prior distributions
    for (j in 1:p+1){
        beta[j] ~ dnorm( 0.0 ,
                1.0E-4 )
    }
    tau   ~ dgamma( 0.01, 0.01 )
    # other parameters of interest
    s2 <- 1/tau
    s  <- sqrt(s2)
}
```

□

Data and Initial Value Specification

To complete the model specification in WinBUGS, we need to insert the data and provide some initial values for the MCMC algorithm.

The initial values are used as a starting point in the iterative procedure of the MCMC algorithm. Initial values are needed for all the model parameters, i.e., stochastic components of the model that are not data and for which prior distributions are imposed. The option of random generation of a part or all the initial values is available in WinBUGS but it is advised to be avoided as it may lead to bad starting points entailing slow mixing of the algorithm or even numerical errors such as overflows.

The syntax used to initialize values for the model in Example 2 is

```
# INITS
  list( beta=0, beta1=0, beta2=0,
          tau=1 )
```

The data can be written using the `list` syntax which is similar to the corresponding one used in `R/Splus`. For example, the data of the Example 2 with 2 explanatory variables of $n = 5$ observations can be specified as follows:

```
# DATA
  list(n=5,
       x1=c(1,2,3,4,5),
       x2=c(23,57,59,14,36)
  )
```

Alternatively, the data can be specified in matrix form as follows:

```
# DATA
  list(n=5, p=2,
       X = structure(.Data=c(1,23,2,
```

 © 2011 John Wiley & Sons, Inc.

```
        57,3,59,4,14,5,36)
              .Dim=c(5,2))
```

Both codes above define the same data matrix but with different types of nodes. The second code is preferable when the number of variables involved is large. It is also advised to use R and the command dput to directly import the data in WinBUGS with this format; see Section 3.4.6.2 in Ref 6 for more details.

Alternatively, the data can be represented in a simpler, rectangular form. The names of the vector nodes (followed by []) must be stated in the first row followed by the variable values for each observation in each line. The data must conclude with the command END followed by a blank line. All variables—vector nodes must have the same length.

```
DATA
y[]     x1[]    x2[]    x3[]
20      23      1       14
19       6      0       23
32      18      1       12
22      20      1       12
...     ...     ...     ...
END
(Blank line)
```

RUNNING AN MCMC IN WINBUGS AND OBTAINING POSTERIOR SUMMARIES

Once the model, the data, and the initial values have been specified we can compile and run the MCMC algorithm. The procedure to be followed is described below.

1. Open the Specification tool under the Model menu and highlight the word model. WinBUGS checks the syntax of the model by clicking the check model button. A message appears in the bottom left-hand corner of the window indicating whether the model is syntactically correct or not. If an error message appears, the model code must be corrected and revised and then the model syntax must be checked again (note that the cursor indicates the location of the error in the code).
2. If the model is syntactically correct, the load data box becomes accessible. Highlight the word list, if the data are presented in the list form, or the first line of the rectangular form and press the load data button. The message data loaded should appear. If more than one dataset is available we repeat the above procedure until all the data are loaded.
3. If steps 1 and 2 are successful, then the model is compiled by clicking the corresponding button. The message model compiled should appear. If an error message is produced, then the model code must be corrected and the procedure must be started from step 1 again.
4. Once the model is compiled successfully, the load inits button becomes active. Highlight the word list that contains the initial values of the stochastic components and click the load inits button. The button gen inits can alternatively be used to generate random initial values for the parameters. If the message this chain contains uninitialized variables appears, then some stochastic nodes do not have initial values. In such cases gen inits can be also used. If the message model is initialized appears in the left bottom bar of WinBUGS then the MCMC is ready to start generating random values.
5. Open the Update tool under the Model menu and specify the number of iteration in the burn-in period in the box entitled updates. The number in the box refresh shows how often the results are refreshed, thin defines the lag between stored iterations and iteration shows the current number of iterations of the MCMC algorithm. Insert the desired numbers and click update. The iteration counter will start changing until the required number of iterations is reached.
6. Open the Sample Monitor tool under the Inference menu and select the parameters for which we wish to infer about. Write the name of the parameters in the node box and press set. This should be repeated for all the parameters that we wish to monitor. Return to the update tool and select the number of iterations that we wish to generate and then press the updates button. When the counter reaches the required number of iterations then an MCMC sample from the posterior distribution is available for the monitored variables.

Analysis of the MCMC output and inference concerning the posterior distribution can be obtained by the Sample Monitor tool. Summaries of the posterior distributions can be derived by typing the name of the parameter of interest in the node box (or using the pull down menu) and pressing the stats button. The posterior summaries of all monitored parameters can be extracted if we use the asterisk

 © 2011 John Wiley & Sons, Inc.

(*) in the `node` box. A density plot of the marginal posterior distribution of each node can be obtained using the `density` button.

The remaining buttons can be used to obtain a basic analysis of the MCMC output by checking the convergence of the chain (`history`), the Monte Carlo (MC) error (`stats`) and the autocorrelation (`auto cor`). The thinning interval of observations used to derive the summaries can be controlled by the corresponding box. A detailed diagnostic analysis of the MCMC output can be conducted in R, using the CODA[7] and BOA[8] packages. The observations of the posterior distributions of the WinBUGS output can be obtained in a compatible format using the option `coda`.

AN ILLUSTRATING EXAMPLE—IMPLEMENTING BAYESIAN LASSO IN WINBUGS

The Lasso[9] is a shrinkage and variable selection method for normal linear regression models. Its estimates are obtained by imposing the L_1 norm penalty on the linear regression coefficients. Owing to the shape of the L_1 norm, the regression coefficients are shrunk toward zero and some of them are set exactly equal to zero.

The Lasso estimates have an apparent Bayesian perspective as they correspond to the posterior mode of a Bayesian normal linear regression model using a double-exponential prior distribution for the regression coefficients. The corresponding model is described by the following formulation:

$$\begin{aligned} Y_i|\boldsymbol{\beta}, \tau &\sim \text{Normal}(\mu_i, \tau^{-1}), \text{ for } i = 1, \ldots, n \\ \boldsymbol{\mu} &= \mathbf{X}\beta \text{ with } \boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)^T \\ \beta_j &\sim \text{DE}\left(0, \frac{1}{\tau\lambda}\right), \quad \text{for} \quad j = 1, \ldots, p, \\ \tau &\sim \text{Gamma}(a, d), \end{aligned}$$

where $\tau = 1/\sigma^2$ is the precision of the Normal regression model and λ is the shrinkage parameter which controls the prior variance given by $2/(\tau\lambda)^2$. Different values of λ correspond to different levels of shrinkage. Small values of λ correspond to large variance introducing low information in the Bayesian model and essentially not imposing any shrinkage on the model parameters. On the other hand, large values of λ correspond to strong prior that $\boldsymbol{\beta}$ is close to zero resulting to high levels of shrinkage. To ensure that λ has the same meaning for all covariate effects, we assume, without loss of generality, that all the variables have mean zero and variance equal to one.

Example

We illustrate the Bayesian Lasso regression using a simulated data set that consists of $n = 50$ observations and $p = 5$ covariates generated from independent standardized normal distributions and the response from

$$Y_i \sim \text{Normal}(X_{i3} + X_{i4}, 2.5^2), \quad \text{for} \quad i = 1, \ldots, 50.$$

The Bayesian Lasso is performed on this dataset for different values of λ by considering 2,000 updates after discarding additional 1,000 iterations as burn-in period.

In the Bayesian implementation, we standardize both the response and the covariates. For this reason, the constant parameter is not included in the model for the standardized variables as it is natural to set it equal to zero. All the variables are standardized within the WinBUGS code (see lines 2–14 in the code of Algorithm 1). Note that WinBUGS allows to define stochastic nodes twice only in the case of transforming response variables as in this example. The original regression parameters are calculated in WinBUGS using simple logical nodes as

$$\begin{aligned} \beta_0 &= \overline{y} - S_y \sum_{j=1}^{p} b_j \overline{x}_j \\ \beta_j &= \frac{S_y}{S_{x_j}} b_j \\ \sigma^2 &= S_y^2 \sigma_z^2 \end{aligned}$$

where σ_z^2 is the error variance, b_j the coefficients of the regression model using the standardized variables, $\overline{y}$ and $\overline{x}_j$ the sample means, and S_y^2 and $S_{x_j}^2$ the sample variances for Y and X_j respectively.

The code for the above-defined Bayesian Lasso model is given in Algorithm 1. The model was initialized by setting all standardized coefficients equal to zero and the corresponding precision equal to one. Following the arguments in Ref 10, we choose the value of $\lambda = 0.067$.

We follow the procedure described in Section 'Running an MCMC in WinBUGS and Obtaining Posterior Summaries' to obtain summaries for the posterior distribution. The posterior summaries of the parameters of the Lasso regression coefficients and the variance of the regression model are displayed in Figure 1 for the standardized data and in Figure 2 for the original data. The posterior means and medians of the coefficients of X_1 and X_2 are very low indicating that these are the least important variables. As expected, the variables used to generate

 © 2011 John Wiley & Sons, Inc.

```
model{
    # calculating means and standard deviations
    meany <- mean( y[] )
    sdy     <- sd(y[])
    for (j in 1:p){
        meanx[j] <- mean(X[,j])
        sdx[j]     <- sd(X[,j])
        meansd[j] <- meanx[j]/sdx[j]
    }
    # standardizing data
    for(i in 1:n) {
        z.y[i] <- (y[i]-meany)/sdy
        for (j in 1:p){  Z.X[i,j] <-  (X[i,j]-meanx[j])/sdx[j] }
    }
    # model likelihood
    for(i in 1:n) {
        z.y[i] ~ dnorm(mu[i], tau)
        mu[i] <- inprod(Z.X[i,],b[])    # Z.X is nxp
    }
    #
    # prior for b (parameters of the model using standardized variables
        )
    tt <- lambda*tau
    for(j in 1:p){ b[j] ~ ddexp(0, tt ) }
    #
    # prior for tau (for the model with standardized variables)
    tau ~ dgamma(0.0001, 0.0001)
    #
    sz.sq <- 1/tau
    sz    <- sqrt(sz.sq)
    #
    # calculating the parameters for the original model
    # (without standardized variables)
    beta0<- meany - sdy * inprod( b[], meansd[] )
    for (j in 1:p){ beta[j] <- sdy*b[j]/sdx[j] }
    sigma.sq <- sdy*sdy*sz.sq
    sigma <- sqrt(sigma.sq)
}
```

ALGORITHM 1 | WinBUGS Code for the Bayesian Lasso Model.

the response (X_3, X_4) have the highest posterior measures, whereas the last covariate (X_5) has moderate measures. Similar conclusions are drawn from the Figure 3, which gives the densities of the posterior summaries of the Lasso coefficients. The results here provide some evidence about the important variables, although the variable selection problem is not directly addressed. Moreover, we observe that the posterior means of $\boldsymbol{\beta}$ are similar to the ordinary least square estimates $\left[\widehat{\boldsymbol{\beta}} = (0.16, -0.19, 0.067, 1.19, 1.67, -1.11)^T\right]$ concluding that our prior was essentially noninformative implementing minor (or no) shrinkage on the model parameters. For illustration, we also rerun the model with $\lambda = 2$ resulting to a posterior with summaries given in Figure 4. For this value of λ, the posterior means are shrunk by 42% for β_1 and from 5 to 20% for the rest of the coefficients. Differences of the coefficients obtained using the two values of λ are depicted in the box plots of Figure 5. The most noteworthy change is observed for β_5, for which zero lies outside the 95% posterior credible interval in the first run and inside this interval in the second run.

We can extend the aforementioned model in order to incorporate the variable selection procedure in our formulation. This can be achieved by introducing a vector of binary indicators $\boldsymbol{\gamma}$ that highlights which variables are included in the model (with $\gamma_i = 1$) or not (with $\gamma_i = 0$) as in Refs 11 and 12. This formulation was proposed by Lykou and Ntzoufras[10] and is described below.

$$Y_i|\boldsymbol{\beta}, \tau, \boldsymbol{\gamma} \sim \text{Normal}(\mu_i, \tau^{-1}), \text{ for } i = 1, 2, \ldots, n$$
$$\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}^{(\boldsymbol{\gamma})} \text{ with } \boldsymbol{\beta}^{(\boldsymbol{\gamma})} = (\beta_1^{(\boldsymbol{\gamma})}, \beta_2^{(\boldsymbol{\gamma})}, \ldots, \beta_p^{(\boldsymbol{\gamma})})^T$$

© 2011 John Wiley & Sons, Inc.

1_node_stats_standardized

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
b[1]	−0.0505	0.1021	0.002783	−0.2448	−0.05336	0.152	1001	2000
b[2]	0.02172	0.09605	0.00228	−0.1711	0.02279	0.2092	1001	2000
b[3]	0.3659	0.09715	0.002324	0.1757	0.3666	0.5611	1001	2000
b[4]	0.5782	0.09821	0.002123	0.3859	0.5757	0.7774	1001	2000
b[5]	−0.2293	0.1006	0.002069	−0.4231	−0.2312	−0.03014	1001	2000
sz	0.6609	0.06534	0.001564	0.5464	0.6557	0.8078	1001	2000

FIGURE 1 | Posterior summaries of the Lasso regression parameters using standardized data ($\lambda = 0.067$).

2_node_stats_full

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
beta0	0.1655	0.1622	0.0032	−0.1563	0.1688	0.4795	1001	2000
beta[1]	−0.1889	0.382	0.01041	−0.9156	−0.1996	0.5686	1001	2000
beta[2]	0.07233	0.3199	0.007593	−0.5699	0.07589	0.6966	1001	2000
beta[3]	1.19	0.3158	0.007557	0.5714	1.192	1.824	1001	2000
beta[4]	1.668	0.2834	0.006126	1.113	1.661	2.243	1001	2000
beta[5]	−1.103	0.4837	0.009954	−2.035	−1.112	−0.145	1001	2000
sigma	2.197	0.2173	0.0052	1.817	2.18	2.686	1001	2000

FIGURE 2 | Posterior summaries of the Lasso regression parameters for the unstandardized data ($\lambda = 0.067$).

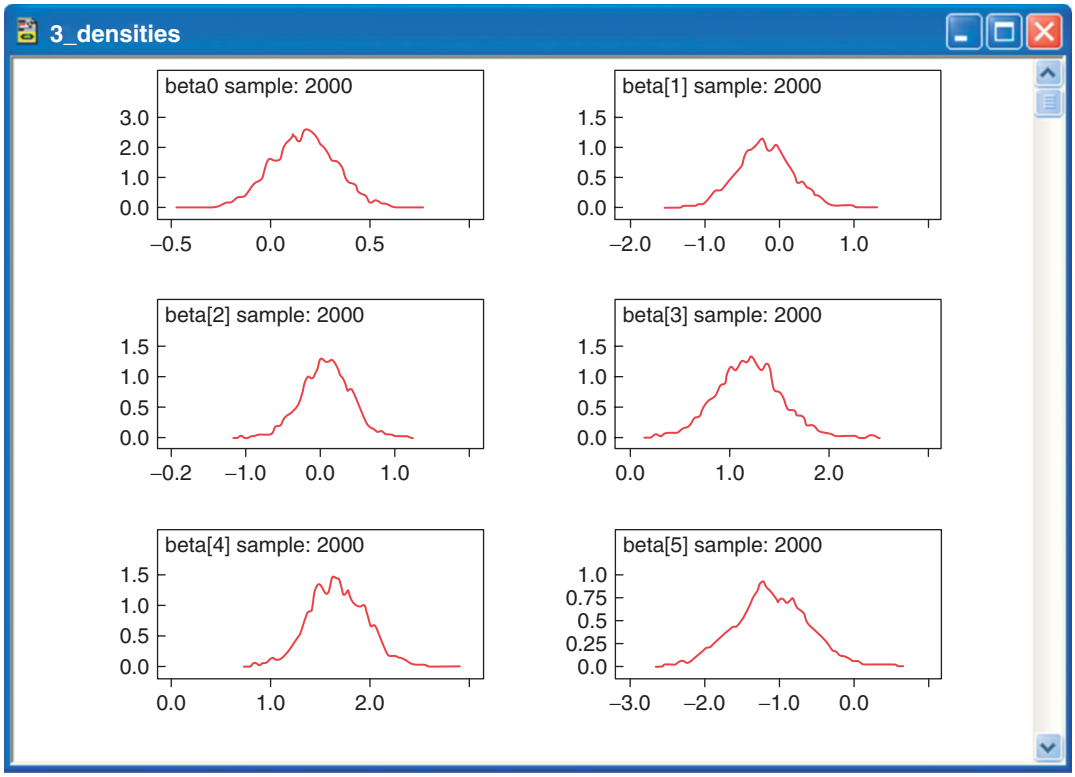


FIGURE 3 | Posterior densities of the Lasso regression coefficients for the unstandardized data ($\lambda = 0.067$).

© 2011 John Wiley & Sons, Inc.

$$\beta_j^{(\boldsymbol{\gamma})} = \gamma_j \beta_j$$

$$\beta_j|\tau \sim \mathrm{DE}\left(0, \frac{1}{\tau\lambda}\right), \quad \text{for} \quad j = 1, \ldots, p,$$

$$\tau \sim \mathrm{Gamma}(a, d),$$

$$\gamma_j \sim \mathrm{Bernoulli}(\pi_j)$$

where Bernoulli(π) is the Bernoulli distribution with success probability π. The actual model parameters $\boldsymbol{\beta}^{(\boldsymbol{\gamma})}$ are denoted with a vector of length p with elements $\beta_j^{(\boldsymbol{\gamma})} = \gamma_j \times \beta_j$ (for $j = 1, \ldots, p$) which are constrained to zero if X_j is not included in the model formulation.

In order to extend the model code of Algorithm 1 according to the above-mentioned formulation we need to

1. substitute line 18 with the following syntax

```
mu[i] <- inprod(Z.X[i,],
          bgamma[])
```

in order to substitute β_j by $\gamma_j b_j$,

4_stats_lambda2

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
b[1]	−0.0291	0.09658	0.002671	−0.2226	−0.0229	0.1582	1001	2000
b[2]	0.01712	0.09101	0.002357	−0.1641	0.01255	0.2111	1001	2000
b[3]	0.3292	0.1033	0.002385	0.1267	0.3278	0.5392	1001	2000
b[4]	0.5424	0.1067	0.002194	0.3387	0.541	0.7633	1001	2000
b[5]	−0.1876	0.1063	0.002806	−0.4028	−0.1829	0.006072	1001	2000
beta[1]	−0.1089	0.3613	0.009992	−0.8328	−0.08568	0.5918	1001	2000
beta[2]	0.05702	0.3031	0.007851	−0.5466	0.04179	0.703	1001	2000
beta[3]	1.07	0.3359	0.007752	0.4119	1.066	1.753	1001	2000
beta[4]	1.565	0.3078	0.006331	0.9774	1.561	2.202	1001	2000
beta[5]	−0.9025	0.5115	0.0135	−1.938	−0.88	0.02921	1001	2000
beta0	0.2077	0.167	0.003606	−0.131	0.2079	0.5444	1001	2000
sigma	2.408	0.2394	0.006507	1.999	2.385	2.93	1001	2000
sz	0.7242	0.07201	0.001957	0.6013	0.7174	0.8811	1001	2000

Standardized data: b [j] = b_j ; sz = σ_z
Unstandardized data: beta = β_0 (constant term); beta [j] = β_j ; sigma = σ

FIGURE 4 | Posterior summaries of the Lasso regression parameters for standardized and unstandardized data ($\lambda = 2$).

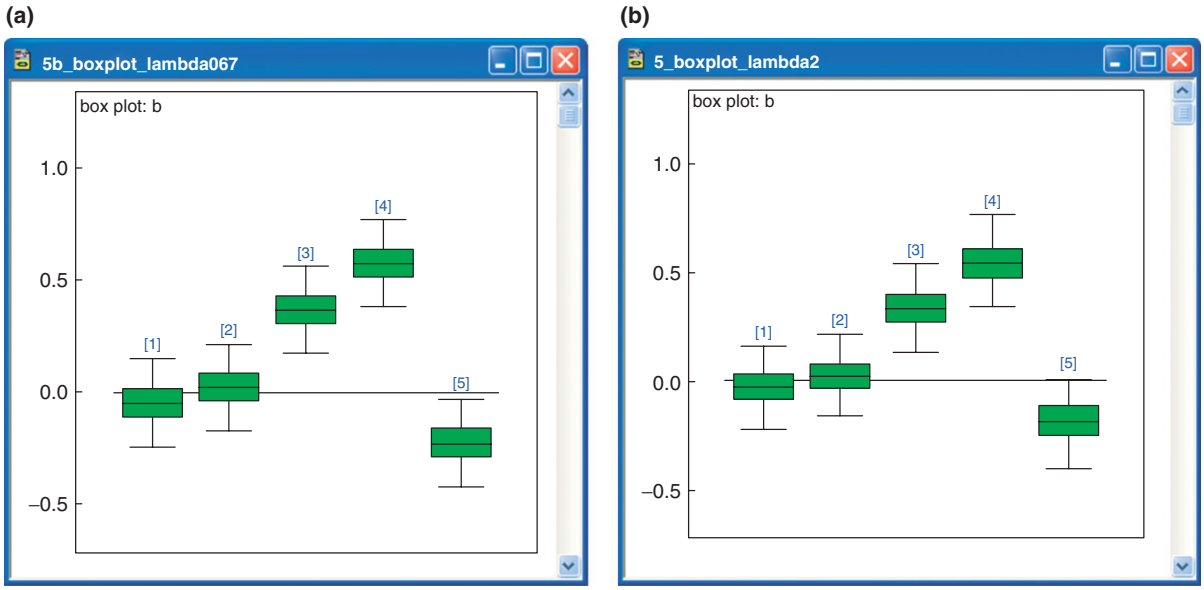


FIGURE 5 | Posterior box plots describing the 95% credible intervals of the regression coefficients using the standardized data (obtained by inserting node `b` in the node box of `Compare` tool inside the `Inference menu`.). (a) $\lambda = 0.067$; (b) $\lambda = 2$.

© 2011 John Wiley & Sons, Inc.

2. substitute line 33 by

```
beta0<- meany - sdy *
 inprod( bgamma[], meansd[] )
```

in order to calculate correctly the constant parameter for any given model,

3. add the following syntax

```
for (j in 1:p){
    # b x gamma for the
        standardized data
    bgamma[j] <- b[j] *
        gamma[j]
    # prior for gamma
    gamma[j]~dbern(0.5)
    # beta*gamma for the
        unstandardized data
    betagamma[j] <- beta[j] *
        gamma[j]
    }
```

in order to define $b_j^{\boldsymbol{\gamma}} = \gamma_j b_j$, $\beta_j^{\boldsymbol{\gamma}} = \gamma_j \beta_j$ and the prior for each γ_j.

Posterior summaries for $\boldsymbol{\beta}^{(\boldsymbol{\gamma})}$ are given in Figure 6 for $\lambda = 0.067$. We clearly observe that the first two coefficients are shrunk toward zero (the posterior means shrunk by 99.6 and 96% respectively). Also the posterior distribution of β_5 was dramatically pushed toward zero with the posterior mean shrunk by 83.7%. On the other hand, the posterior means for the coefficients of X_3 and X_4 remained almost unchanged (shrunk by 8.6 and 0.06% respectively). The posterior inclusion probabilities for each covariate can be estimated from the posterior means of each γ_j as given in Figure 7. From the results, we clearly observe that covariates X_3 and X_4 must be included in the model having posterior probabilities 0.9 and 1.0, respectively. Covariates X_1 and X_2 have very low posterior probabilities ($\sim 1.8\%$), while covariate X_5 is included in the model with posterior probability of 17%.

Figure 8 displays the densities of the posterior distributions of $\boldsymbol{\beta}^{(\boldsymbol{\gamma})}$. The posterior distributions of the coefficients for the non-important variables have high spikes at zero except for covariate X_4, which is the most important one in this illustration and its posterior distribution is well placed away from zero. The posterior distribution of β_3 is bimodal, with one mode at zero and one at a positive value. However, the posterior median of the distribution and the posterior mean of the corresponding inclusion probability (given by the mean of γ_3 and it is equal to 0.90) indicate that this variable should also be included in the model.

DISCUSSION: A FUTURE FULL OF PROMISES

Over the last years WinBUGS gained a tremendous popularity within the scientific community. WinBUGS has been used in a wide range of practical problems having different scientific backgrounds. It has been the object of interest in various short courses and workshops. More and more universities include Bayesian data analysis courses using WinBUGS in their syllabus; see 'BUGS resources online' link in the

6_stats_gamma

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
beta0	0.2324	0.1997	0.009046	–0.08211	0.2041	0.7027	1001	10000
betagamma[1]	7.3E-4	0.05531	5.839E-4	0.0	0.0	0.0	1001	10000
betagamma[2]	0.002618	0.04639	5.232E-4	0.0	0.0	0.0	1001	10000
betagamma[3]	1.087	0.4784	0.02172	0.0	1.155	1.844	1001	10000
betagamma[4]	1.669	0.2978	0.003231	1.076	1.669	2.251	1001	10000
betagamma[5]	–0.1804	0.4392	0.01275	–1.541	0.0	0.0	1001	10000
sigma	2.39	0.264	0.00703	1.941	2.367	2.98	1001	10000
bgamma[1]	1.951E-4	0.01479	1.561E-4	0.0	0.0	0.0	1001	10000
bgamma[2]	7.862E-4	0.01393	1.571E-4	0.0	0.0	0.0	1001	10000
bgamma[3]	0.3343	0.1472	0.006681	0.0	0.3552	0.5672	1001	10000
bgamma[4]	0.5783	0.1032	0.00112	0.373	0.5783	0.78	1001	10000
bgamma[5]	–0.0375	0.09131	0.002651	–0.3204	0.0	0.0	1001	10000
sz	0.7188	0.07941	0.002114	0.5837	0.7118	0.8963	1001	10000

Standardized data: bgamma [j] = $\gamma_j \times b_j$; sz = σ_z
Unstandardized data: beta = β_0 (constant term); betagamma [j] = $\gamma_j \times \beta_j$; sigma = σ

FIGURE 6 | Posterior summaries of the Lasso regression coefficients with variable selection.

 © 2011 John Wiley & Sons, Inc.

6_stats_gamma2

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
gamma[1]	0.0181	0.1333	0.001983	0.0	0.0	0.0	1001	10000
gamma[2]	0.0178	0.1322	0.001988	0.0	0.0	0.0	1001	10000
gamma[3]	0.9016	0.2979	0.01782	0.0	1.0	1.0	1001	10000
gamma[4]	1.0	0.0	1.0E-12	1.0	1.0	1.0	1001	10000
gamma[5]	0.173	0.3782	0.01213	0.0	0.0	1.0	1001	10000

FIGURE 7 | Posterior summaries the indicator parameters included in the Bayesian Lasso model.

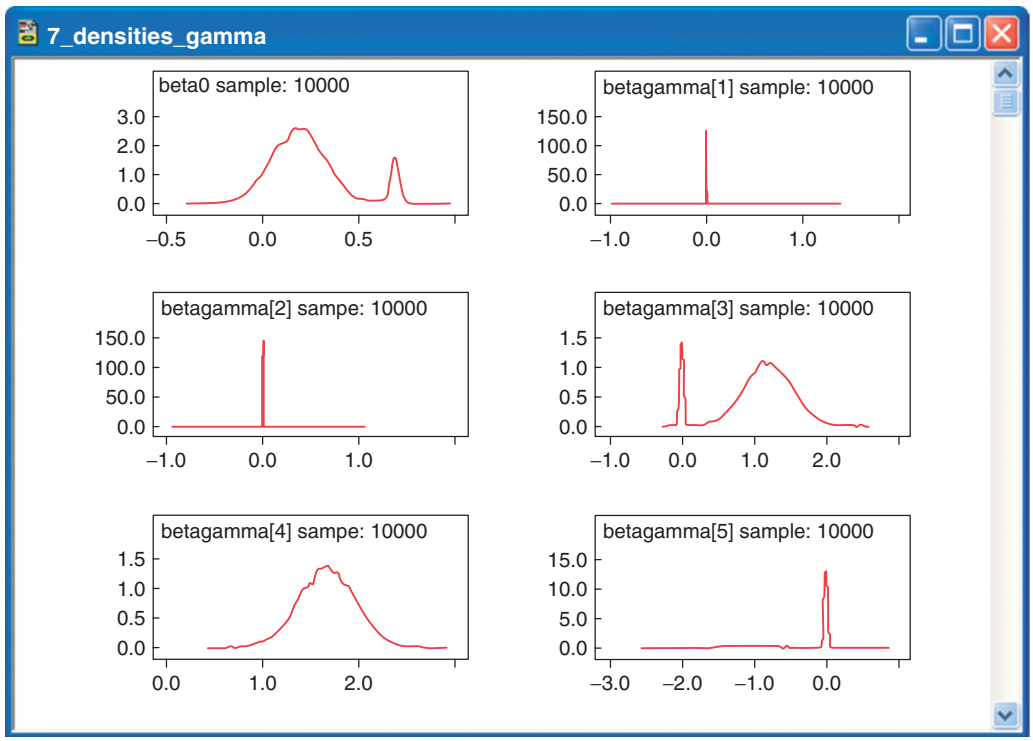


FIGURE 8 | Posterior densities for model parameters $\beta^{(\gamma)}$.

WinBUGS website for a coherent list of such activities and courses.

The popularity of WinBUGS has motivated various expansions and made WinBUGS applicable in a wider range of disciplines, such as social, actuarial science, population genetics, and archaeology. GeoBUGS developed by the team in the Imperial College at St Mary's Hospital can be used to fit spatial models and produce a range of maps as output. PKBUGS has been developed by Dave Lunn to fit pharmacokinetic models. The WinBUGS development interface (WBDev)[13] allows the users to define their own distributions and functions. This was originally designed for social scientists but it has been used by other researchers as well. Lunn has also developed the WinBUGS Differential Interface (WBDiff) which allows the use of complex functions via arbitrary systems of ordinary differential equations and the WinBUGS jump interface to perform variable selection through the Reversible Jump MCMC.[14,15]

WinBUGS's popularity has also been extended due to its compatibility with other statistical packages and software. WinBUGS runs from GenStat, which is a software for bioscience. It can also be called via R using the R2WinBUGS R-library on Comprehensive R Archive Network (CRAN) site. There are also codes that can be used to call WinBUGS through Stata,[16] SAS and a code for Excel that does not require any knowledge of WinBUGS. MATBUGS is a Matlab interface for WinBUGS, which has been extended to run in Linux systems. Details about it can be found on www.mrc-bsu.cam.ac.uk/bugs/winbugs/remote14.shtml.

© 2011 John Wiley & Sons, Inc.

Note that WinBUGS was now stabilized in version 1.4.3 and it will not be developed any more. Instead, the interest of the group is now turned on the development of OpenBUGS (www.openbugs.info) which is an open source version of WinBUGS with additional features and contributions. This project started in 2004 when Andrew Thomas moved from London to Helsinki and now is stable and reliable package running under Windows, Linux, and MAC operating systems. The possibility that researchers will be able to contribute and improve an already popular and successful software leaves high expectations for the future.

CONCLUSION

This article summarizes the basic concepts required to perform Bayesian analysis using the WinBUGS. It provides information on model specification and coding, algorithm implementation and output interpretation along with some toy examples, and a more detailed illustration that demonstrates the implementation of Bayesian Lasso in WinBUGS. It can be used as a brief introduction to WinBUGS for researchers with or without statistical background.

WinBUGS is a useful computational tool that fits complicated Bayesian models using MCMC methods. It is widely popular due to its numerous extensions and applications in various scientific fields. It is relatively straightforward to use with syntax similar to the one in R and Splus. Alternatively, DOODLE interface can be used to specify the structure of the model through a graphical representation. The recent development of OpenBUGS (the open source version of the program) creates high expectations for the future where any researcher will be able to contribute to the development and the improvement of this popular software.

REFERENCES

1. Lunn D, Spiegelhalter D, Thomas A, Best N. The bugs project: Evolution, critique and future directions. *Stat Med* 2009, 28:3049–3082.
2. Spiegelhalter D, Thomas A, Best N, Lunn D. WinBUGS User Manual, Version 1.4. MRC Biostatistics Unit, Institute of Public Health and Department of Epidemiology and Public Health, Imperial College School of Medicine, UK, 2003. Available at: http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml. (Accessed May 4, 2011).
3. Spiegelhalter D, Thomas A, Best N, Lunn D. WinBUGS Examples, vol 1. MRC Biostatistics Unit, Institute of Public Health and Department of Epidemiology and Public Health, Imperial College School of Medicine, UK, 2003. Available at: http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml. (Accessed May 4, 2011).
4. Spiegelhalter D, Thomas A, Best N, Lunn D. WinBUGS Examples, vol 2. MRC Biostatistics Unit, Institute of Public Health and Department of Epidemiology and Public Health, Imperial College School of Medicine, UK, 2003. Available at: http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml. (Accessed May 4, 2011).
5. Spiegelhalter D, Thomas A, Best N, Lunn D. WinBUGS Examples, vol 3. MRC Biostatistics Unit, Institute of Public Health and Department of Epidemiology and Public Health, Imperial College School of Medicine, UK, 2003. Available at: http://www.mrc-bsu.cam.ac.uk/bugs.
6. Ntzoufras I. *Bayesian Modeling Using WinBugs*. New York: John Wiley & Sons; 2009.
7. Best N, Cowles MK, Vines K. *CODA: Convergence Diagnostics and Output Analysis Software for Gibbs Sampling Output,* Version 0.30. Cambridge: MRC Biostatistics Unit, Institute of Public Health; 1996.
8. Smith BJ. *Bayesian Output Analysis Program (BOA), Version 1.1.5 User's Manual*, Technical Report. Department of Public Health, The University of Iowa, 2005. Available at: http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml. (Accessed May 4, 2011).
9. Tibshirani R. Regression shrinkage and selection via the lasso. *J R Stat Soc [SerB]* 1996, 58:267–288.
10. Lykou A, Ntzoufras I. "On Bayesian Lasso Variable Selection and the specification of the shrinkage parameter", Technical Report, Athens University of Economics and Business, 2011.
11. Kuo L, Mallick B. Variable selection for regression models. *Sankhyā, (B)* 1998, 60:65–81.
12. Dellaportas P, Forster J, Ntzoufras I. On Bayesian model and variable selection using MCMC. *Stat Comput* 2002, 12:27–36.
13. Lunn D. WinBUGS Development Interface (WBDev). *ISBA Bull* 2003, 3:10–11.
14. Lunn DJ, Whittaker JC, Best N. A Bayesian toolkit for genetic association studies. *Genet Epidemiol* 2006, 30:231–247.
15. Lunn DJ, Best N, Whittaker J. Generic reversible jump MCMC using graphical models. *Stat Comput* 2009, 19:395–408.
16. Thompson JT, Palmer T, Moreno S. Bayesian analysis in Stata using WinBUGS. *Stata J* 2006, 6:530–549.

 © 2011 John Wiley & Sons, Inc.