Running Head: A SAS Interface for WinBUGS

A SAS Interface for Bayesian Analysis with WinBUGS

Zhiyong Zhang[1], John J. McArdle[2], Lijuan Wang[1], and Fumiaki Hamagami[3]

[1]University of Notre Dame [2]University of Southern California

[3]University of Virginia

To cite this paper, use

Zhang, Z., McArdle, J. J., Wang, L., & Hamagami, F. (2008). A SAS interface for Bayesian analysis with WinBUGS. *Structural Equation Modeling*, *15*(4), 705–728.

Corresponding Author: Zhiyong Zhang, Department of Psychology, University of Notre Dame, Notre Dame, IN 46556. E-Mail: zzhang4@nd.edu.

Abstract

Bayesian methods are becoming very popular despite some practical difficulties in implementation. To assist in the practical application of Bayesian methods, we show how to implement Bayesian analysis with WinBUGS as part of a standard set of SAS routines. This implementation procedure is first illustrated by fitting a multiple regression model and then a linear growth curve model. A third example is also provided to demonstrate how to iteratively run WinBUGS inside SAS for Monte Carlo simulation studies. The SAS codes used in the current study are easily extended to accommodate many other models with only slight modification. This interface can be of practical benefit in many aspects of Bayesian methods because it allows the SAS users to benefit from the implementation of Bayesian estimation and it also allows the WinBUGS user to benefit from the data processing routines available in SAS.

A SAS Interface for Bayesian Analysis with WinBUGS

Bayesian methods have received more and more attention in social and behavioral researches (e.g., Myung & Pitt, 1997; Seltzer & Choi, 2003; Lee, 2004) and these models have been successfully applied to item response models (e.g., Chang, 1996; Fox & Glas, 2001), factor analytic models (e.g., Bartholomew, 1981; Lee, 1981), structural equation models (e.g., Scheines, Hoijtink, & Boomsma, 1999; Congdon, 2003), genetic models (e.g., Eaves & Erkanli, 2003), growth curve models (e.g., Zhang, Hamagami, Wang, Grimm, & Nesselroade, 2007), and multilevel models (e.g., Seltzer, Wong, & Bryk, 1996). In a recent debate by Trafimow (2003, 2005) and Lee & Wagenmakers (2005), the advantages and disadvantages of Bayesian methods were discussed and a promising future of Bayesian applications has been suggested. Based on the review of applications of Bayesian methods in social and behavioral research, Rupp, Dey, and Zumbo (2004) further concluded that both applied and theoretical communities could not afford to miss the opportunities opened up by Bayesian methods.

A drawback to the implementation of Bayesian analysis and estimation is the programming and computation demands. However, with the development of the computation capacity, the cost of computation is acceptable given its benefits. Furthermore, the emergence of the free available WinBUGS software (Spiegelhalter, Thomas, Best, & Lunn, 2003) has made the programming much easier than before. WinBUGS is accepted as the most widely used and convenient tool for estimating both simple and complex Bayesian models (Congdon, 2001, 2003; Cowles, 2004).

A complete WinBUGS program consists of three parts: (1) Model Specification, (2) Data Input, and (3) Starting Values. Users first need to learn how to specify a model in

WinBUGS syntax and these specifications vary across different models. In this paper, we will illustrate how to specify three models: a multiple regression model, a growth curve model, and a confirmatory factor model. Because the contributed WinBUGS example programs for many models are available freely on the WinBUGS development website, we focus on helping readers understand WinBUGS codes and customize the codes for their own empirical data analysis.

The data format in the Data Input and Starting Values parts of WinBUGS is very similar to Splus / R data format (Spiegelhalter, Thomas, Best, & Lunn, 2003). Researchers who are not familiar to the Splus/R data format may find it difficult to transform data to be compatible with WinBUGS. Fortunately, there are some free programs or macros that can transform different formats of data to WinBUGS format, such as an R function R2WinBUGS by Sturtz and Ligges, a set of SAS macros by Sparapani, an excel macro xl2bugs by Misra, and a standalone program BAUW by Zhang and Wang (see Appendix A for Internet links).

Although WinBUGS can be used as menu-driving software, we present a batch procedure to call and run WinBUGS inside a SAS script. This procedure has several advantages over other methods. First, it is easy to run WinBUGS for researchers who are already familiar with SAS. Second, it permits a researcher to use SAS procedures before and after Bayesian modeling. It is easy to describe and transform data first in SAS, run the WinBUGS program for Bayesian estimation, and then save and plot results using SAS procedures. Third, although WinBUGS provides a “menu” to run the program, the “batch” processing approach used here decreases the probability of mistakes and allows users to easily repeat similar analyses. This procedure is especially useful and convenient

for analysis requiring the repetition, such as in Monte Carlo simulation studies. This procedure is demonstrated in the following sections.

A Complete Procedure to Run WinBUGS inside SAS

To run WinBUGS inside SAS, the procedure portrayed in Figure 1 can be followed. In the following sections, we describe this procedure using three examples: (1) a multiple regression model, (2) a linear growth curve model, and (3) a confirmatory factor analysis model. The first example aims to demonstrate how to apply the whole procedure in Figure 1, including how to specify a model, transform data, create starting values, and run WinBUGS inside SAS to implement Bayesian analysis. The second example is based on the linear growth curve model (e.g., Meredith & Tisak, 1990; McArdle & Nesselroade, 2003) and aims to illustrate how to specify a more complex model. The third example aims to show how to iteratively run WinBUGS inside SAS for the Monte Carlo simulation study.

--------- *Insert Figure 1 about here* ----------

Example 1. The Multiple Regression Model

For the purpose of demonstration, we use a multiple regression model with two predictors. The model can be written as

$$y[i] = b0 + b1 * x1[i] + b2 * x2[i] + e[i], i=1, \ldots, N.$$

In its probability form, this model can be expressed as

$$(1) \qquad y[i] \mid x1[i], x2[i] \sim N(\mu[i], \sigma_e^2)$$

$$\mu[i] = b0 + b1 * x1[i] + b2 * x2[i],$$

where σ_e^2 is the residual or measurement error variance, b0 is the intercept, and b1 and b2 are regression coefficients. A data set with the sample size N=1000 was generated from

this model with the population parameter values set as b0 = 1, b1 = 2, b2 = 3, and $\sigma_e^2 = 4$. The SAS codes for data generatation are given in **CODE 1**. The complete codes for running WinBUGS inside SAS to fit this multiple regression model are provided in **CODE 2** through **CODE 8**.

CODE 1.Data generatation

```
DATA Sim_Reg;
  b0=1; b1=2; b2=3; sig_e=2; seed=20060118; N = 1000;
  DO _N_ = 1 TO N;
      x1=RANNOR(seed);
      x2=RANNOR(seed);
      e=RANNOR(seed);
      y = b0+b1*x1+b2*x2+sig_e*e;
   KEEP y x1 x2;
   OUTPUT;
   END;
RUN;
```

CODE 2.Model specification

```
DATA model;
INPUT model $80.;
CARDS;/*start the model*/
model{
#Model specification
   for (i in 1:N) {
      y[i]~dnorm(muy[i], Inv_sig2_e)
     muy[i]<-b0+b1*x1[i]+b2*x2[i]
   }
#priors
   b0~dnorm(0, 1.0E-6)
   b1~dnorm(0, 1.0E-6)
   b2~dnorm(0, 1.0E-6)
   Inv_sig2_e~dgamma(1.0E-3, 1.0E-3)
#parameter transformation
   Sig2_e<-1/Inv_sig2_e
}
;
RUN;
DATA _NULL_;
  SET model;
  FILE "C:\SASWinBUGS\RegModel.txt";
  PUT model;
RUN;
```

CODE 3.Data transformation

```
%_lexport(data=Sim_Reg, file='C:\SASWinBUGS\RegData.txt',
var=y x1 x2);
```

CODE 4.Starting values specification

```
DATA _NULL_;
FILE "C:\SASWinBUGS\RegInit.txt";
PUT "list(b0=0, b1=0, b2=0, Inv_sig2_e=1)";
RUN;
```

CODE 5.Batch scripts to run WinBUGS

```
DATA _NULL_;
FILE "C:\program files\WinBUGS14\RegBatch.txt";
PUT // @@
#1 "display('log')"
#2 "check('C:/SASWinBUGS/RegModel.txt')"
#3 "data('C:/SASWinBUGS/RegData.txt')"
#4 "compile(1)"
#5 "inits(1, 'C:/SASWinBUGS/RegInit.txt')"
#6 "gen.inits()"
#7 "update(2000)"
#8 "set(b0)"
#9 "set(b1)"
#10 "set(b2)"
#11 "set(Sig2_e)"
#12 "dic.set()"
#13 "update(5000)"
#14 "dic.stats()"
#15 "coda(*,'C:/SASWinBUGS/output')"
#16 "save('C:/SASWinBUGS/bugslog.txt')"
#17 "quit()"
;
RUN;
```

CODE 6.Run WinBUGS in SAS X window

```
DATA _NULL_;
FILE "C:\SASWinBUGS\runreg.bat";
PUT 'CD C:\program files\WinBUGS14';
PUT 'WinBUGS14.exe /PAR RegBatch.txt';
PUT 'EXIT';
RUN;

DATA _NULL_;
X "C:\SASWinBUGS\runreg.bat";
RUN; QUIT;
```

CODE 7.View log file and DIC and debug errors

```
DATA log;
INFILE "C:\SASWinBUGS\bugslog.txt" TRUNCOVER;
INPUT log $80.;
log=translate(log," ","09"x);
RUN;

PROC PRINT DATA=log;
RUN;
```

CODE 8.Statistical inference

```
%coda2sas(out=coda, infile='C:\
SASWinBUGS\outputIndex.txt',
chain='C:\SASWinBUGS\output1.txt', stats=1);
QUIT;
```

Step 1: Install SAS and WinBUGS

The first step is to install SAS and WinBUGS. Since SAS is widely used, we assume that SAS has been installed and only focus on the installation of WinBUGS. WinBUGS is free software and can be downloaded from its website (Appendix A). The download is an executable file and it can be setup as a usual "double click and follow screen instructions" Windows program. WinBUGS requires a key for its unrestricted use. The key is sent by e-mail once the user completes a registration form on the WinBUGS website.

Step 2: Setup SAS environment for WinBUGS

A set of free SAS macros written by Sparapani (Appendix A) can be used to transform data from SAS data format to WinBUGS data format. The macros can be downloaded from their website (See Appendix A, #4). A slightly modified version was used in this article, which is also freely available (Appendix A, #7). The following instructions can be followed to setup the macros.

1. Put the macros into a folder. To avoid unintentionally deleting these macros, we suggest putting the macros into the folder "`C:\Program Files\SAS\bugs`". "`bugs`" is a new folder that needs to be created first.

2. Modify the SAS configuration file, sasv8.cfg for SAS 8.x or sasv9.cfg for SAS 9.x. Open the file in notepad and add the two lines below at the end and save it.

```
-insert sasautos 'C:\Program Files\SAS\bugs'
-insert sasautos '!SASROOT\core\sasmacro'
```

Step 3: Express the model in WinBUGS language

To implement Bayesian analysis in WinBUGS, we first need to express the models using WinBUGS syntax. For a typical WinBUGS program, the model specification part must include two sub-parts: the expression of the model and the choice of prior distributions (Spiegelhalter, Thomas, Best, & Lunn, 2003). For the regression model, the codes are given in **CODE 2**.

All WinBUGS programs start with a keyword `model` and the whole model specification part needs to be put within a pair of brackets `{ }`. The first section of the model specification part can be viewed as the direct translation of the probability form of the model. For the regression model in Equation (2), WinBUGS codes for the *i*th individual were

`y[i] ~ dnorm(muy[i], Inv_sig2_e)`, and

`muy[i] <- b0 + b1*x1[i] + b2*x2[i]`.

The first line indicates that `y[i]` had (`~`) a normal distribution (`dnorm`) with two arguments: mean `muy[i]` and precision `Inv_sig2_e`. The precision in the second arguments is the reciprocal of the variance ($1/\sigma_e^2$). The mean was equal to (`<-`) the combination of the two predictor `x1` and `x2` with the regression coefficients `b1` and `b2`. Because we had N=1000 individuals, we used a `for (i in 1:N){…}` loop to repeat this specification for each individual. `For` is the keyword for a loop. `i in 1:N` in the parentheses means replacing `i` using `1, 2, …, N` and then implement everything in the

brackets {} following `for (i in 1:N)`.

In the second section of the model specification part, we need to choose a prior distribution for each parameter in the model. For this regression model, there were 4 parameters, `b0`, `b1`, `b2`, and `Inv_sig2_e`. For the regression intercept and regression coefficients, the normal distribution priors with mean 0 and precision 1.0E-6 were specified as `bk~dnorm(0, 1.0E-6)` with `k=0,1,2`, which is a widely used non-informative prior (Congdon, 2001, 2003). For the precision parameter, a Gamma distribution (`dgamma`) prior with shape and scale parameter = 1.0E-3 was used `Inv_sig2_e~dgamma(1.0E-3, 1.0E-3)`, which is also a widely used non-informative prior for the variance parameter (Congdon, 2001, 2003). Finally, we transformed the precision back to the variance.

By running the codes in **CODE 2**, the WinBUGS model specification codes for the multiple regression model were saved into a file called '`RegModel.txt`'.

Step 4: Configure a SAS program to run WinBUGS

In this step, we configured a SAS program to create WinBUGS compatible data, construct a staring value file, run WinBUGS, and analyze the MCMC data generated by WinBUGS. The following 5 steps can be done in SAS.

Step 4-1: Create a WinBUGS data file

By using the SAS macros in Step 2, a SAS data set can be converted to the WinBUGS format. WinBUGS uses a keyword `list(…)` to organize the data. The data in the parentheses can be a scalar, a vector, or an array. For a scalar, the format is `ScalarName=data`. For example, `N=1000` and `T=5`. For a vector, the format is `VectorName=c(data1, data2, …, dataN)`. The keyword `c` combines the values

separated by the comma in the parentheses into a vector. For example, `Mu=c(0,0)` is a vector with 2 elements. For an array, the format is `ArrayName=structure(.Data = c(data1, data2, … ), .Dim = c(nrow, ncol,…) )`. WinBUGS reads data in `.Data = c(data1, data2, … )` into an array by filling the right-most index for dimensions in `.Dim = c(nrow, ncol,…)` first. For example, for a two-dimension (3×2) array,

$$\texttt{y=structure(.Data=c(1,2,3,4,5,6), .Dim=c(3,2))}=\begin{pmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{pmatrix}.$$

The SAS macros set up in Step 2 can convert SAS data into a `list` with `vector` (`_lexport`) or `array` (`_sexport`). Using the codes in **CODE 3**, a SAS data set was converted to WinBUGS vector data and saved into a file. In this example, the macro `_lexport` was used. `data=` defines the SAS data set to be used. Here it was a SAS data set called `Sim_Reg`. `file=` defines the file to save the data in WinBUGS data format. `var=` defines the variables to transform. In the generated data file, three vectors, `y`, `x1`, and `x2` along with a scalar representing the sample size (`N=1000`) were saved.

Step 4-2: Create a starting value file

For each parameter in the model, a starting value needs to be either specified manually or generated by WinBUGS. Usually, for each canonical parameter, we give it a starting value manually and let WinBUGS generate the others. The format of the starting values has the same format as the data. The same way to transform data can be used to create starting values. Usually, the starting values are relatively easier to handle and can be put together directly. The SAS codes in **CODE 4** created a starting value file for the

regression model. All the staring values were put together in a `list` and saved in the file "`RegInit.txt`".

Step 4-3: Create a script file to run WinBUGS

In this step, we used the batch mode to run WinBUGS. **CODE 5** provides the SAS codes for the regression model.

Line #1 opened a log window to trace the history and errors of the implementation process. Line #2 checked whether the syntax in the file '`RegModel.txt`' was correct. Line #3 read in the data, Line #4 compiled the model, Line #5 initialized the parameters using the data in the starting value file, and Line #6 generated the staring values for the parameters that were not specified in the starting value file. Line #7 generated 2000 data points for each parameter but these data points (called burn-in data points) were discarded to ensure convergence. In Lines #8–11 the parameters were specified to be estimated. The parameters need to be specified since the sampled data points for unspecified parameters will not be saved. Line #12 was used to monitor the DIC (Spiegelhalter et. al. 2002) that can be used as a fit statistic to compare models. Line #13 generated the other 5000 data points which were saved to be analyzed in SAS for statistical inference. Line #14 wrote the DIC into the log file. Line #15 saved the generated data points in Line #13 into two files. The first file was the index file '`outputIndex.txt`' that included the index for each estimated parameter defined in Lines #8-11. The second file was the data file '`output1.txt`' that saved the data points generated. These files were also called CODA (Convergence Diagnostic and Output Analysis) files. Line #16 saved the log file where the DIC and error information can be found. Line #17 was used to quit the WinBUGS program when finished.

Three comments are worth emphasizing here. First, this script file must be saved where WinBUGS is installed, usually "`C:\program files\WinBUGS14\`". Second, one can change the folder where the model file, data file, staring value file, coda file, and log file are saved. However, the slash "`/`" instead of the usual backslash "`\`" needs to be used in the path. Third, WinBUGS is sensitive to the lowercase and uppercase letters.

Step 4-4: Run WinBUGS and debug errors

To run WinBUGS, we first created a `run.bat` file using the codes in the first paragraph of **CODE 6**. We then ran WinBUGS in the X window using the codes in the second paragraph. After implementing the first two paragraphs of the codes, a DOS window opened and the WinBUGS program implemented the procedure specified in Step 4-3. After finished, the WinBUGS program exited and the SAS window returned. Then the first thing to check is the log file. The codes in **CODE 7** can read the log file and print its content in the output window of SAS. Any errors in running WinBUGS can be targeted by the information provided in the log file.

Step 4-5: Read CODA files into SAS and implement statistical inferences

The CODA file generated by WinBUGS can be read into SAS using the macro `coda2sas` with the codes in **CODE 8**. The first argument "`out=`" specified the name of a SAS data set to save the generated data points. Here a data set called "`coda`" was created. "`infile=`" specified the index file and "`chain=`" specified the data file saved in Step 4-3. By specifying "`stats=1`", the macro `coda2sas` generated the history plot and the histogram with the overlaid kernel density and calculated the descriptive statistics for each parameter. If more analyses are needed, one can work on the data set "`coda`". For the regression model, the history and histogram plots for the regression model are

given in Figure 2. From the history plots, the generated sequences for all parameters converged through the "eyeball" check. The density plots and descriptive statistics for the four model parameters are given in Figure 2 and Table 1, respectively. From Table 1, the estimated parameters were very close to the population values used to generate the data.

---------- Insert Figure 2 and Table 1 about here -------

Example 2. The Linear Growth Curve Model

We have shown how to use the procedure in Figure 1 to estimate a multiple regression model. In this example, we present a linear growth curve model to demonstrate how to specify a more complex model using WinBUGS syntax. The linear growth curve model (e.g., McArdle & Nesselroade, 2003) can be written as

$$\begin{aligned} y[i,t] &= L[i] + t \times S[i] + e[i,t] \\ L[i] &= \mu_L + v_L[i] \qquad\qquad i = 1,\cdots,\mathrm{N};\ t = 1,\cdots,\mathrm{T}, \\ S[i] &= \mu_S + v_S[i] \end{aligned}$$

where $y[i,t]$ represents the observed score for the ith individual at occasion t, $L[i]$ represents the level and $S[i]$ represents the slope for the ith individual, $e[i,t]$ represents the measurement error, μ_L and μ_S are the average level and slope of N individuals, and $v_L[i]$ and $v_S[i]$ are the individual deviances for the initial level and slope from the average level and slope for ith individual.

Using probability density function, this model can be expressed as

$$\begin{aligned} & y[i,t] \mid L[i], S[i] \sim N(\mu[i,t], \sigma_e^2) \\ (2)\quad & \mu[i,t] = L[i] + t \times S[i] \qquad\qquad i = 1,\cdots,\mathrm{N};\ t = 1,\cdots,\mathrm{T}, \\ & \begin{pmatrix} L[i] \\ S[i] \end{pmatrix} \sim MN\left(\begin{bmatrix} \mu_L \\ \mu_S \end{bmatrix}, \begin{bmatrix} \sigma_L^2 & \sigma_{LS} \\ \sigma_{LS} & \sigma_S^2 \end{bmatrix} \right) \end{aligned}$$

where N represents the univariate normal distribution, σ_e^2 represents the variance of measurement errors, MN represents the multivariate normal distribution, σ_L^2 and σ_S^2

represent the variances of the level and slope respectively, and σ_{LS} is the covariance between the level and slope. Equation (2) indicates that the level and slope have a bivariate normal distribution and the observed variable has a univariate normal distribution with the mean expressed as the combination of the level and slope.

Based on this linear growth curve model, we simulated a data set with N=1000 participants, and T=5 occasions using SAS with the population parameter values μ_L=10, μ_S =5, σ_e^2 =1, σ_L^2=4, σ_S^2=1, and σ_{LS}=1 or ρ_{LS}=.5. Since the maximum likelihood estimation (MLE) has commonly applied to obtain parameter estimates for the linear growth curve model (e.g., Demidenko, 2004; Laird & Ware, 1982), we briefly compare the results from Bayesian estimation (BE) with those from MLE.

For the linear growth curve model, the model specification part is given in **CODE 9**.

CODE 9. Model specification of the linear growth model

```
DATA model;
INPUT model $80.;
CARDS;/*Start of the model scripts*/
 model
 {#Model
    for (i in 1:N){
       LS[i,1:2]~dmnorm(Mu[1:2], Inv_cov[1:2,1:2])
          for (t in 1:T){
              y[i,t]~dnorm(MuY[i,t], Inv_sig2_e)
              MuY[i,t]<-LS[i,1]+LS[i,2]*t
           }
     }
    #Prior
    Mu[1:2]~dmnorm(Mu0[1:2], Inv_cov0[1:2,1:2])
    Mu0[1]<-0
    Mu0[2]<-0
    Inv_cov0[1,1]<-1.0E-6
    Inv_cov0[2,2]<-1.0E-6
    Inv_cov0[2,1]<-Inv_cov0[1,2]
    Inv_cov0[1,2]<-0

    Inv_cov[1:2,1:2]~dwish(R[1:2,1:2], 2)
    R[1,1]<-1
```

```
      R[2,2]<-1
      R[2,1]<-R[1,2]
      R[1,2]<-0

      Inv_sig2_e~dgamma(.001,.001)

      #Transform of the parameters
      MuL<-Mu[1]
      MuS<-Mu[2]
      Cov[1:2,1:2]<-inverse(Inv_cov[1:2,1:2])
      Sig2_L<-Cov[1,1]
      Sig2_S<-Cov[2,2]
      rho<-Cov[1,2]/sqrt(Cov[1,1]*Cov[2,2])
      Sig2_e<-1/Inv_sig2_e
    }
    ;
  /*end of the model scripts*/
  RUN;

  DATA _NULL_;
    SET model;
    FILE 'C:\SASWinBUGS\GrowthModel.txt';
    PUT model;
  RUN;
```

Since the level and slope are bivariate normally distributed, we need to specify a bivariate normal distribution for them which is `dmnorm` in WinBUGS. The bivariate normal distribution has two augments: mean vector and covariance matrix. In WinBUGS, the second augment for `dmnorm` is the precision which is the inverse of the covariance matrix. In WinBUGS, this bivariate distribution was expressed as

```
LS[i,1:2]~dmnorm(Mu[1:2], Inv_cov[1:2,1:2])
```

where `LS[i,1:2]` is a 2×1 vector with two elements `LS[i,1]` and `LS[i,2]`; `LS[i,1]` is the level and `LS[i,2]` is the slope for individual *i*; `Mu[1:2]` is a 2×1 mean vector; and `Inv_cov[1:2,1:2]` is the inverse of the covariance matrix of the initial level and slope. Since we had N=1000 individuals, we used a `for (i in 1:N){…}` loop to repeat this specification for each individual. The observed variable `y` had a

univariate normal distribution which was expressed as

```
y[i,t]~dnorm(MuY[i,t], Inv_sig2_e).
```

Because each individual had an observation from occasion 1 to T=5, we used a second loop `for (t in 1:T)` nested in the first one to represent this.

For the linear growth model, there were 6 parameters, `Mu[1,2]`, `Inv_cov[1:2,1:2]`, and `Inv_sig2_e`. We gave the mean vector `Mu[1,2]` a bivariate normal distribution prior. For the precision matrix (`Inv_cov[1:2,1:2]`), a Wishart distribution prior was used since it is the multivariate generalization of the Gamma distribution. For the precision of `y`, a Gamma distribution prior was used.

By running the codes in **CODE 9**, the WinBUGS scripts for the growth curve model were saved into a file called '`GrowthModel.txt`'.

In this example, we used the N×T (1000×5) array data `y[i,t]`. Using the scripts in **CODE 10**, the SAS data set `Sim_LinGM` was converted to WinBUGS array data using macro `_sexport` and saved into a file called `GrowthData.txt`. All the starting values were put together in a `list` and saved in the file "`InitValues.txt`". Note that this set of starting values included all three types of data.

CODE 10. Data transformation and starting values for the linear growth curve model

```
%_sexport(data=Sim_LinGM,
        file ='C:\SASWinBUGS\GrowthData.txt',
        var =y1-y5);

DATA _NULL_;
FILE "C:\SASWinBUGS\InitValues.txt";
PUT "list(Mu=c(0,0), Inv_cov= structure(.Data =
     c(1,0,0,1),.Dim=c(2,2)), Inv_sig2_e=1) ";
```

The batch scripts and the `.bat` file for this linear growth curve model were very similar to those for the regression model and we did not repeat them here. The complete

SAS codes for this model are available by request. After running the complete SAS codes, all parameter estimates from WinBUGS along with those from SAS MIXED are summarized in Table 2. From Table 2, the parameter estimates from WinBUGS were very close to the population values. Furthermore, the parameter estimates from WinBUGS and SAS MIXED were nearly identical, which demonstrates that Bayesian method estimation provides the same level of accuracy as MLE when non-informative priors are used.

--------- *Insert Table 2 about here* ----------

Example 3. Monte Carlo Simulation of a Confirmatory Factor Model

Bayesian methods have been mainly used as an alternative to MLE or to estimate complex models which usually cannot be easily estimated with MLE. Simulation studies are necessary when evaluating new or complex models. WinBUGS is not very flexible for simulation studies because it can only run a single model or a single data set at one time. However, SAS can be sued to iteratively implement the simulation procedure. To demonstrate how to use SAS to iteratively run WinBUGS, we use a confirmatory factor model with one latent factor and four observed variables. A path diagram with the population parameter values is plotted in Figure 3. We generated 100 sets of data from the population models and parameter estimates were obtained for each data set using WinBUGS. We compared the mean of the parameter estimates from all 100 sets of data with the population values.

---------- Insert Figure 3 about here----------

For the simulation study, the model specification, starting values, and script file to run WinBUGS are the same for each data set. The following codes in **CODE 11** can be used to set up those for the confirmatory factor model. In this example, a new WinBUGS

command "`stat()`" was used, which calculated the summary statistics for each parameter inside WinBUGS.

CODE 11.Common scripts for the CFA simulation

```
TITLE "Model specification for the CFA";
FILENAME model "C:\SASWinBUGS\cfamodel.txt";
DATA model;
  INPUT model $80.;
  CARDS;/*start the model*/
  model{
    for (i in 1:N){
       for (t in 1:T){
           y[i,t]~dnorm(muy[i,t],Inv_sig2[t])
           muy[i,t]<-fload[t]*fscore[i]
        }
    fscore[i]~dnorm(0, 1)
    }
    for (t in 1:T){
      fload[t]~dnorm(0, 1.0E-6)I(0,)
      Inv_sig2[t]~dgamma(0.001, .001)
      Para[t]<-fload[t]
      Para[t+4]<-1/Inv_sig2[t]
    }
  }
  ;
RUN;
DATA _NULL_;
  SET model;
  FILE model;
  PUT model;
RUN;

TITLE "Starting values for CFA";
DATA _NULL_;
  FILE "C:\SASWinBUGS\cfaini.txt";
  PUT "list(fload=c(.5,.5,.5,.5), Inv_sig2=c(1,1,1,1))";
RUN;

TITLE "Batch scripts to run WinBUGS";
FILENAME runcfa 'c:\program files\winbugs14\runcfa.txt';
DATA _NULL_;
  FILE runcfa;
  PUT@1 "display('log')";
  PUT@1 "check('C:/SASWinBUGS/cfamodel.txt')" ;
  PUT@1 "data('C:/SASWinBUGS/cfadata.txt')";
  PUT@1 "compile(1)";
```

```
  PUT@1 "inits(1, 'C:/SASWinBUGS/cfaini.txt')";
  PUT@1 "gen.inits()";
  PUT@1 "update(2000)";
  PUT@1 "set(Para)";
  PUT@1 "update(3000)";
  PUT@1 "stats(*)";
  PUT@1 "save('C:/SASWinBUGS/cfalog.txt')";
  PUT@1  "quit()";
RUN;

DATA _NULL_;
  FILE "C:\SASWinBUGS\runcfa.bat";
  PUT '"C:\program files\WinBUGS14\WinBUGS14.exe" /PAR
       runcfa.txt';
  PUT 'exit';
RUN;
```

For the Monte Carlo simulation study, each data set generated from the population model was different and the parameter estimates from each data set generally were likely to be somewhat different. Thus, we need to generate multiple data sets and estimate the parameters for each data set iteratively. To do this, we used a macro that can be called iteratively. Each time this macro was called, it generated a data set and obtained parameter estimates from the model. The macro `simcfa(n)` for the CFA is given in **CODE 12**. In the first part, a data set was generated from the population model. Then this data set was saved into a file "`cfadata.txt`" in WinBUGS data format. In the next part, WinBUGS was run in X window to implement the Bayesian analysis based on this generated data set. Finally, the log file was read into SAS to obtain the parameter estimates. Notice that we did not save the CODA files and calculate the parameter estimates in SAS. Instead, we read in the parameter estimates from the log file directly. In this case, we need to make sure the generated sequences converged. In the current example, we first ran one set of data and found that the generated sequences for all parameters converged after 100 iterations. Although we may use 100 as the burn-in data

points, we used 2000 to ensure the convergence for all the other data sets.

CODE 12. The macro for data generation and model estimaiton

```
%MACRO simcfa(n);
TITLE 'Generate the Data';
DATA Sim_CFA;
*setting the true parameter values;
fload=.8; sig2=.36;
* setting statistical parameters;
  N = 200; seed = 20060802+&n; M=4;
* need to setup arrays so we can have more variables;
ARRAY y_score{4} y1-y4;
ARRAY e_score{4} y1-y4;

* generating raw data;
  DO _N_ = 1 TO N;
* now the indicator variables ;
     f_score=RANNOR(seed);
      DO t = 1 TO M;
         y_score{t} = fload*f_score
+sqrt(sig2)*RANNOR(seed);
   END;
   KEEP y1-y4;
   OUTPUT;
   END;
RUN;

/*Data*/
%_sexport(data=Sim_CFA, file='C:\SASWinBUGS\cfadata.txt',
var=y1-y4);


/*Run WinBUGS*/
DATA _NULL_;
   X "C:\SASWinBUGS\runcfa.bat";
RUN;
QUIT;

/*Read in the log file to view the parameters*/
TITLE 'Simulation '&n;
DATA log;
   INFILE "C:\SASWinBUGS\cfalog.txt" TRUNCOVER ;
   INPUT log $80.;
   log=translate(log," ","09"x);
   IF (SUBSTR(log, 2, 4) ne 'Para') then delete;
RUN;
```

```
PROC PRINT DATA=log;
RUN;
%MEND;
```

To run the macro above 100 times to generate 100 data sets and obtain 100 sets of parameters, we configured another macro `runsimcfa` which is given in **CODE 13**.

CODE 13.The macro for running simulation iteratively

```
%MACRO runsimcfa;
   %LET n=1;
      %DO %WHILE(&n <= 100);
         %simcfa(&n);
      %LET n=%EVAL(&n+1);
   %END;
%MEND runsimcfa;
*run the macro
%runsimcfa;
```

After running the codes in **CODE 13**, the 100 sets of parameter estimates were printed in the SAS output window. Usually, for each parameter, we need to calculate 3 numbers: the mean and the standard deviation (s.d.) of each parameter estimate, and the mean of the associated standard errors (MSE) from the 100 sets of data. All of these can be calculated using the SAS codes in **CODE 14**.

CODE 14. Data process of the simulation results

```
/*Save the output and log into files*/
DM OUTPUT 'FILE "C:\SASWinBUGS\allresults.txt"';
DM LOG   'FILE "C:\SASWinBUGS\allresults.log"';

TITLE “Analyze the Monte Carlo simulation results”;
DATA temp;
  INFILE "C:\SASWinBUGS\allresults.txt" TRUNCOVER ;
  INPUT all $90.;
  IF (SUBSTR(all, 7, 4) NE 'Para')  THEN DELETE;
  FILE "C:\SASWinBUGS\temp.txt";
  PUT all;
RUN;

DATA temp;
INFILE  "C:\SASWinBUGS\temp.txt";
INPUT parid parname $ parest parsd MCerror p25 median p975
start sample;
```

```
id=int((_N_-.1)/8)+1;
parest=abs(parest);
RUN;

/*Parameter Estimates*/
PROC TRANSPOSE DATA=temp OUT=parest PREFIX=par;
   BY id ;
   ID parid;
   VAR parest;
RUN;
/*Calculate the mean and s.d. of the parameters*/
PROC MEANS DATA=parest;
VAR par1-par8;
RUN;

/*SDs*/
PROC TRANSPOSE DATA=temp OUT=parsd PREFIX=sd;
   BY id ;
   ID parid;
   VAR parsd;
RUN;
/*Calculate the mean of the s.e.*/
PROC MEANS DATA=parsd;
VAR sd1-sd8;
RUN;
```

After running **CODE 11** through **CODE 14**, we can obtain the results in Table 3. The means of the parameter estimates are very close to the population parameter values to generate the data. Furthermore, the standard deviations were the same as the MSE indicating the estimated standard errors were consistent with the true standard errors.

----------Insert Table 3 about here----------

Discussion

WinBUGS is a powerful tool for implementing Bayesian analysis and estimating complex models (Rupp et al., 2004) and SAS is widely used statistical software in academic and research institutes. Their combination will advance the application of both Bayesian methods and sophisticated models in social and behavioral research. The whole procedure we presented and the SAS codes we provided can conveniently interface SAS

and WinBUGS. This procedure is beneficial to many researchers including advanced Bayesian users who already have rich experiences in Bayesian analysis and researchers who are familiar with SAS but yet to find out the utility of Bayesian approach.

The procedure in Figure 1 was illustrated using a multiple regression model, a linear growth curve model, and a confirmatory factor model. In the first example, we demonstrated how to apply the proposed interface between SAS and WinBUGS using a multiple regression model. The second example showed how to specify a more complex model and the last example focused on the iterative use of WinBUGS for Monte Carlo simulation studies. All three examples can be replicated and modified to accommodate new models.

Two concerns about the Bayesian analysis include the computational time and programming intensity. However, with the availability of powerful computing facilities, the computing time is not a big problem any more. For example, the multiple regression model took about 10 seconds to finish the estimation procedure on an “out-of-dated” laptop (Celeron 1.7 and 512M RAM). The growth curve model with N=1000 and T=5 took about 120 seconds. For the confirmatory factor model example, it took only 30 minutes to finish the whole simulation study (with 100 sets of data). Furthermore, although the different models need different model specification, the example WinBUGS codes for many models can be obtained freely. By simply replacing the model specification part in our example and with a few other minor changes, a new data set can be analyzed by a new model.

For the aim of illustration, we only used three relatively simple models as examples. However, the same procedure allows and shows advantages when estimating more

complex models which cannot be analyzed in SAS easily, such as the change point models (McArdle & Wang, 2007; Wang & McArdle, under review), dynamic item response models (Ram et al., 2005) and categorical dynamic factor models (Zhang & Nesselroade, 2007). Complexity of models also made the difference of computation time less noticeable between Bayesian and MLE methods. However, the precision of the parameter estimates is even better for Bayesian methods.

To close the discussion, we would like to evaluate the proposed procedure based on our practical experience. First, this procedure is very useful for analyzing data using the similar model. For example, when we analyzed the cognitive data using the same linear growth curve model, we only need to import the data into SAS and run the exact same procedure without changing anything except the name of the data set. Second, this procedure is especially useful for simulation studies. It is well known that WinBUGS is not flexible for simulation studies because it can run only single replication at one time. Our procedure can be viewed as a useful supplement to WinBUGS. This procedure has been proved useful in Bowels (2006), Zhang, Hamaker, and Nesselroade (in press), and Zhang and Nesselroade (2007) for different simulation studies.

References

Bartholomew, D. J. (1981). Posterior analysis of the factor model. *British Journal of Mathematical and Statistical Psychology, 34,* 93–99.

Bowels, R. P. (2006). Item response models for intratask change to examine the impacts of proactive interference on the aging of working memory span. Doctoral dissertation. Department of Psychology, University of Virginia.

Chang, H.-H. (1996). The asymptotic posterior normality of the latent trait for polytomous IRT models. *Psychometrika, 61*, 445–463.

Cowles, M. K. (2004). Review of WinBUGS 1.4. *The American Statistician, 58* (4), 330-336.

Congton, P. (2001). *Bayesian statistical modeling*. New York: Wiley.

Congton, P. (2003). *Applied Bayesian modeling*. New York: Wiley.

Demidenko, E. (2004). *Mixed models: Theory and applications*. New York: Wiley.

Edwards,W., Lindman, H., & Savage, L. J. (1963). Bayesian statistical inference for psychological research. *Psychological Review, 70,* 193–242.

Eaves, L. & Erkanli, A. (2003). Markov chain Monte Carlo approaches to analysis of genetic and environmental components of human developmental change and G X E interaction. *Behavior Genetics, 33,* 279-299

Fox, J.-P., & Glas, C. A. W. (2001). Bayesian estimation of a multilevel IRT model using Gibbs sampling. *Psychometrika, 66,* 271–288.

Laird, N. M., & Ware, J. H. (1982). Random-effects models for longitudinal data. *Biometrics, 38* (4), 963-974.

Lee, S. (1981). A Bayesian approach to confirmatory factor analysis. *Psychometrika, 46,*

153–160.

Lee, M. D. (2004). A Bayesian analysis of retention functions. *Journal of Mathematical Psychology*, *48*(5), 310-321.

Lee, M. D., & Wagenmakers, E. (2005). Bayesian statistical inference in psychology: Comment on Trafimow (2003). *Psychological Review, 112*, 662–668.

McArdle, J. J., & Nesselroade, J. R. (2003). Growth curve analysis in contemporary psychological research. In J. Schinka & W. Velicer (Eds.), *Comprehensive handbook of psychology: Research methods in psychology* (Vol. 2, p. 447-480). New York: Wiley.

McArdle, J. J. & Wang, L. (2007). Modeling age-based turning points in longitudinal Life-span growth curves of cognition. In P. Cohen (Ed.), *Turning points research*, Mahwah: Erlbaum

Meredith, W. & Tisak, J. (1990). Latent curve analysis. *Psychometrika, 55*, 107-122.

Myung, I. J., & Pitt, M. A. (1997). Applying Occam's razor in modeling cognition: A Bayesian approach. *Psychonomic Bulletin & Review, 4*(1), 79–95.

Ram, N., Chow, S., Bowles, R.P., Wang, L., Grimm, K., Fujita, F., & Nesselroade, J.R. (2005). Examining interindividual differences in cyclicity of pleasant and unpleasant affect using spectral analysis and item response modeling. *Psychometrika, 70*(4), 773-790.

Rupp, A. A., Dey, D. K., & Zumbo, B. D. (2004). To Bayes or not to Bayes, from whether to when: Applications of Bayesian methodology to modeling. *Structural Equation Modeling: A Multidisciplinary Journal, 11*(3), 424-451.

Scheines, R., Hoijtink, H., & Boomsma, A. (1999). Bayesian estimation and testing of

structural equation models. *Psychometrika, 64*, 37–52.

Seltzer, M. H., Wong, W. H., & Bryk, A. S. (1996). Bayesian analysis in applications of hierarchical models: Issues and methods. *Journal of Educational and Behavioral Statistics, 21*, 131–167.

Seltzer, M., & Choi, K. (2003). Sensitivity analysis for hierarchical models: Downweighting and identifying extreme cases using the t distribution. In S. P. Reise & N. Duan (eds). *Multilevel modeling: Methodological advances, issues, and applications* (p25-52). Mahwah, NJ: Lawrence Erlbaum Associates.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Linde, A. v. d. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64* (4), 583-639.

Spiegelhalter, D. J., Thomas, A., Best, N., & Lunn, D. (2003). WinBUGS Manual Version 1.4. (MRC Biostatistics Unit, Institute of Public Health, Robinson Way, Cambridge CB2 2SR, UK, http://www.mrc-bsu.cam.ac.uk/bugs)

Trafimow, D. (2003). Hypothesis testing and theory evaluation at the boundaries: Surprising insights from Bayes's theorem. *Psychological Review, 110*, 526–535.

Trafimow, D. (2005). The ubiquitous Laplacian assumption: Reply to Lee and Wagenmakers (2005). *Psychological Review, 112*, 669–674.

Wang, L. & McArdle, J. J. (1st revision under review). Estimating unknown change points by using Bayesian methods. *Structural Equation Modeling*.

Zhang, Z., Hamagami, F., Wang, L., Grimm, K. J., & Nesselroade, J. R. (2007). Bayesian analysis of longitudinal data using growth curve models. *International Journal of Behavioral Development 31*(4), 374-383.

Zhang, Z., Hamaker, E. L., & Nesselroade, J. R. (in press). Comparisons of four methods for estimating dynamic factor models. *Structural Equation Modeling*.

Zhang, Z., & Nesselroade, J. R. (2007). Bayesian estimation of categorical dynamic factor models. *Multivariate Behavioral Research*, *42*(4), 729-756.

Appendix

Appendix A. List of the programs or macros

1. SAS: http://www.sas.com

2. WinBUGS: http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml

3. R2WinBUGS by Sibylle Sturtz and Uwe Ligges:

http://cran.r-project.org/src/contrib/Descriptions/R2WinBUGS.html

4. SAS Macros by Rodney Sparapani: http://www.mcw.edu/pcor/bugs/

5. xl2bugs by Sanjog Misra: http://smisra.simon.rochester.edu/software.htm

6. BAUW by Zhiyong Zhang and Lijuan Wang: http://bauw.psychstat.org

7. Modified version of Rodney Sparapani's SAS Macros: http://bauw.psychstat.org

Table 1.

The parameter estimates for the regression model.

	True	estimate	s.e.	CI
b0	1	0.99	0.064	(0.87, 1.12)
b1	2	2.04	0.065	(1.92, 2.17)
b2	3	3.01	0.064	(2.89, 3.14)
sig2_e	4	4.06	0.184	(3.71, 4.45)

Note. s.e.: standard error; CI: confidence interval.

Table 2.

Parameter estimates for the linear growth model

		WinBUGS		SAS MIXED	
	True Value	Estimate	s.e.	Estimate	s.e.
μ_L	10	10.12	0.071	10.12	0.071
μ_S	5	5.01	0.033	5.01	0.034
σ_L^2	4	3.96	0.227	3.96	0.226
σ_S^2	1	1.04	0.051	1.04	0.051
σ_e^2	1	0.97	0.025	0.97	0.025
ρ_{LS}	0.5	0.45	0.034	0.44	0.038

Note. s.e.: standard error.

Table 3.

Results from the confirmatory factor analysis

Parameters	TRUE	Mean	s.d.	MSE
Factor Loadings	0.8	0.81	0.06	0.06
	0.8	0.82	0.06	0.06
	0.8	0.80	0.06	0.06
	0.8	0.81	0.06	0.06
Uniqueness Variances	0.36	0.35	0.04	0.05
	0.36	0.37	0.05	0.05
	0.36	0.38	0.05	0.05
	0.36	0.36	0.05	0.05

Note. Mean: the average value of the parameter estimates from 100 sets of simulated data. s.d.: the standard deviation of the parameter estimates from 100 sets of simulated data. MSE: the average standard errors of the parameter estimates from 100 sets of simulated data.

Figure Captions

Figure 1. The flow chart to run WinBUGS inside SAS

Figure 2. History plots and Histogram plots of parameters from the regression model

Figure 3. Path diagram for the population confirmatory factor model

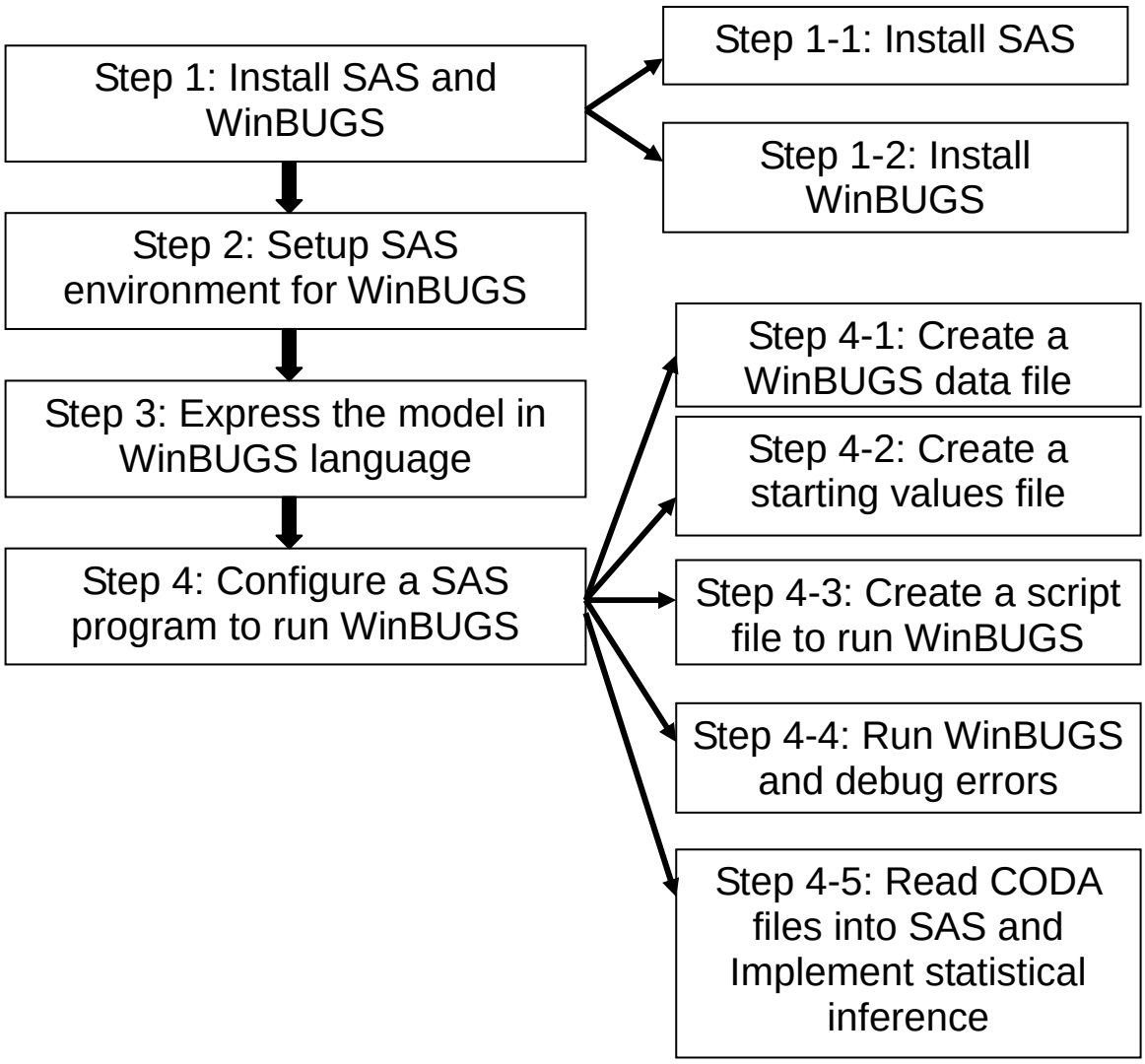

Figure 1. The flow chart to run WinBUGS inside SAS

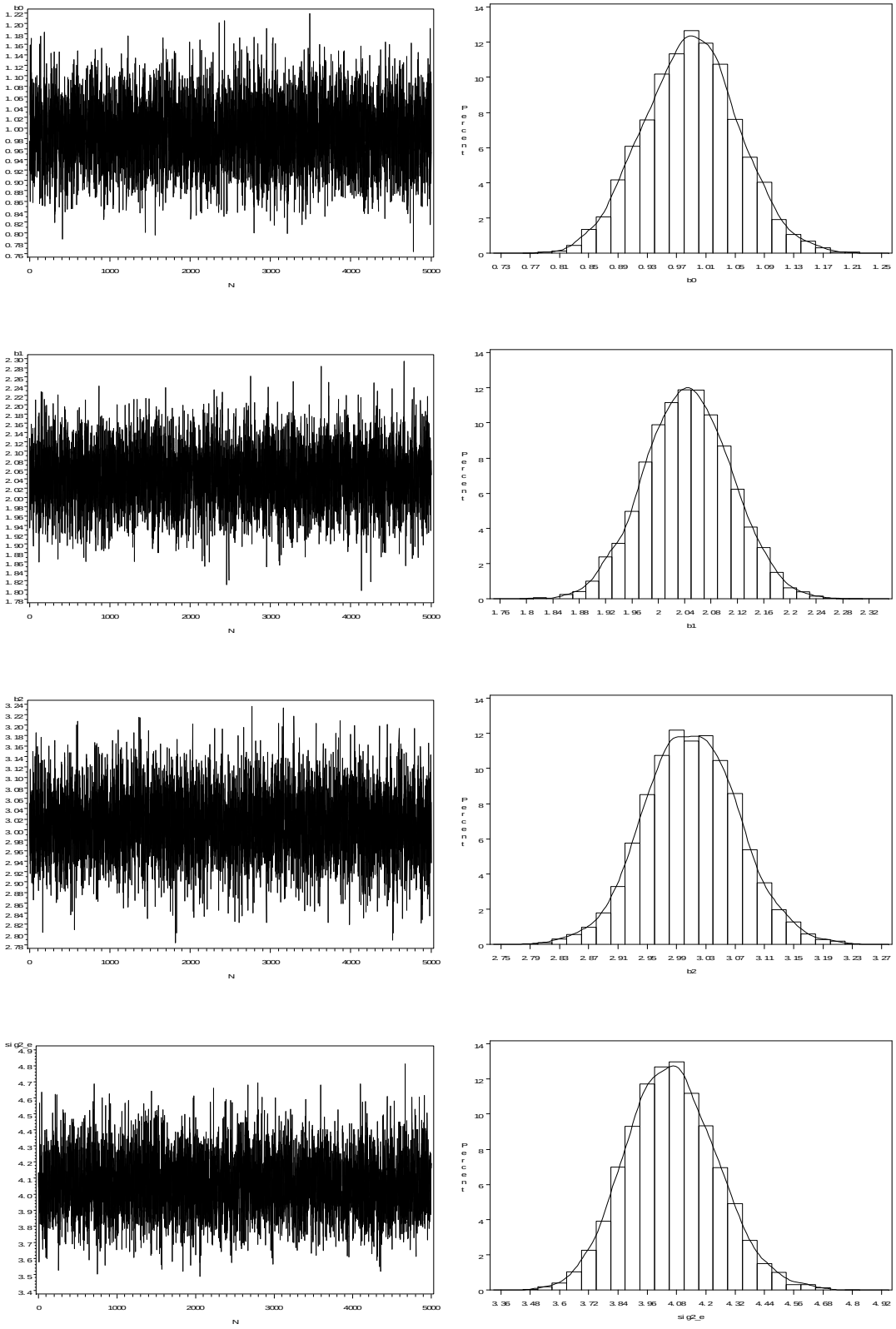

Figure 2. History plots and Histogram plots of parameters from the regression model

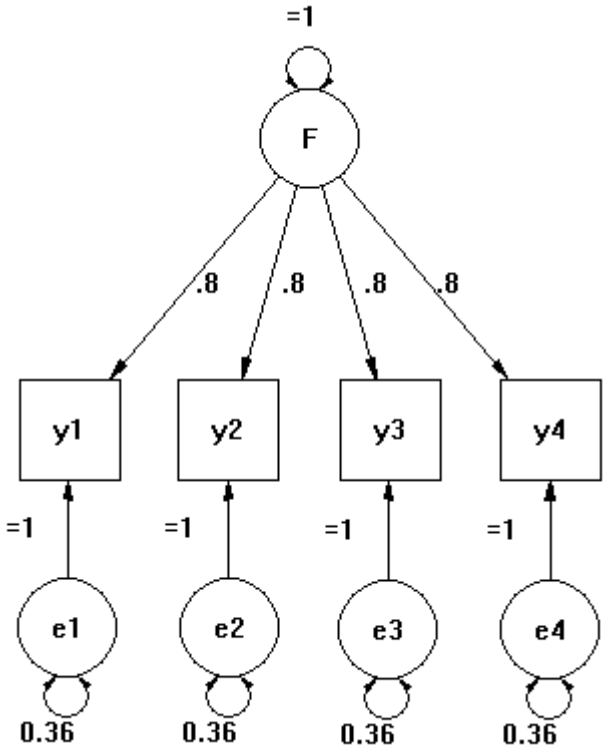

Figure 3. Path diagram for the population confirmatory factor model