Journal of Statistical Software

December 2007, Volume 23, Issue 9. *http://www.jstatsoft.org/*

WinBUGSio: A SAS Macro for the Remote Execution of WinBUGS

Michael K. Smith
Pfizer Global
Research and Development

Helen Richardson
Pfizer Global
Research and Development

Abstract

This is a macro which facilitates remote execution of **WinBUGS** from within SAS. The macro pre-processes data for **WinBUGS**, writes the **WinBUGS** batch-script, executes this script and reads in output statistics from the **WinBUGS** log-file back into SAS native format. The user specifies the input and output file names and directory path as well as the statistics to be monitored in **WinBUGS**. The code works best for a model that has already been set up and checked for convergence diagnostics within **WinBUGS**. An obvious extension of the use of this macro is for running simulations where the input and output files all have the same name but all that differs between simulation iterations is the input dataset. The functionality and syntax of the macro call are described in this paper and illustrated using a simple linear regression model.

Keywords: Bayesian, macro, remote execution, SAS, **WinBUGS**.

1. Introduction

WinBUGS version 1.4 (Spiegelhalter, Thomas, Best, and Lunn 2003) now incorporates a batch script facility which allows the user to run **WinBUGS** models without having to point and click their way through the graphical user interface. This new functionality opens the way for remote running of **WinBUGS** models from other software packages. The aim of the macro described below is to allow the user to run a **WinBUGS** model from within SAS (SAS Institute Inc. 1999). It is assumed that the user has already tested the model within **WinBUGS**, checking syntax, convergence criteria and other standard tasks. The macro itself does not write the model or check its reasonableness but instead handles input and output from SAS to **WinBUGS** and back including actually calling **WinBUGS** itself. The authors assume that the reader knows sufficient SAS language to be able to create the requisite datasets and call macros.

WinBUGS is a powerful Bayesian analysis tool using Markov-Chain Monte-Carlo (MCMC) methods (Gilks, Richardson, and Spiegelhalter 1995) to sample model parameters (nodes) from the posterior distribution. The **WinBUGS** model specifies the model likelihood and prior distributions for the analysis. Specification of the full conditional distributions required for MCMC simulations is handled "behind the scenes". The workflow for running the model within **WinBUGS** is as follows:

1. Check the model syntax.
2. Load in the data from an ASCII file, in column format and / or constants specified in S language `list(...)` format.
3. Compile the code.
4. Load the initial values for the MCMC process.
5. Perform a "burn-in" of the MCMC to reach the stationary distribution for sampling values from the joint posterior distribution.
6. Set the model parameters that you wish to monitor and generate statistics from the posterior distribution.
7. Perform iterations of the MCMC to generate samples of these model parameters from the posterior distribution.
8. Calculate summary statistics of these MCMC samples for the model parameters.
9. Save the results to a log file.

2. Main features

The WinBUGSio macro provides functionality for the input and output of information to and from **WinBUGS** as well as executing the batch code. The macro gives the user a number of different options for specifying the dataset to be used in analysis. The dataset can be an internal, working SAS dataset which will be written out to an ASCII text file in column format and / or S language `list(...)` format, or the user can specify an external ASCII file or external SAS dataset containing data in the appropriate format for **WinBUGS**. If the users opt to use existing SAS datasets for analysis, then they are required to specify which variables are required in the **WinBUGS** data file and also specify labels to be used for these. Since **WinBUGS** is case-sensitive the user needs to be careful to use labels which correspond to data items specified in the model, for example covariate labels, response variable names etc. However this step leaves flexibility with the user to re-label data items between the SAS dataset and **WinBUGS**, or to use a new SAS dataset with an existing **WinBUGS** model without having to rename variables in the **WinBUGS** model. The second step is to write the batch code for running the model remotely using **WinBUGS**. The standard **WinBUGS** batch commands for checking model syntax, reading the appropriate datasets, compiling the model and reading in initial value files are pre-specified in the macro. The user specifies the appropriate directory path and filenames for the various files and a list of the model parameters or nodes to be sampled from the joint posterior distribution. This list creates the necessary commands within the batch script to sample these parameters. **WinBUGS** requires nodes to be monitored for them to be output as summary statistics from the posterior distribution. Finally the batch code commands for producing summary statistics from the posterior distribution are given, and the summary statistics are written out to a log file. The batch script is written out to the default **WinBUGS** installation directory and the user specifies the filename for the script. The macro then executes this batch script in **WinBUGS** by executing a DOS command for running **WinBUGS** in batch mode. **WinBUGS** will open and the batch commands executed. However, the macro code is written so that **WinBUGS** runs in a minimised window and as soon as the batch script completes successfully the window is closed and control is returned to SAS. This allows "quiet" running of the macro and impact on the desktop work is minimised.

Finally the log file containing the summary statistics from the MCMC sample of the posterior distribution for each monitored node is read into SAS and converted into a SAS dataset. The macro also provides functionality for the user to select variables which will have all MCMC iterations stored (**CODA** output, see Best, Cowles, and Vines 1997) and read in to SAS for post-processing.

This completes the functionality of the macro, however it is possible to wrap the macro in a simulation framework creating a new SAS input dataset at each iteration of the simulation, running the macro as described above and then storing the output SAS dataset in a larger appended dataset. This provides functionality for conducting simulations of operating characteristics under a fully Bayesian framework using convenient, commonly available tools such as SAS and **WinBUGS**.

3. Macro call syntax

The macro call allows the user to set up quickly the necessary parameters which will be used to control the output of data to **WinBUGS**. Deeper functionality is available to more experienced users who wish to use external data sources or control the output of data to **WinBUGS** more closely.

```
%WinBUGSIO(
dsin=WBinput,
dir=c:/program files/WinBUGS14/,
datafile1=data1,
datafile2=data2,
constlist=%str((N=5, xbar=3)),
dsvar=(covar, Response),
dsfmt=(best12.,best12.),
varlab=('x[]','Y[]'),
initsfile=(initfile1,...,initfilen),
modelfile=model,
burnin=100,
sample=500,
nodes=(alpha, beta, mu, sigma),
codavar=(alpha,beta,sigma),
logfile=log,
batchfile=batch,
dsout=WBoutput);
```

Where default values exist for the macro variables in the macro call, these are included in brackets below. If no default value exists this will be stated, and the user must supply a value.

- `dsin` (default: `WBinput`)
  Specifies the input SAS dataset to be used as the basis for the **WinBUGS** analysis. The dataset should contain the variables and information to be used to create the ASCII file dataset required for running **WinBUGS**. The dataset can be either a local SAS working dataset (in the temporary WORK library) or an external referenced library dataset with an associated LIBNAME extension. External SAS permanent datasets can also be specified by their file path. If the dataset file is in the directory specified below then no extra file path information is required, otherwise full file path information should be given.

- `dir` (default: `C:/PROGRAM FILES/WinBUGS14/`)
  Specifies the directory path for the ASCII dataset files, MCMC initial value file, model file and log file. This will be the directory that stores all input and output files. Batch files are always written to the **WinBUGS** default installation directory for convenience. Since these are themselves created by the macro using the specified inputs and outputs and can be recreated at any time when the macro is re-run it was decided that there was little need to write this batch file to the same directory. All the filenames specified below, except for the batch file, will be written to this path. The path must already exist on the computer otherwise the macro will halt with an error.

- `datafile1` (default: `data1`)
  Specifies the filename for the column-format dataset. This file is an ASCII TXT file of any length containing the **WinBUGS** variable names for each column of data and the data itself. This allows for great flexibility in specifying quite large datasets as well as larger rectangular style arrays if appropriate subscripting is used in the variable labels (explained below). This macro variable in conjunction with the `dsvar` macro variable allows the user to specify external ASCII text files containing preprepared **WinBUGS** datasets. If `dsvar=external` and `datafile1` specifies a full path and filename then this external file will be used in the batch script.

- `datafile2` (default: `data2`)
  Specifies the filename for the ASCII file containing the constants used in the **WinBUGS** model file. These are usually items such as the number of observations, number of experimental units, fixed constants for the model which are readily available to **WinBUGS** etc. Similarly to `datafile1`, the user can specify external ASCII text files for pre-prepared **WinBUGS** datasets. If `constlist=external` and `datafile2` specifies a full path and filename then this file will be used in the batch script.

- `constlist` (no default)
  Specifies the constants described above in S language `list(...)` format. For example `constlist=%str(N=5, xbar=3)` creates an ASCII file with contents `list(N=5,`

xbar=3). The macro adds the 'list' statement and parentheses. The context and meaning of this example can be seen in the worked example below. Items in this list will be constructed into a list format and written to the file specified in the datafile2 macro variable. If constlist has an assigned value and dsvar has no assigned value then the macro assumes that list format data only is being used for **WinBUGS** analysis. If constlist = EXTERNAL then the external constants list file given in the datafile2 statement will be used in the batch script.

- dsvar (no default)
  This macro variable contains a list of the input SAS dataset variables to be written to the column format data file specified in the datafile1 macro variable. The list should contain SAS variable names. Only the specified variables from the SAS dataset that appear in this list will be written to the datafile. Similarly to constlist, if dsvar is assigned a value, but constlist is not then the macro assumes that list format data only is being used in analysis. If dsvar = EXTERNAL then the external column format dataset specified in the datafile1 statement will be used in the batch script.

- varlab (no default)
  This macro variable gives labels to be used as column headers for the column format dataset specified in the datafile1 macro variable. These titles will refer to **WinBUGS** data, covariates and other variables as specified in the **WinBUGS** model, and will use the **WinBUGS** model names and labels. As **WinBUGS** is case sensitive, case sensitivity is important in specifying these labels. Each variable label is enclosed in quotes to allow easy parsing of the list. The labels are interpreted by **WinBUGS** in the same way as S language vectors or arrays - subscripted by square brackets. Thus the label 'Y[]' is a vector variable called Y in the **WinBUGS** code, likely to be subscripted within a loop within the **WinBUGS** model code:

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

  This construct also allows the user to specify matrix structures within the column format dataset thus:

```
c('x[]','Y[,1]','Y[,2]','Y[,3]','Y[,4]')
```

  This choice of varlab would specify a covariate x and a two-dimensional array for Y which has four columns and the number of rows equal to the length of the number of observations in the datafile.

- initsfile (default: inits)
  Specifies a list of the filenames for the locations of the MCMC initial values for each model parameter. If more than one chain is required then a list of initial value files can be specified e.g. (inits1, inits2, inits3).

- modefile (default: model)
  Specifies the location of the **WinBUGS** model file for the analysis. We assume that the

user has already written this model file and has performed basic checks that the model provides sensible inferences and converges to the stationary distribution of the MCMC process for sampling from the joint posterior distribution. The warning in the **WinBUGS** help file and manual that MCMC sampling can be dangerous is not to be taken lightly, and the users are encouraged to work from known results and expand complexity slowly in their modelling. The authors of this macro cannot be held responsible for failure by the user to check properly their own models.

- `burnin` (default: `100`)
  Specifies the number of MCMC iterations to treat as the burn in on the Markov Chain. Inference on the specified nodes will not include these iterations as we assume that during this time the Markov Chain has not yet converged to a stationary distribution.

- `samples` (default: `500`)
  Specifies the number of MCMC iterations to be used in inference about the nodes of interest. Depending on the priors used (whether they are conjugate or not) and properties of the Markov Chain and how well it is 'mixing' i.e. how well it is sampling from the whole joint posterior distribution we may need more or fewer samples in order to make sensible inferences. We assume that the users have already determined how many samples will be needed to make valid inferences by running their model interactively within **WinBUGS** and performing standard diagnostic criteria.

- `nodes` (no default)
  Specifies a list of model parameters for which the user wants to report summary statistics of the MCMC sample from the joint posterior distribution. These summary statistics are provided by **WinBUGS** and include the mean, standard deviation, median, lower 5% and upper 95% quantiles, the number of samples and a measure of the Monte Carlo standard error of the mean which is given by the sample standard deviation divided by the square root of the number of simulation draws (number of samples taken). Due to limitations of the **WinBUGS** batch language other quantiles are not available, although these can be found in the GUI interface tool within **WinBUGS**. Other quantiles can be specified by hard-coding an option deep within the **WinBUGS** framework but since this persists across **WinBUGS** runs, it is not offered as an option here and further discussion will be limited. If this option interests the reader, please contact the first author. It is also worth pointing out that only stochastic nodes and functions of stochastic nodes are available for monitoring. It is not possible to monitor data or constants. The SAS macro code does not check whether the values input in this macro variable are valid stochastic nodes. Instead **WinBUGS** ignores any that are not valid stochastic nodes in the batch command statements and prints an error in the log file. Unfortunately **WinBUGS** will not display summary statistics for any model parameters that are not specified in this list, however it is not necessary, and often not desirable, to print out summary statistics for all stochastic nodes in a model and so the user should identify the elements of the model that are important for inference. The **WinBUGS** wildcard `*` can be used to monitor all nodes in the **WinBUGS** model.

- `CODAvar` (no default)
  Specifies a list of monitored nodes for which the values of the MCMC iterations will be written to file. **CODA** is a set of S language diagnostic functions which can be used

to assess model fit and show summaries of the posterior distribution. The iteration values are stored in a text file, read into SAS and stored in the session WORK library. If no nodes are listed in the `CODAvar` variable then no output is produced. The user may specify any of the node variables listed in the `NODES` macro variable, or choose the wildcard `*` which outputs all monitored nodes to the **CODA** output file. If specific nodes are given in `CODAvar` then SAS datasets are created in the WORK directory with **CODA** output for each variable. If the wildcard `*` is used then a SAS dataset called 'CODA' is created in the WORK directory.

- `logfile` (default: `log`)
  Specifies the location of the **WinBUGS** log from the batch run. The macro automatically includes a batch statement to echo commands and write output statistics to an ASCII text log file. In order to facilitate easy reading of this file for SAS we have limited output to summary statistics although it is possible to output diagnostic graphs, kernel density plots, MCMC sample history etc. to the log.

- `batchfile` (default: `batch`)
  Specifies the filename of the batch file for running **WinBUGS**. Since this file is written with each invocation of the macro and the content of the batch file is regulated by the macro it is not essential to rename the batch file for each run. The batch file is automatically written to the default **WinBUGS** installation directory. This is purely to facilitate calling the file from the DOS command statement used for executing **WinBUGS** from the SAS macro.

- `dsout` (default: `WBoutput`)
  Specifies the SAS dataset that the node summary statistics will be written to. The SAS macro reads the log file, identifies the location of the start and end of the summary statistics block and reads all data between these markers. Thus the number of nodes monitored becomes irrelevant and SAS simply creates a dataset of the appropriate size. All the variables, except for the node name, in the resulting SAS dataset are numeric variables. This facilitates further processing of this dataset for example in identifying model parameters that have a 95% probability of being larger than a null value. In the example below we can easily identify whether the slope parameter `beta` has a 95% probability of being larger than zero. In cases where the monitored node is a vector then the vector subscript may appear in the variable name. In this case simple SAS code could be used to extract these subscripts. This could be used for example to find the dose where the estimated mean effect gives a clinically relevant difference or the time point at which a certain growth level is attained.

4. Example

As an illustrative example we will show the settings that might be used to analyse the **WinBUGS** 'Line' example. In this trivial example we wish to fit a simple linear regression to 5 observations. The dataset is as follows, `x=(1,2,3,4,5)` and `Y=(1,3,3,3,5)`. We will center the x covariate on its mean value, 3. We have `N=5` observations. The model is `Yi=alpha + beta*(xi-xbar)`. The **WinBUGS** code for this model is given by:

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

This model specifies that the observations `Yi` are normally distributed with residual error `1/tau`. **WinBUGS** uses precision = 1/variance in its specification of Normal distributions. The expected regression fit for each observation is given by the usual linear regression function. Within the **WinBUGS** model we also specify the prior distributions

```
tau ~ dgamma(0.001,0.001)
sigma <- 1 / sqrt(tau)
alpha ~ dnorm(0.0,1.0E-6)
beta ~ dnorm(0.0,1.0E-6)
```

Note that the precision `tau` is given a conjugate Gamma prior distribution, and that we transform this to give us an estimate of sigma. `Alpha` and `beta` are given the conjugate Normal prior distributions, centered on zero and with low precision (high variance). We assume from this point on that this model is available in an ASCII TXT file format, that the syntax has been correctly specified and that the user has checked convergence and inferences for reasonableness before running the SAS **WinBUGSio** macro.

The dataset for this analysis can be specified in two ways. Either as column format with columns for `x` and `Y` (note the case!) and a second datafile with the constants `N` and `xbar`, or as one list format datafile with vectors for x and Y, and constants `N` and `xbar`. We will illustrate how to set up the column format version here as this is probably how data will most often be passed to **WinBUGS**. Let us assume that a SAS dataset called `Line` has been created with columns `covar` and response containing the data for `x` and `Y` respectively. The model file is called `Linemodel.TXT`, **WinBUGS** MCMC initial value file is called `Lineinits.TXT` and all outputs are to be written to the directory C:\Temp. The macro automatically assumes that the extensions for the input and output files are `*.TXT` and so we do not need to specify this extension. We wish to monitor the nodes `alpha` (intercept), `beta` (slope), `mu` (expected model fits), and `sigma` (residual standard error). The macro call for this problem would be as follows:

```
%WinBUGSIO(
dsin=line,
dir= c:\temp\,
datafile1=data1,
datafile2=data2,
constlist=\%str(N=5, xbar=3),
dsvar=\%str(x, y),
dsfmt=\%str(),
varlab=\%str('x[]', 'Y[]'),
initsfile=lineinits,
modelfile=linemodel,
burnin=100,
samples=500,
```

Node	Mean	Sd	MCerror	2.5%	Median	95.5%	Start	sample
`alpha`	2.971	0.5993	0.02599	1.61	2.984	4.02	101	500
`beta`	0.7682	0.431	0.01646	-0.06315	0.7883	1.512	101	500
`mu[1]`	1.435	1.061	0.0395	-0.3374	1.408	3.505	101	500
`mu[2]`	2.203	0.7464	0.0291	0.8154	2.207	3.688	101	500
`mu[3]`	2.971	0.5993	0.02599	1.61	2.984	4.02	101	500
`mu[4]`	3.74	0.73	0.03234	2.268	3.773	5.117	101	500
`mu[5]`	4.508	1.038	0.04425	2.341	4.563	6.194	101	500
`sigma`	1.012	0.7447	0.06597	0.4013	0.7932	3.203	101	500

Table 1: Expected **WinBUGS** MCMC summary output for the `line` example.

```
nodes=\%str(alpha, beta, sigma),
codavar=\%str(),
logfile=log,
batchfile=batch,
dsout=WBoutput);
```

Note the case of the variables `x` and `Y` for the **WinBUGS** model code. Within SAS the case is not important, but it is important for the user to check that the case of these variable labels for the **WinBUGS** dataset are matched by what is expected in the model code. Table 1 below shows the expected SAS dataset of **WinBUGS** MCMC summary output using the macro parameters described above.

The macro reads in the variable names from the log file as well as the numeric values. Note that the number of samples used for inference is 500, starting at iteration 101. This shows that a burn-in of 100 samples was used and that the samples of the nodes of interest were taken from sample 101.

5. Discussion

The SAS macro presented here has been written to facilitate easy interface between SAS and **WinBUGS** and to act as a "one stop shop" for data output and input to and from **WinBUGS**. Interested readers are encouraged to visit the "Remote Running" page of the **WinBUGS** website: `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/remote14.shtml` which has additional helper functions and code for running **WinBUGS** from SAS as well as other packages such as R, **Excel**, Stata etc. The excellent **Excel** plugin of Woodward (2005) takes this to the next level, adding a very useful front end for the users to specify their model which is then parsed and interpreted by **WinBUGS**. Additional functionality for this macro is planned to include the ability to write out the datasets and batch files ready for running in **WinBUGS** then stop prior to actually running the batch script. This should enable users to run the model interactively within WinBUGS and allow error trapping, diagnostics of MCMC convergence etc.

Acknowledgments

The authors would like to thank Nebele Bekele of MD Anderson Cancer Center, Houston, TX and James H. Roger of Glaxo SmithKline, UK for their input to the development of this macro; to beta testers who provided useful feedback on the macro and its functionality and to the reviewers of this manuscript for their comments.

References

Best NG, Cowles MK, Vines K (1997). ***CODA**: Convergence Diagnosis and Output Analysis Software for Gibbs Sampling Output, Version 0.4.* MRC Biostatistics Unit, University of Cambridge, Cambridge, UK. URL http://www.mrc-bsu.cam.ac.uk/bugs/classic/coda04/readme.shtml.

Gilks WR, Richardson S, Spiegelhalter DJ (1995). *Markov Chain Monte Carlo in Practice.* Chapman and Hall, New York.

SAS Institute Inc (1999). *SAS/STAT Software, Version 8.* Cary, NC. URL http://www.sas.com/.

Spiegelhalter DJ, Thomas A, Best NG, Lunn D (2003). ***WinBUGS** Version 1.4 User Manual.* MRC Biostatistics Unit, Cambridge. URL http://www.mrc-bsu.cam.ac.uk/bugs/.

Woodward P (2005). "**BugsXLA**: Bayes for the Common Man." *Journal of Statistical Software*, **14**(5). URL http://www.jstatsoft.org/v14/i05/.

Affiliation:

Michael K. Smith
Pharmacometrics Statistics
Pfizer Global Research and Development
Sandwich, CT13 9NJ, United Kingdom
E-mail: Mike.K.Smith@Pfizer.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 23, Issue 9 *Submitted:* 2005-12-16
December 2007 *Accepted:* 2007-09-07