Outline

- An introduction to WinBUGS
  - What is WinBUGS
  - Download
  - Regesiter
- How to run the WinBUGS program
- How to write the WinBUGS codes
  - Basic Linear SEM (LSEM.odc)
  - Nonlinear SEM (NSEM.odc)
  - Linear SEM with ordered categorical data (Ordered.odc)
- R2WinBUGS package
  - Simulation (model_sim.txt, Sim.R)
  - Bayes Factor (model_BF.txt, BF.R)

Introduction to WinBUGS

- What is WinBUGS?
  - WinBUGS (**Win**dows version of **B**ayesian inference **U**sing **G**ibbs **S**ampler) was mainly developed using MCMC techniques such as the Gibbs sampler and the Metropolis-Hastings (MH) algorithm.
  - It has been shown that under broad conditions, this software can provide reliable Bayesian statistics such as the estimates of parameters and latent variables in the proposed model.

Download WinBUGS

- The WinBUGS 1.4 version can be downloaded from the website:

  http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml

- The WinBUGS manual that provides brief introductions is also available online.

- When you finish installing WinBUGS, remember download the key file for unrestricted use.

  http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/WinBUGS14_immortality_key.txt

Install WinBUGS

1. Exit all other programs currently running.
2. Copy WinBUGS14.exe to your computer.
3. Go into Explore and double click on WinBUGS14.exe.
4. Follow the instructions in the dialog box.
5. You should have a new directory called WinBUGS14 within Program Files.

Install WinBUGS

6. Inside the WinBUGS14 directory is a program called WinBUGS14.exe.

7. Right-click on the pretty WinBUGS icon, select ‘create shortcut’, then drag this shortcut to the desktop.

8. Double click on WinBUGS14.exe to run WinBUGS14.

Note

- There appears to be a problem with installing WinBUGS and/or various patches in Windows 7 and 10.
- Windows 7 and 10 don't seem to like anyone overwriting files in the "C:\Program Files" directory (regardless of permissions). Hence we recommend that WinBUGS be installed elsewhere, e.g. “D:\WinBUGS".
- For more details:

http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml#install

How to run the WinBUGS Program

1. Input the WinBUGS codes
   - Click **File-New**, input the WinBUGS codes, data, and the initial values in the opened blank window (details will be presented later);
   - Click **File-Save**, save the codes.
2. Check model
   - Highlight the word model at the beginning of the aforementioned written code;
   - Click **Model-Specification…**;
   - Click **check model** in the 'Specification Tool' window.

- Error messages will be shown in the status bar at the bottom left of the WinBUGS program window, and the cursor will be at the place where the error was found.
- If there are no errors, then the message ‘model is syntactically correct’ will appear at the bottom left of the WinBUGS program window.

3. Load data

- Highlight the word list
- Click **load data** in the ‘Specification Tool’ window
- When the data have been successfully loaded, the message ‘data loaded' should appear at the bottom left of the WinBUGS program window.

4. Compile model

- Reset the **number of chains** if multiple chains are to be run;
- Click **compile** in the ‘Specification Tool’ window;
- The message ‘model compiled’ should appear at the bottom left of the WinBUGS program window.

5. Load initial values

- Highlight the word list;
- Click **load inits** in the 'Specification Tool' window;
- Repeat the process to load all the initial values;

- If the model is fully initialized after loading the initial values, then a message saying ‘initial values loaded: model initialized’ will appear at the bottom left of the WinBUGS program window.
- Otherwise, the message ‘the chain contains uninitialized variables’ will appear at the bottom left of the WinBUGS program window.

- Click **gen inits** in the ‘Specification Tool’ window to generate the uninitialized initial values.

6. Input the parameters of interested

- Click **Inference**-**Samples…**;
- Type the name of the parameters to be monitored into the box marked **node** in the ‘Sample Monitor Tool’ window;
- Click **set** in the ‘Sample Monitor Tool window’;
- Repeat the process to input all the parameters

7. Sample observations for the parameters

- Click **Model-Update...**;
- Type the total number of MCMC iterations (for example, 5000 or 10000; the default value is 1000) into the box named **updates** in the 'Update Tool' window;
- Type the number of thin (i.e., the samples from every k-th iteration will be stored) into the box marked **thin** (the default is 1);
- Click **update** and wait for the program to finish simulating the specified number of iterations.

8. Check convergence

- Click **Inference**-**Samples…**;
- Choose the parameters you inputted in Step 6 in the box marked **node** in the 'Sample Monitor Tool' window;
- Click **history**, a graphics window showing the sample trace will appear.

9. Obtained Bayesian statistics

- Specify the burn-in iterations (i.e. the first few iterations, say J, will be discarded), this is carried out by specifying the number J in the box marked **beg** in the 'Sample Monitor Tool' window, the remaining iteration will be used to obtain the Bayesian statistics;
- Click **stats** in the 'Sample Monitor Tool' window, the results for statistics inference, for example, the mean (the Bayesian estimates), standard deviation, Monte Carlo error, will be shown.

How to write WinBUGS codes

- Basic notation
  - Stochastic node: In the text-based model description, stochastic nodes are represented by the node name followed by a twiddles symbol (**~**) followed by the distribution name followed by a comma-separated list of parents enclosed in brackets. For example:

    y[i,j]~dnorm(mu[i,j], psi[j])

    That is to say, y[i,j] is a stochastic node distributed as normal, with which mean is mu[i,j], and **variance is 1/psi[j]**.

- Logical Nodes
  - Logical nodes are represented by the node name followed by a left pointing arrow followed by a logical expression of its parent nodes. For example:

```
mu[i,1]<-u[1]+eta[i,1]
mu[i,2]<-u[2]+lam[1]*eta[i,1]
```

  - **It is important to keep in mind that logical nodes are included only for notational convenience - they cannot be given data or initial values!**

- Array and indexing
  - Arrays are indexed by terms within square brackets.

    The four basic operators +, -, *, and / along with appropriate bracketing are allowed to calculate an integer function as an index, for example: y[i,j], y[i+1, j+1], y[i*j, k].

- Repeated structure
  - Repeated structures are specified using a "for-loop". The syntax for this is:

    for (i in a:b) { list of statements }
  - For example:

```
for (i in 1:N) {
   for (j in 1:P) { y[i,j]~dnorm(mu[i,j], psi[j]) }
}
```

- Formatted data: S-plus format
  - This allows scalars and arrays to be named and given values in a single structure headed by key-word list. There must be **NO SPACE** after list. For example:

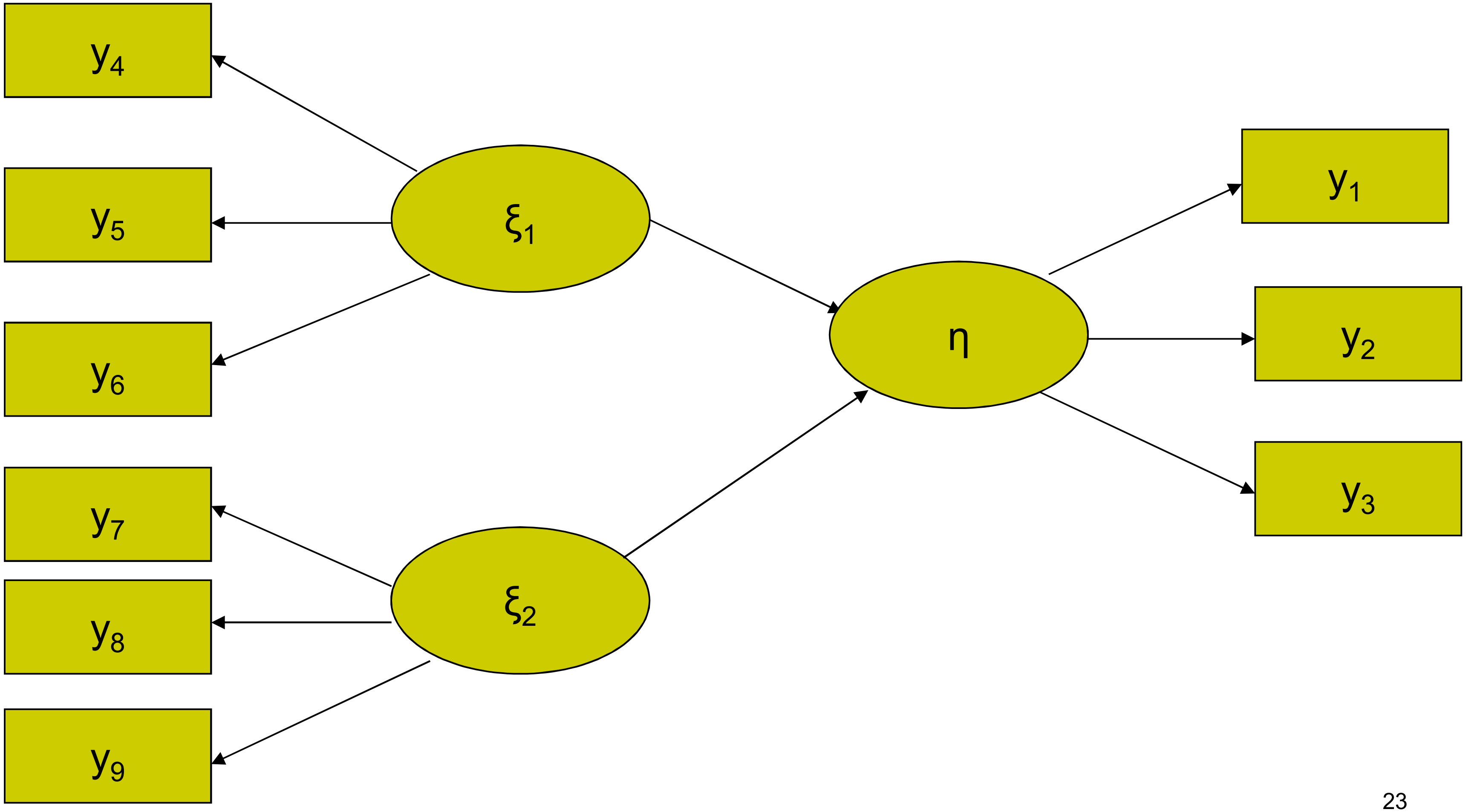
    list(N=300, P=9, ux=c(0,0),

    R=structure(.Data=c(1,0,0,1), .Dim=c(2,2)),

    y=structure(.Data=c(…), .Dim=c(300,9)))

  - **WinBUGS reads data into an array by filling the right-most index first (by row), whereas the S-Plus program fills the left-most index first (by column).**

Linear SEM



- Measurement equation
  - For $i = 1 \ldots, N$, $j = 1, \ldots, 9$; when given $\boldsymbol{\theta}$ and $\boldsymbol{\omega}_i$,the distribution of y_{ij} is $y_{ij} \sim N[\mu_{ij}, \psi_j]$, so in the WinBUGS language, the measurement equation can be written as follows:

```
model {
   for (i in 1:N) {
      #measurement equation
      for (j in 1:P) { y[i,j]~dnorm(mu[i,j], psi[j]) }
      mu[i,1]<-u[1]+eta[i]
      mu[i,2]<-u[2]+lam[1]*eta[i]
      mu[i,3]<-u[3]+lam[2]*eta[i]
      mu[i,4]<-u[4]+xi[i,1]
      ……
      mu[i,9]<-u[9]+lam[6]*xi[i,2]
      #structure equation
      ……
   }
   #prior
   …...
}
```

- Structure equation
  - Similarly, when given $\boldsymbol{\theta}$ and $\boldsymbol{\xi}_i$, the distribution of η_i is $\eta_i \sim N[\nu_i, \psi_\delta]$. Moreover, $\boldsymbol{\xi}_i \sim N[\mathbf{0}, \mathbf{\Phi}]$. Therefore, the structure equation is written as follows:

```
model {
    for (i in 1:N) {
        #measurement equation
        …
        #structure equation
        eta[i]~dnorm(v[i], psd)
        v[i]<-gam[1]*xi[i,1]+gam[2]*xi[i,2]
        xi[i,1:2]~dmnorm(ux[1:2], phi[1:2,1:2])
    }
    #prior
    …
}
```

- Prior inputs

```
model {
  for (i in 1:N) {
     #measurement equation
     …
     #structure equation
     …
  }
  #prior
  for (i in 1:P) { u[j]~dnorm(0,1) }

  lam[1]~dnorm(0.8, psi[2]) lam[2]~dnorm(0.8, psi[3])
  …
  gam[1]~dnorm(0.5, psd) gam[2]~dnorm(0.5, psd)

  for (i in 1:P) { psi[i]~dgamma(9,4)  sgm[i]<-1/psi[i] }

  psd~dgamma(9,4)    sgd<-1/psd

  phi[1:2,1:2]~dwish(R[1:2,1:2], 5)
  phx[1:2,1:2]<-inverse(phi[1:2,1:2])
}
```

- Data
  - In the data list, we will specify N (the sample size), P (the dimension of y), ux (the mean of xi), R (the hyperparameter), and y (the data).

```
model {
    …
}
data
list(N=300, P=9, ux=c(0,0),
    R=structure(.Data=c(1,0,0,1), .Dim=c(2,2)),
    y=structure(.Data=c(…), .Dim=c(300,9)))
```

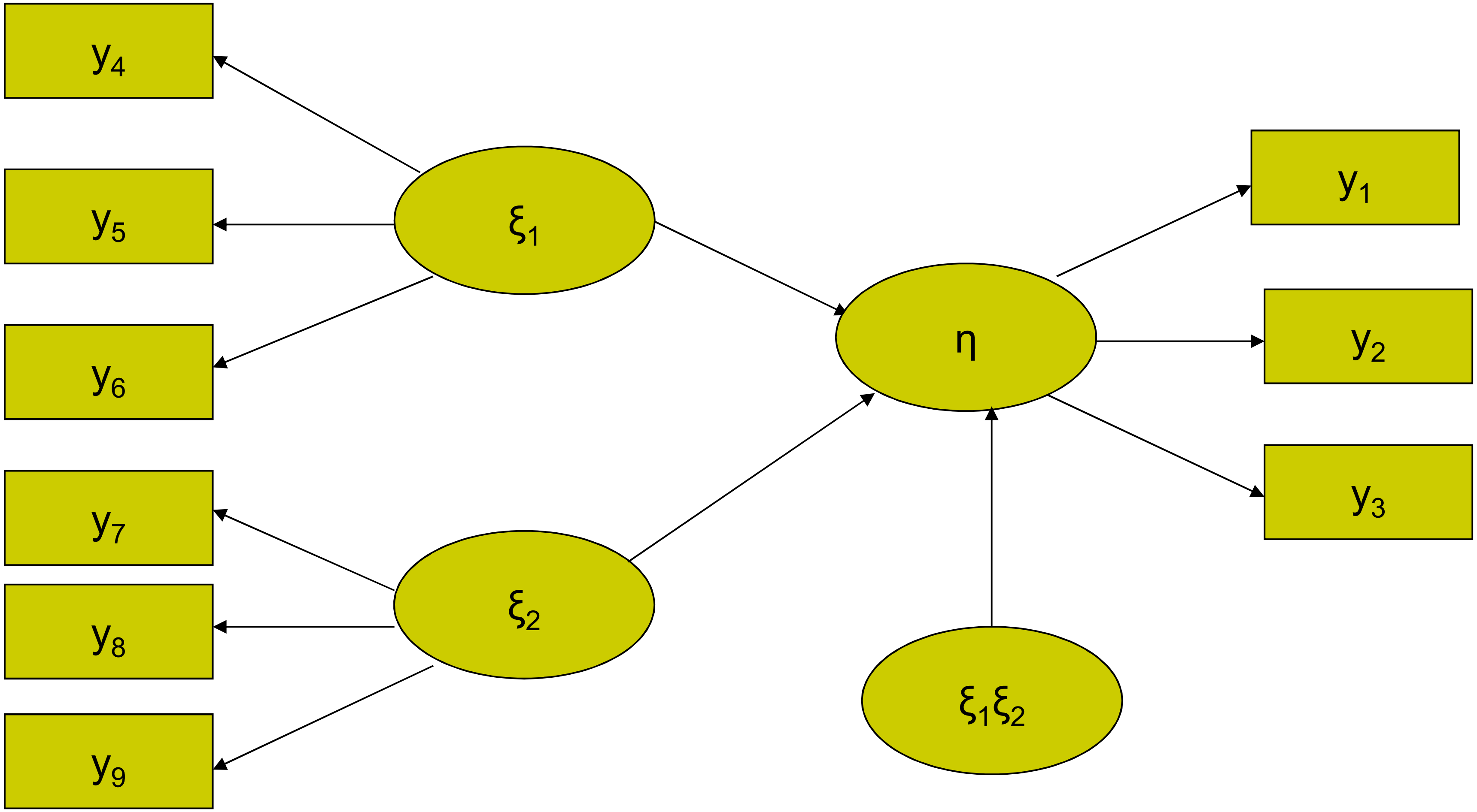
- Initial values
  - In the initial values lists, we will give different initial values of 3 different chains.

```
model {
   …
}
data
list(…)

Initial values
list(u=c(0,0,0,0,0,0,0,0,0), lam=c(1,1,1,1,1,1),
    gam=c(1,1,1), psi=c(1,1,1,1,1,1,1,1,1), psd=1,
    phi=structure(.Data=c(1,0,0,1), .Dim=c(2,2)))

list(…)
list(…)
```

Nonlinear SEM

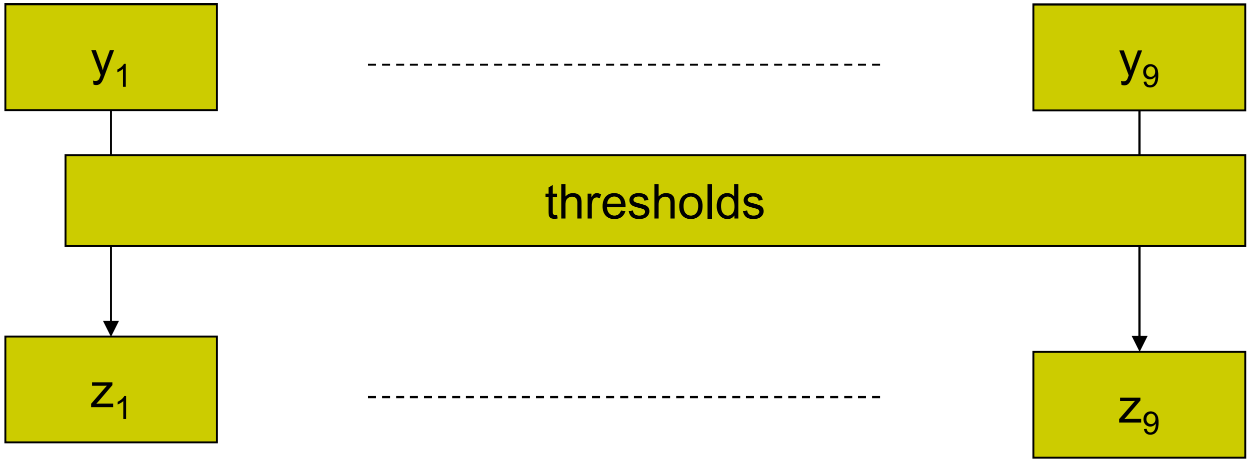


- WinBUGS code
  - The measurement equation is the same as the linear SEM, so we just rewrite the structure equation:

```
model {
    for (i in 1:N) {
        #measurement equation
        …
        #structure equation
        eta[i]~dnorm(v[i], psd)
        v[i]<-gam[1]*xi[i,1]+gam[2]*xi[i,2]+gam[3]*xi[i,1]*xi[i,2]
        xi[i,1:2]~dmnorm(ux[1:2], phi[1:2,1:2])
    }
    #prior
    …
}
```

Linear SEM with Ordered Categorical Data

- Path Diagram



- Model
  - The distribution of y_{ij} is $y_{ij} \sim N[\mu_{ij}, \psi_j] I(y_{ij} \in A_{ij})$, where $A_{ij} = (\alpha_{j,z_{ij}}, \alpha_{j,z_{ij}+1}]$, so in the WinBUGS language, the measurement

    equation can be written as follows:

```
model {
    for (i in 1:N) {
        #measurement equation
        for (j in 1:P) {
            y[i,j]~dnorm(mu[i,j], psi[j])I(thd[j,z[i,j]], thd[j,z[i,j]+1])
        }
        ……
        #structure equation
        ……
    }
    #prior
    ……
}
```

- Data an Initial values
  - In the data list, we will specify N (the sample size), P (the dimension of y), ux, R (the hyperparameter), and z (the data).
  - Furthermore, it is not straightforward to apply WinBUGS to simultaneously estimate the unknown thresholds and structural parameters, so we will fix thd (the thresholds) and put it into the data list.
  - In the initial values lists, it is desirable to give the initial values of xi (the explanatory latent variable) to save computer time.

R2WinBUGS Package

WinBUGS is an interactive program, and it is not convenient to use it to do simulation directly.

However, WinBUGS can be run in batch mode using scripts, and the R package **R2WinBUGS** uses this feature and provides tools to directly call WinBUGS after the manipulation in R.

Furthermore, it is possible to work on the results after importing them back into R.

The implementation of R2WinBUGS is mainly based on the R function ‘bugs(),’ which takes data and initial values as input.

It automatically writes a WinBUGS script, calls the model, and saves the simulation for easy access in R.

The description of the R function “bugs()”

bugs	*Run WinBUGS from R*

Description

The bugs function takes data and starting values as input. It automatically writes a WinBUGS script, calls the model, and saves the simulations for easy access in R.

Usage

```
bugs(data, inits, parameters.to.save, model.file = "model.txt",
    n.chains = 3, n.iter = 2000, n.burnin = floor(n.iter/2),
    n.thin = max(1, floor(n.chains * (n.iter - n.burnin)/1000)),
    bin = (n.iter - n.burnin) / n.thin,
    debug = FALSE, DIC = TRUE, digits = 5, codaPkg = FALSE,
    bugs.directory = "c:/Program Files/WinBUGS14/",
    working.directory = NULL, clearWD = FALSE, useWINE = FALSE,
    WINE = Sys.getenv("WINE"))
```

Details to Run

1. Write the WinBUGS model in an ASCII file.
2. Go into R.
3. Prepare the inputs to the bugs function and run it (see Example).
4. A WinBUGS window will pop up amd R will freeze up. The model will now run inWinBUGS. It might take awhile. You will see things happening in the Log window within WinBUGS. When WinBugs is done, its window will close and R will work again.

More Details

- R2WinBUGS package manual

  http://cran.r-project.org/web/packages/R2WinBUGS/index.html

- Sturtz, S., Ligges, U. and Gelman, A. R2WinBUGS: A package for running WinBUGS from R. *Journal of Statistical Software*, 2005, **12**, 1-16.

  http://cran.r-project.org/web/packages/R2WinBUGS/vignettes/R2WinBUGS.pdf

Calculate Bayes Factor using R2WinBUGS

- Suppose we are going to compared the aforementioned Linear SEM (M_0) and Nonlinear SEM (M_1).

- The Bayes factor is defined as follows:

$$B_{10} = \frac{p(\mathbf{Y}|M_1)}{p(\mathbf{Y}|M_0)}.$$

- The structural equation of the link model is as follows:

$$\eta = \gamma_1\xi_1 + \gamma_2\xi_2 + t\gamma_3\xi_1\xi_2 + \delta.$$

- When t=0, the link model reduces to the Linear SEM (M_0); while t=1, the link model reduces to the Nonlinear SEM (M_1).

- The derivative of the complete data log-likelihood function with respect to t is:

$$U = \sum_{i=1}^{n}[\eta_i - (\gamma_1\xi_{i1} + \gamma_2\xi_{i2} + t\gamma_3\xi_{i1}\xi_{i2})](\gamma_3\xi_{i1}\xi_{i2})/\psi_\delta.$$

- # WinBUGS Code (model.txt)

```
model {
    for (i in 1:N) {
        #measurement equation
        …
        #structure equation
        eta[i]~dnorm(v[i], psd)
        v[i]<-gam[1]*xi[i,1]+gam[2]*xi[i,2]+t*gam[3]*xi[i,1]*xi[i,2]

        uu[i]<-(eta[i]-v[i])*psd*(gam[3]*xi[i,1]*xi[i,2])

        xi[i,1:2]~dmnorm(ux[1:2], phi[1:2,1:2])
    }
    ubar<-sum(uu[])

   #prior
    …
}
```

R2WinBUGS Code (BF.R)

```
library(R2WinBUGS)          #Load R2WinBUGS package
…
data=list(N=300, P=9, ux=c(0,0), R=matrix(c(1, 0, 0, 1), nrow=2),
    y=Y, t=NA)   #Data
init1=list(u=rep(0,9), lam=rep(0,6), psi=rep(1,9), psd=1,
        gam=rep(0,3), phi=matrix(c(1, 0, 0, 1), nrow=2))
inits=list(init1)
parameters=c("ubar")

…
for (i in 1:21) {          #Path sampling
   data$t<-(i-1)*0.05
   model<-bugs(data, inits, parameters,
            model.file="D:/Run/model.txt",
            n.chains=1, n.iter=1500, n.burnin=1000, n.thin=1,
            bugs.directory="C:/Program Files/WinBUGS14/",
            working.directory="D:/Run/")
   u[i]<-model$mean$ubar
}
```

Reference to WinBUGS

- Books
  - Song, X. Y. and Lee, S. Y. *Basic and Advanced Bayesian Structural Equation Modeling: With Applications in the Medical and Behavioral Sciences,* John Wiley and Sons, 2012.
  - Lee, S. Y. *Structural Equation Modeling: A Bayesian Approach*, John Wiley and Sons, 2007.
  - Congdon, P. *Bayesian Statistical Modelling*, John Wiley and Sons, 2006.

Reference to WinBUGS

- WinBUGS User Manual
- WinBUGS homepage
- Search Engines (Google, Bing, …)