STAT 5020

Chapter 3 Bayesian Methods for Estimating Structural Equation Models Supplementary

Department of Statistics

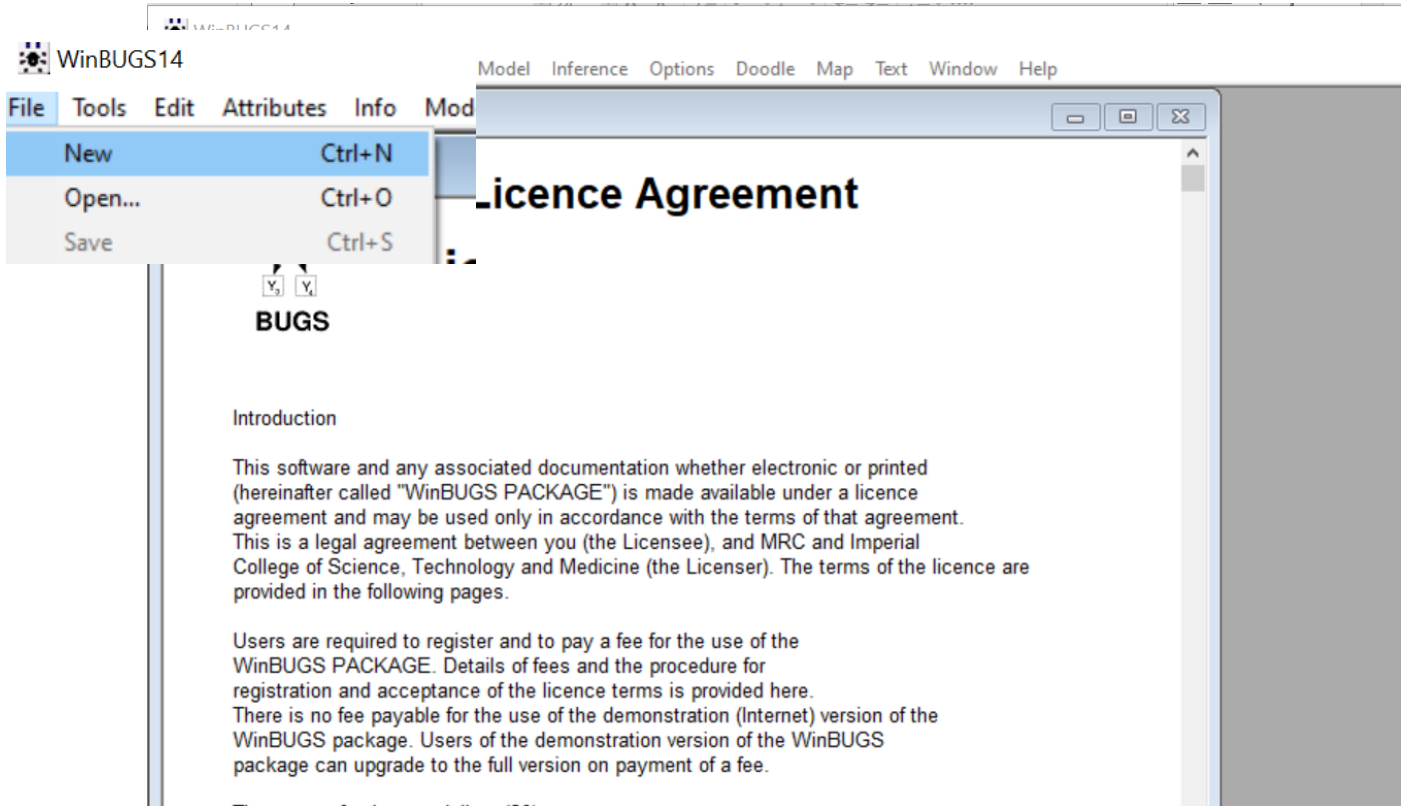
2021/2022 Term 2

Section 1: Estimating with “WinBUGS”

WinBUGS

WinBUGS (Windows version of Bayesian inference Using Gibbs Sampler) was mainly developed using MCMC techniques such as the Gibbs sampler and the Metropolis-Hastings (MH) algorithm.

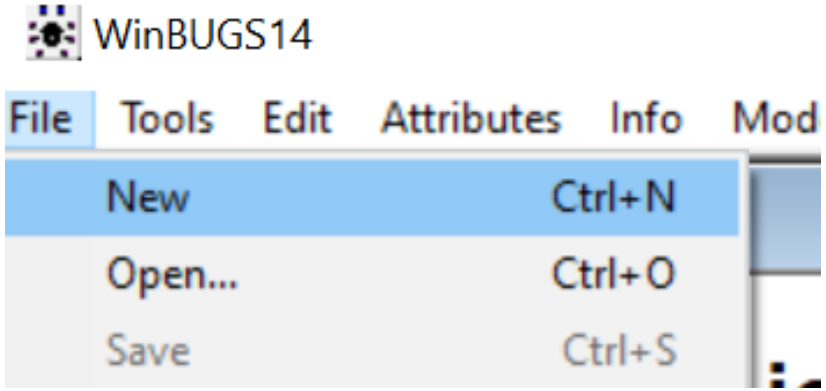
It has been shown that under broad conditions, this software can provide reliable Bayesian statistics such as the estimates of parameters and latent variables in the proposed model.



WinBUGS

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.



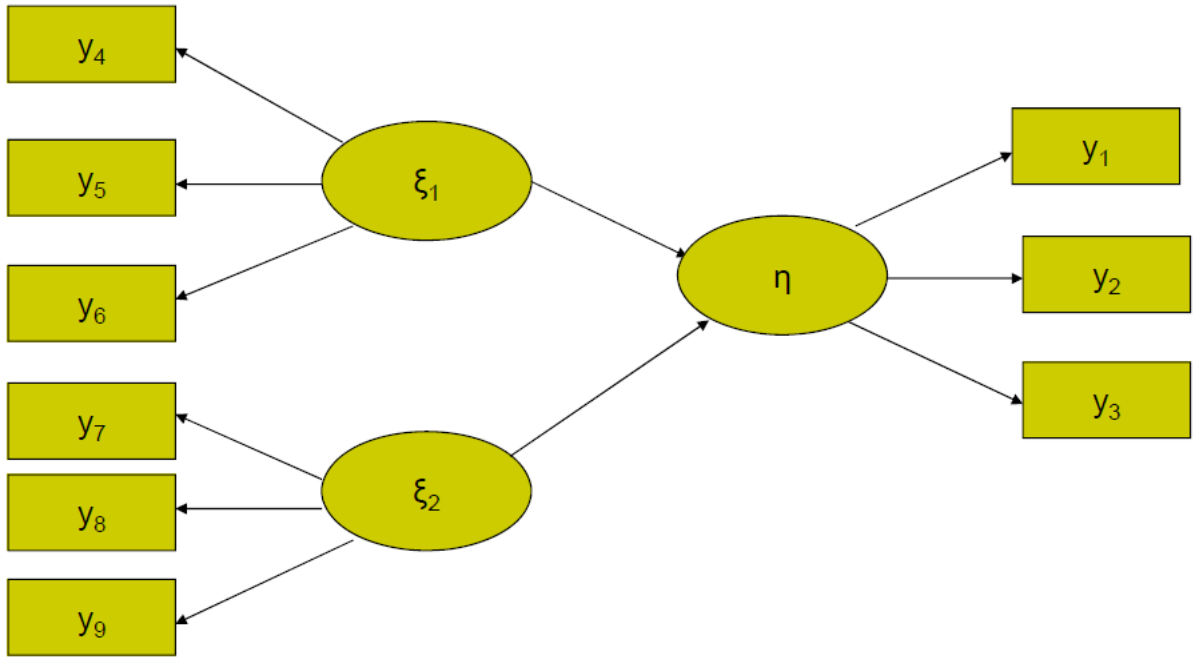
untitled1

```
model {
for (i in 1:N) {
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,5]<-u[5]+lam[3]*xi[i,1]
mu[i,6]<-u[6]+lam[4]*xi[i,1]
mu[i,7]<-u[7]+xi[i,2]
mu[i,8]<-u[8]+lam[5]*xi[i,2]
mu[i,9]<-u[9]+lam[6]*xi[i,2]
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])
}
for (i in 1:P) { u[i]~dnorm(0,1) }
lam[1]~dnorm(0.8, psi[2]) lam[2]~dnorm(0.8, psi[3])
lam[3]~dnorm(0.8, psi[5]) lam[4]~dnorm(0.8, psi[6])
lam[5]~dnorm(0.8, psi[7]) lam[6]~dnorm(0.8, psi[8])
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
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(5.96967721750
,5.17985321872314,5.41505676327374,5.6468405034362,4.3460000155
4896,3.88005053369144,3.99525474326025,5.56676854133538,5.63412
806948818,3.78996404145415,4.90707758353927,3.52159092237011,5.
38788066822738,4.48937452962244,4.19719557523545,5.276300724842
25,6.13685858582264,3.3091718932606,4.49653239101561,6.92173533
432537,5.56604470701561,7.61880935400792,3.60081922046454,5.154
```

WinBUGS MODEL PART

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega} + \boldsymbol{\epsilon}$$

$$\boldsymbol{\eta} = \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\delta}$$



untitled1

```
model {
for (i in 1:N) {
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,5]<-u[5]+lam[3]*xi[i,1]
mu[i,6]<-u[6]+lam[4]*xi[i,1]
mu[i,7]<-u[7]+xi[i,2]
mu[i,8]<-u[8]+lam[5]*xi[i,2]
mu[i,9]<-u[9]+lam[6]*xi[i,2]
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])
}
for (i in 1:P) { u[i]~dnorm(0,1) }
lam[1]~dnorm(0.8, psi[2]) lam[2]~dnorm(0.8, psi[3])
lam[3]~dnorm(0.8, psi[5]) lam[4]~dnorm(0.8, psi[6])
lam[5]~dnorm(0.8, psi[7]) lam[6]~dnorm(0.8, psi[8])
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])
}
```

$N = 300$
$P = 9$

```
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(5.96967721750
,5.17985321872314,5.41505676327374,5.6468405034362,4.3460000155
4896,3.88005053369144,3.99525474326025,5.56676854133538,5.63412
806948818,3.78996404145415,4.90707758353927,3.52159092237011,5.
38788066822738,4.48937452962244,4.19719557523545,5.276300724842
25,6.13685858582264,3.3091718932606,4.49653239101561,6.92173533
432537,5.56604470701561,7.61880935400792,3.60081922046454,5.154
```

WinBUGS MODEL PART

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega} + \boldsymbol{\epsilon}$$

$$\boldsymbol{\eta} = \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\delta}$$

Stochastic node: represented by the node name followed by **a twiddles symbol (~)** and the **distribution name**

Logical node: represented by the node name followed by a **left pointing arrow** and a **logical expression**

untitled1

```
model {
for (i in 1:N) {
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,5]<-u[5]+lam[3]*xi[i,1]
mu[i,6]<-u[6]+lam[4]*xi[i,1]
mu[i,7]<-u[7]+xi[i,2]
mu[i,8]<-u[8]+lam[5]*xi[i,2]
mu[i,9]<-u[9]+lam[6]*xi[i,2]
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])
}
for (i in 1:P) { u[i]~dnorm(0,1) }
lam[1]~dnorm(0.8, psi[2]) lam[2]~dnorm(0.8, psi[3])
lam[3]~dnorm(0.8, psi[5]) lam[4]~dnorm(0.8, psi[6])
lam[5]~dnorm(0.8, psi[7]) lam[6]~dnorm(0.8, psi[8])
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])
}
```

$$N = 300$$
$$P = 9$$

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

Logical nodes are included only for notational convenience - they cannot b given data or initial values

```
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(5.96967721750
,5.17985321872314,5.41505676327374,5.6468405034362,4.3460000155
4896,3.88005053369144,3.99525474326025,5.56676854133538,5.63412
806948818,3.78996404145415,4.90707758353927,3.52159092237011,5.
38788066822738,4.48937452962244,4.19719557523545,5.276300724842
25,6.13685858582264,3.3091718932606,4.49653239101561,6.92173533
432537,5.56604470701561,7.61880935400792,3.60081922046454,5.154
```

WinBUGS

MODEL PART

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega} + \boldsymbol{\epsilon}$$

$$\boldsymbol{\eta} = \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\delta}$$

untitled1

```
model {
for (i in 1:N) {
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,5]<-u[5]+lam[3]*xi[i,1]
mu[i,6]<-u[6]+lam[4]*xi[i,1]
mu[i,7]<-u[7]+xi[i,2]
mu[i,8]<-u[8]+lam[5]*xi[i,2]
mu[i,9]<-u[9]+lam[6]*xi[i,2]
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])
}
for (i in 1:P) { u[i]~dnorm(0,1) }
lam[1]~dnorm(0.8, psi[2]) lam[2]~dnorm(0.8, psi[3])
lam[3]~dnorm(0.8, psi[5]) lam[4]~dnorm(0.8, psi[6])
lam[5]~dnorm(0.8, psi[7]) lam[6]~dnorm(0.8, psi[8])
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])
}
```

$$N = 300$$
$$P = 9$$

Prior inputs

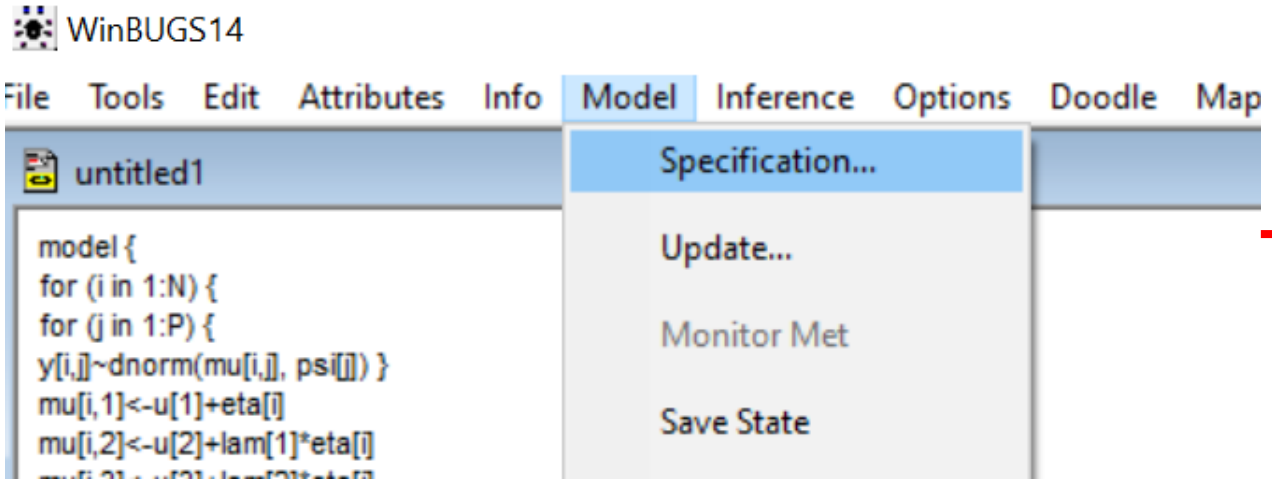
$$[\boldsymbol{\Lambda}_k|\psi_{\epsilon k}] \overset{D}{=} N[\boldsymbol{\Lambda}_{0k}, \psi_{\epsilon k}\mathbf{H}_{0yk}]$$

$$\psi_{\epsilon k} \overset{D}{=} IG[\alpha_{0\epsilon k}, \beta_{0\epsilon k}] \text{ or } \psi_{\epsilon k}^{-1} \overset{D}{=} Gamma[\alpha_{0\epsilon k}, \beta_{0\epsilon k}]$$

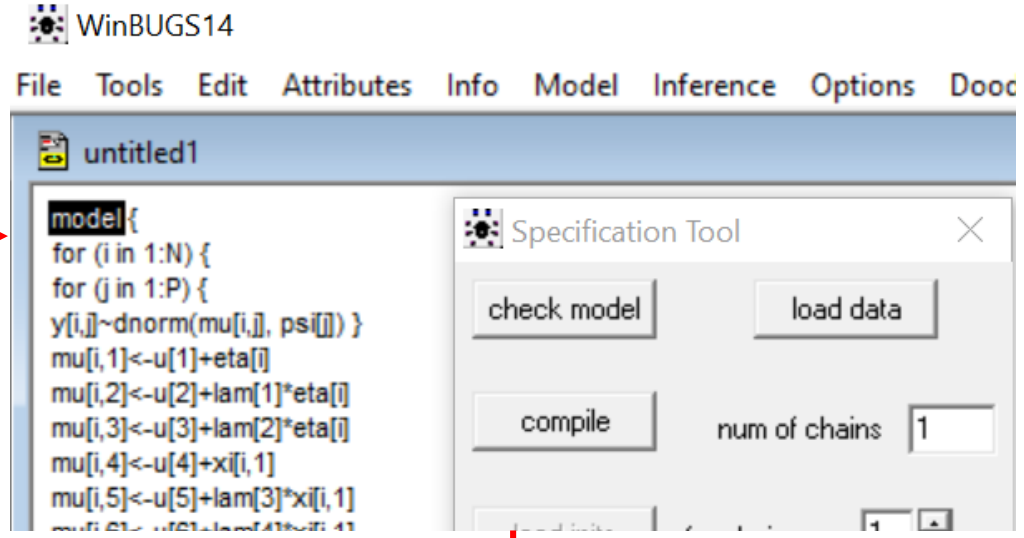
$$IW_q[\mathbf{R}_0^{-1}, \rho_0]$$

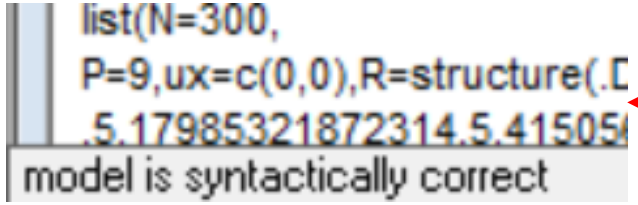
```
list(N=300,
P=9,ux=c(0,0),R=structure(.Data=c(1,0,0,1),.Dim=c(2,2)),y=structure(.Data=c(5.96967721750
,5.17985321872314,5.41505676327374,5.6468405034362,4.3460000155
4896,3.88005053369144,3.99525474326025,5.56676854133538,5.63412
806948818,3.78996404145415,4.90707758353927,3.52159092237011,5.
38788066822738,4.48937452962244,4.19719557523545,5.276300724842
25,6.13685858582264,3.3091718932606,4.49653239101561,6.92173533
432537,5.56604470701561,7.61880935400792,3.60081922046454,5.154
```

WinBUGS MODEL PART



Highlight the word model at the beginning of the aforementioned written code





WinBUGS DATA PART

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

WinBUGS reads data into an array by filling the right-most index first (by row)

```
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])
}
for (i in 1:P) { u[i]~dnorm(0,1) }
lam[1]~dnorm(0.8, psi[2]) lam[2]~dnorm(0.8, psi[3])
lam[3]~dnorm(0.8, psi[5]) lam[4]~dnorm(0.8, psi[6])
lam[5]~dnorm(0.8, psi[7]) lam[6]~dnorm(0.8, psi[8])
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
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(5.96967721750
,5.17985321872314,5.41505676327374,5.6468405034362,4.3460000155
4896,3.88005053369144,3.99525474326025,5.56676854133538,5.63412
806948818,3.78996404145415,4.90707758353927,3.52159092237011,5.
38788066822738,4.48937452962244,4.19719557523545,5.276300724842
25,6.13685858582264,3.3091718932606,4.49653239101561,6.92173533
432537,5.56604470701561,7.61880935400792,3.60081922046454,5.154
03707426772,4.08656416596148,6.19100497193514,6.33709739029276,
5.46043329594432,6.67752759544385,5.96691197042377,3.5573520495
0199,4.73584841530895,6.02351919565167,2.69845802126644,5.32220
424077617,4.56204592933183,4.57678755183993,3.89120737737277,5.
36901919033196,5.27211431748853,5.58990002107531,5.371987633875
55,5.44597449841747,5.52684843962461,4.35193244195084,6.0218370
6837274,6.09423028568049,5.07730598251121,4.72536179159798,5.07
527144425467,5.97051912976969,4.47867647681507,5.83333130836204
```

WinBUGS DATA PART

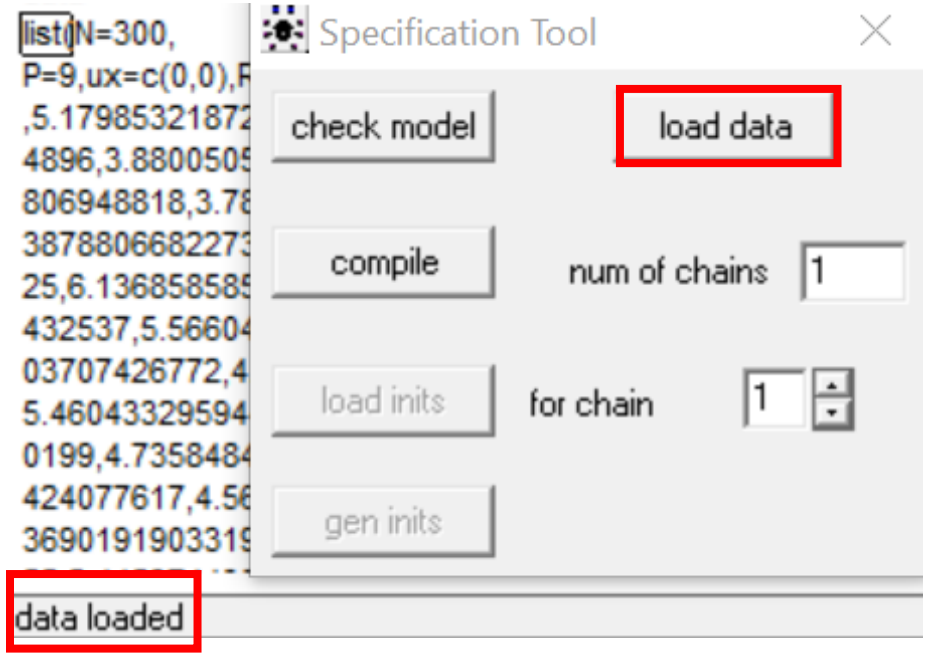
```
##################winbugs generater data#################
library(mvtnorm)   #Load mvtnorm package

N=300                          #Sample size
XI=matrix(NA, nrow=N, ncol=2) #Explanatory latent variables
Eta=numeric(N)                 #Outcome latent variables
Y=matrix(NA, nrow=N, ncol=9) #Observed variables

#The covariance matrix of xi
phi=matrix(c(1, 0.3, 0.3, 1), nrow=2)

eps=numeric(10)

  #Generate Data
  for (i in 1:N) {
    XI[i,]=rmvnorm(1, c(0,0), phi)
    delta=rnorm(1, 0, sqrt(0.5))
    Eta[i]=0.6*XI[i,1]+0.4*XI[i,2]+delta

    eps[1:3]=rnorm(3, -1, sqrt(0.4))
    eps[4:6]=rnorm(3, 0, sqrt(0.4))
    eps[7:9]=rnorm(3, 1, sqrt(0.4))

    Y[i,1]=Eta[i]+eps[1]
    Y[i,2]=0.7*Eta[i]+eps[2]
    Y[i,3]=0.5*Eta[i]+eps[3]
    Y[i,4]=XI[i,1]+eps[4]
    Y[i,5]=0.8*XI[i,1]+eps[5]
    Y[i,6]=0.8*XI[i,1]+eps[6]
    Y[i,7]=XI[i,2]+eps[7]
    Y[i,8]=0.8*XI[i,2]+eps[8]
    Y[i,9]=-0.8*XI[i,2]+eps[9]
  }

options(scipen = 200)
write(t(Y),file="Y.txt",sep=",",ncol=N*9)
```

WinBUGS DATA PART

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).

WinBUGS reads data into an array by filling the right-most index first (by row)

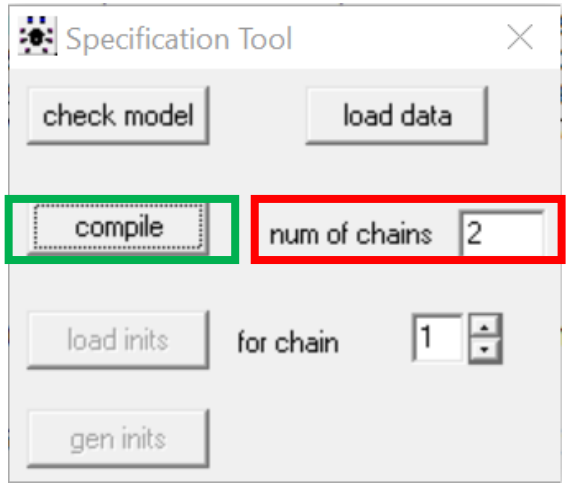
```
350504056,6.0134914277516,5.05319123762966,5.99317720827189,3.3
4220924925449,5.14198470026678,4.32682612147848,3.5581044491931
3,5.83212429121007,5.25660496183973,4.3134298824117,4.984643389
66855,4.14465835412876,3.12781835559333,5.74105903346033,4.6538
5026909991,3.60686724752769,4.21862976243576,5.61698517133772,4.7243923
),.Dim=c(300,9)))
}
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),
```

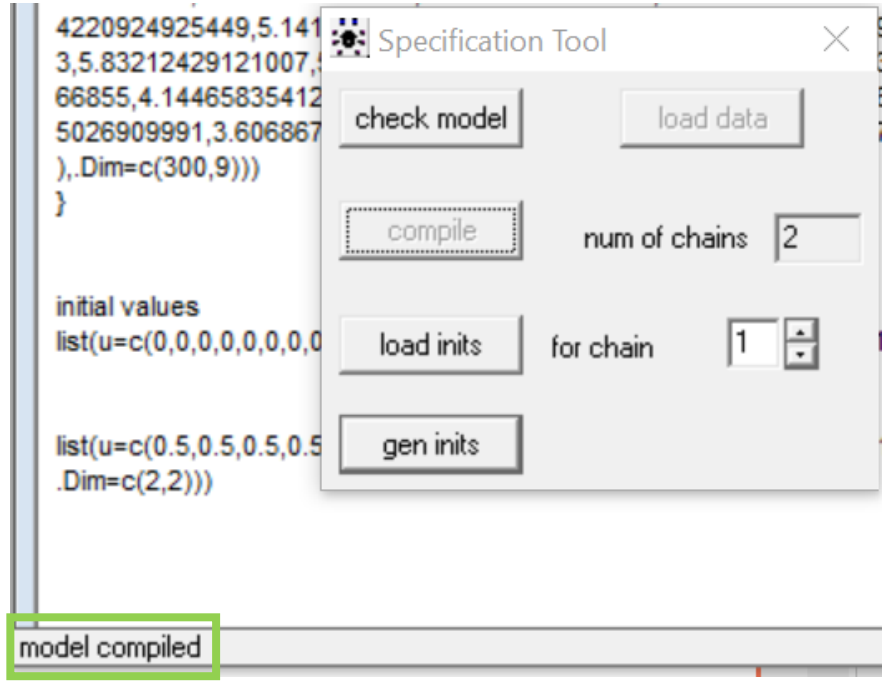


Highligh
the wor
list

WinBUGS Compile model

Reset the number of chains if multiple chains are to be run;

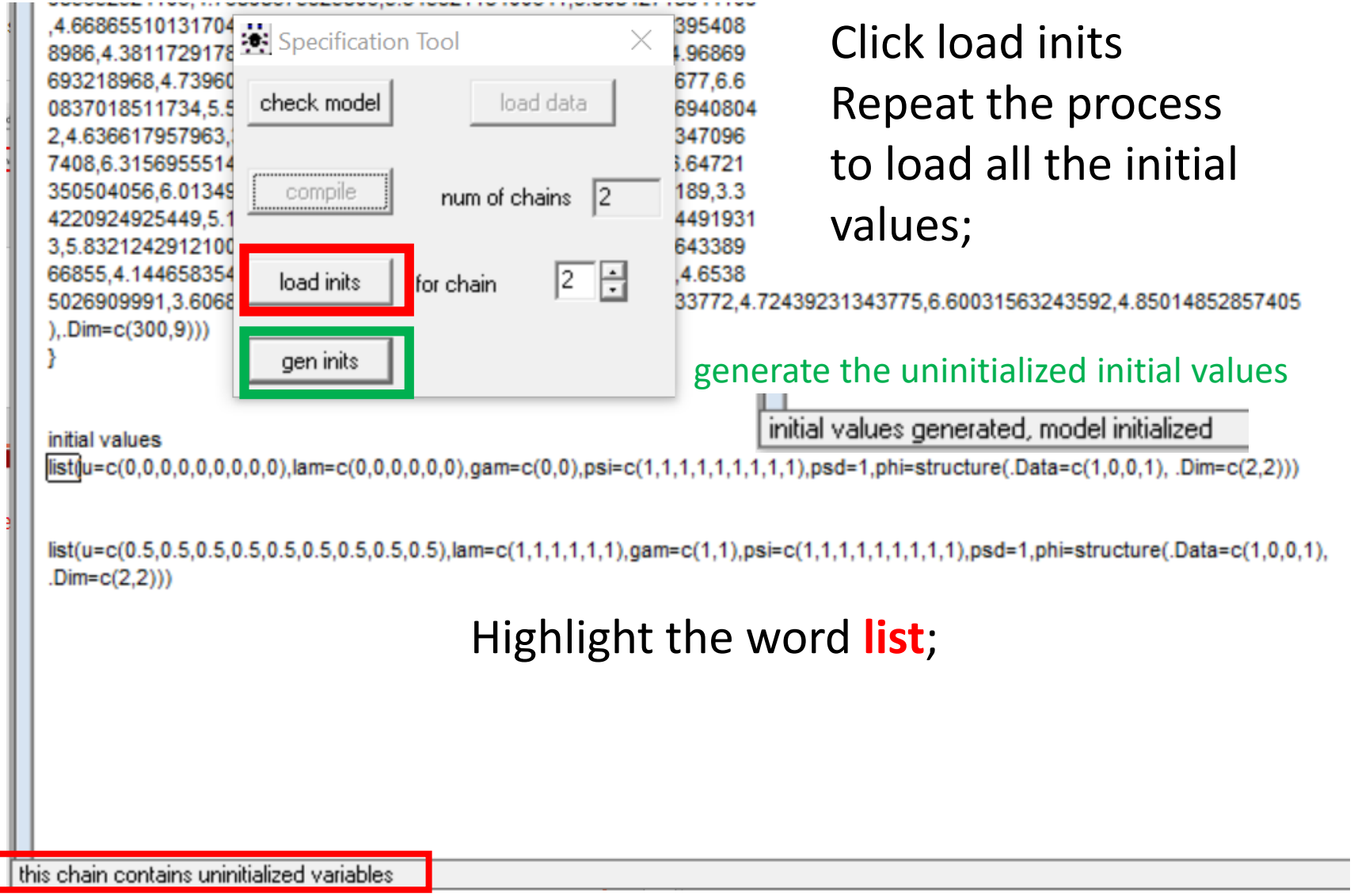




WinBUGS

Compile model

If the model is fully initialized after loading the initial values, then a message saying ‘initial values loaded: model initialized’ will appear.
Otherwise, the message ‘the chain contains uninitialized variables’ will appear.



Click load inits
Repeat the process to load all the initial values;

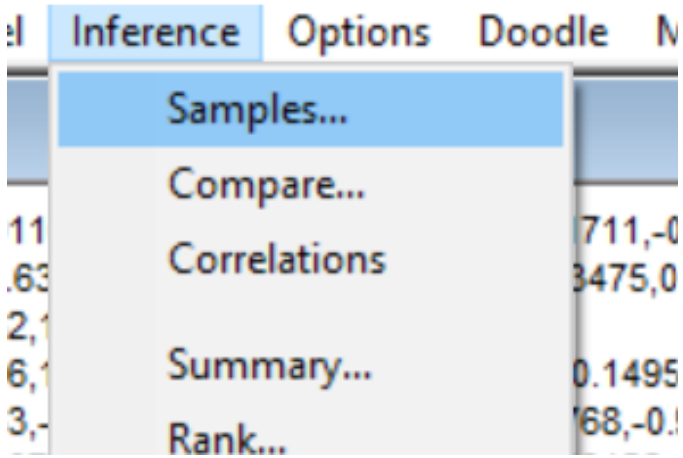
generate the uninitialized initial values

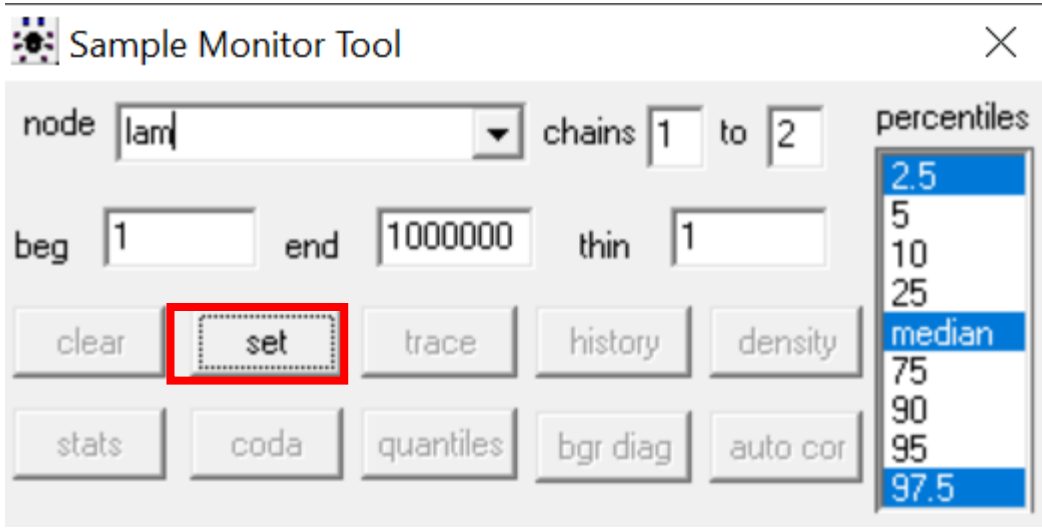
Highlight the word **list**;

WinBUGS

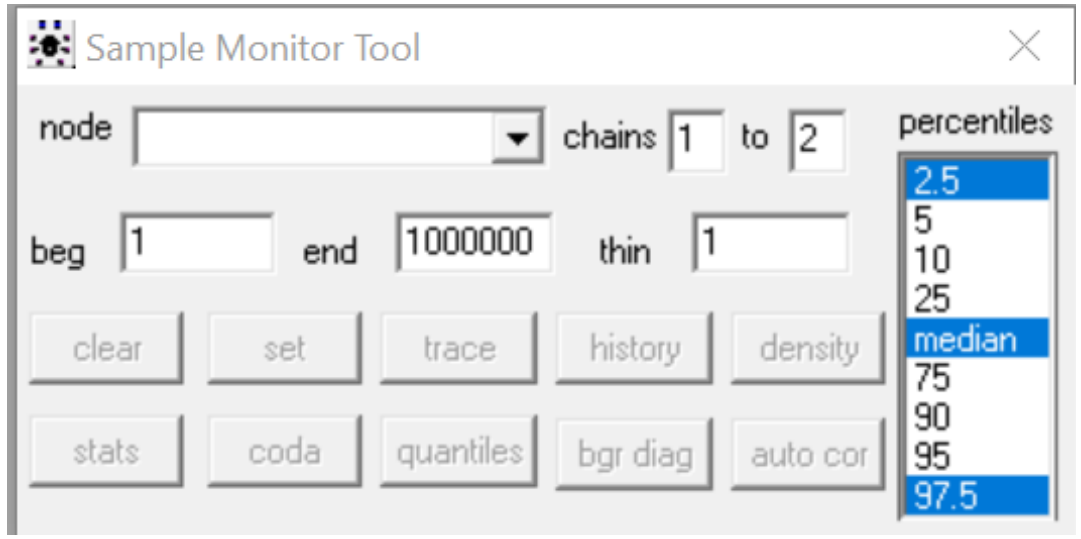
Input the parameters

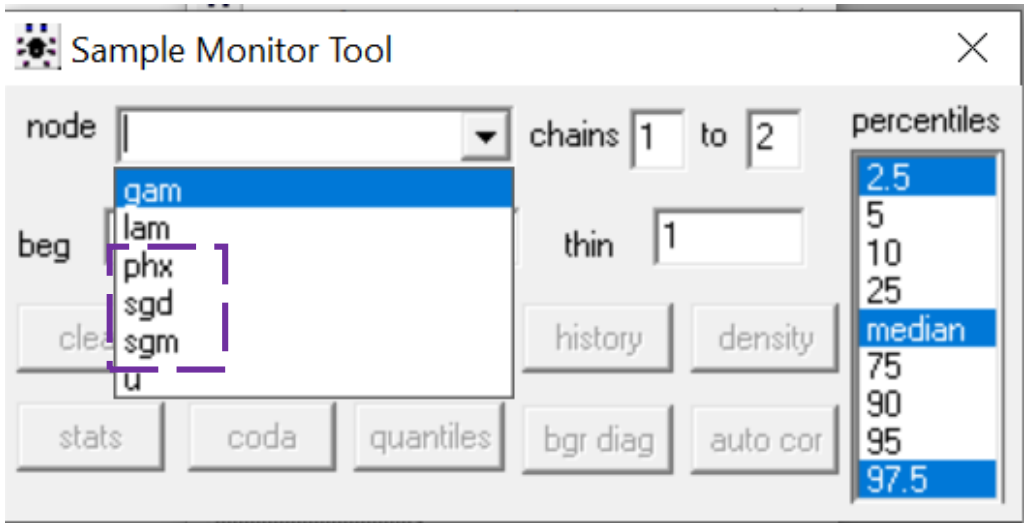
Click Inference-Samples





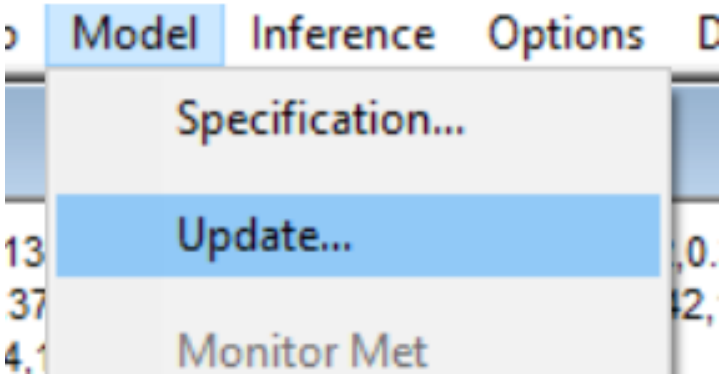
Type the name of the parameters to be monitore
into the box marked node and click set

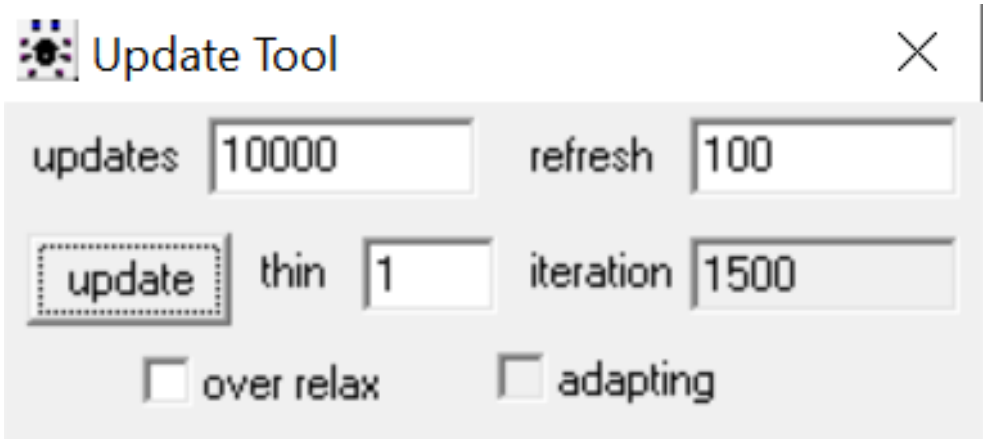




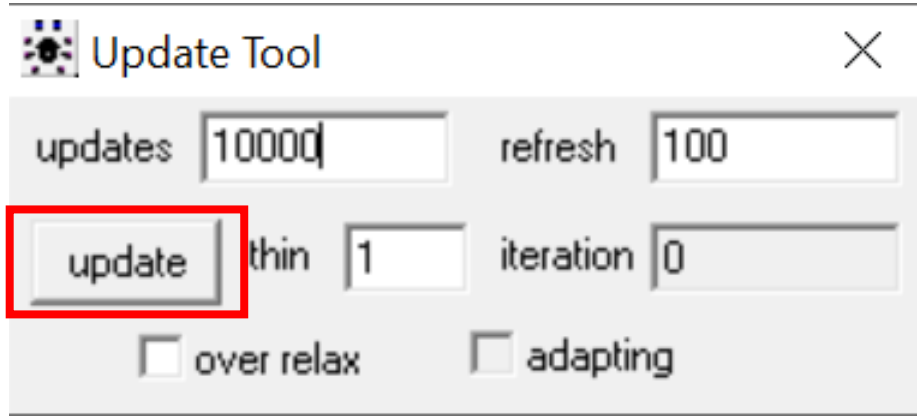
WinBUGS Estimate parameters

Click Model-Update





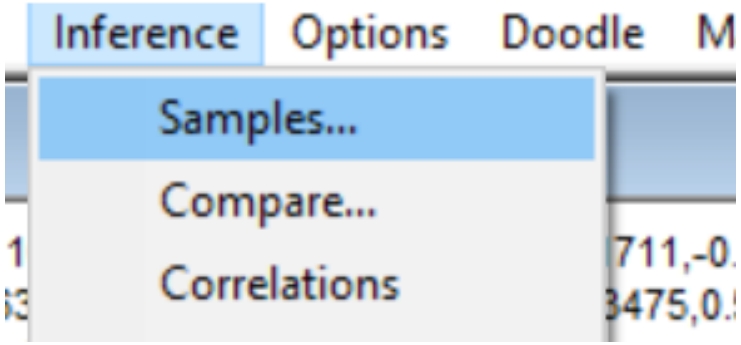
Type the total number of MCMC iterations

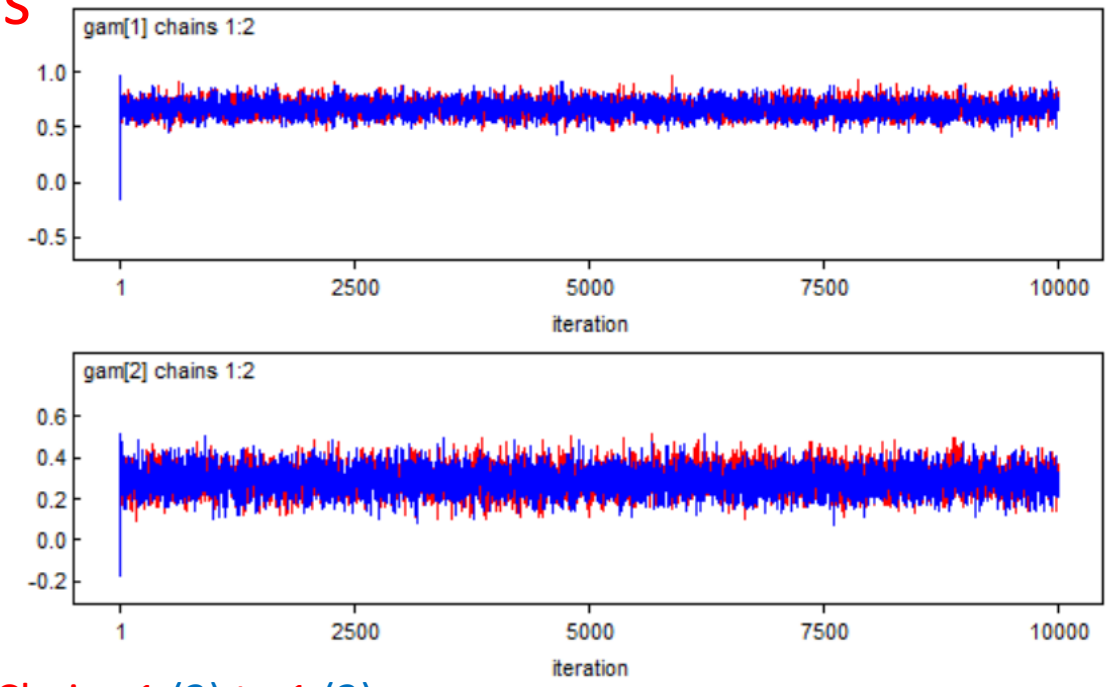


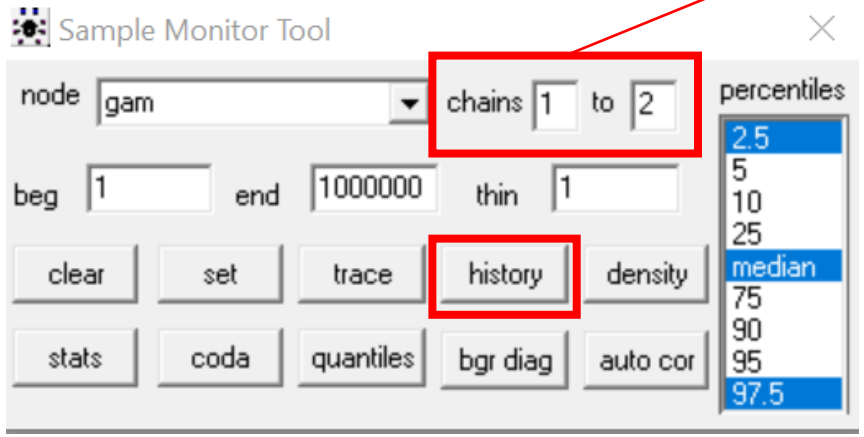
Type the number of thin (i.e., the samples from every k-th iteration will be stored) into the box marked thin (the default is 1);

WinBUGS

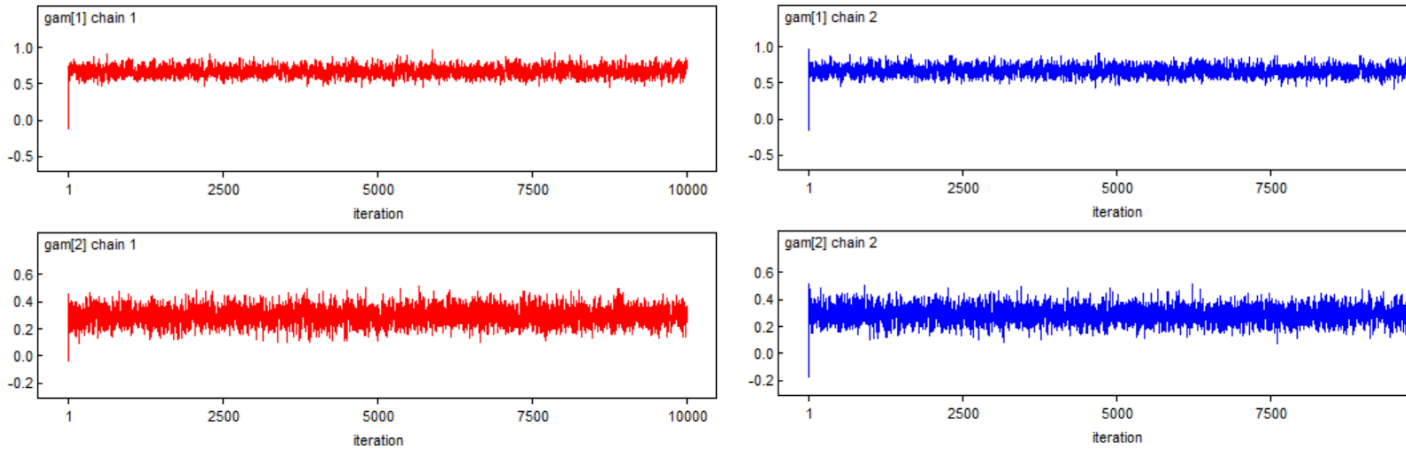
Obtained Bayesian statistics







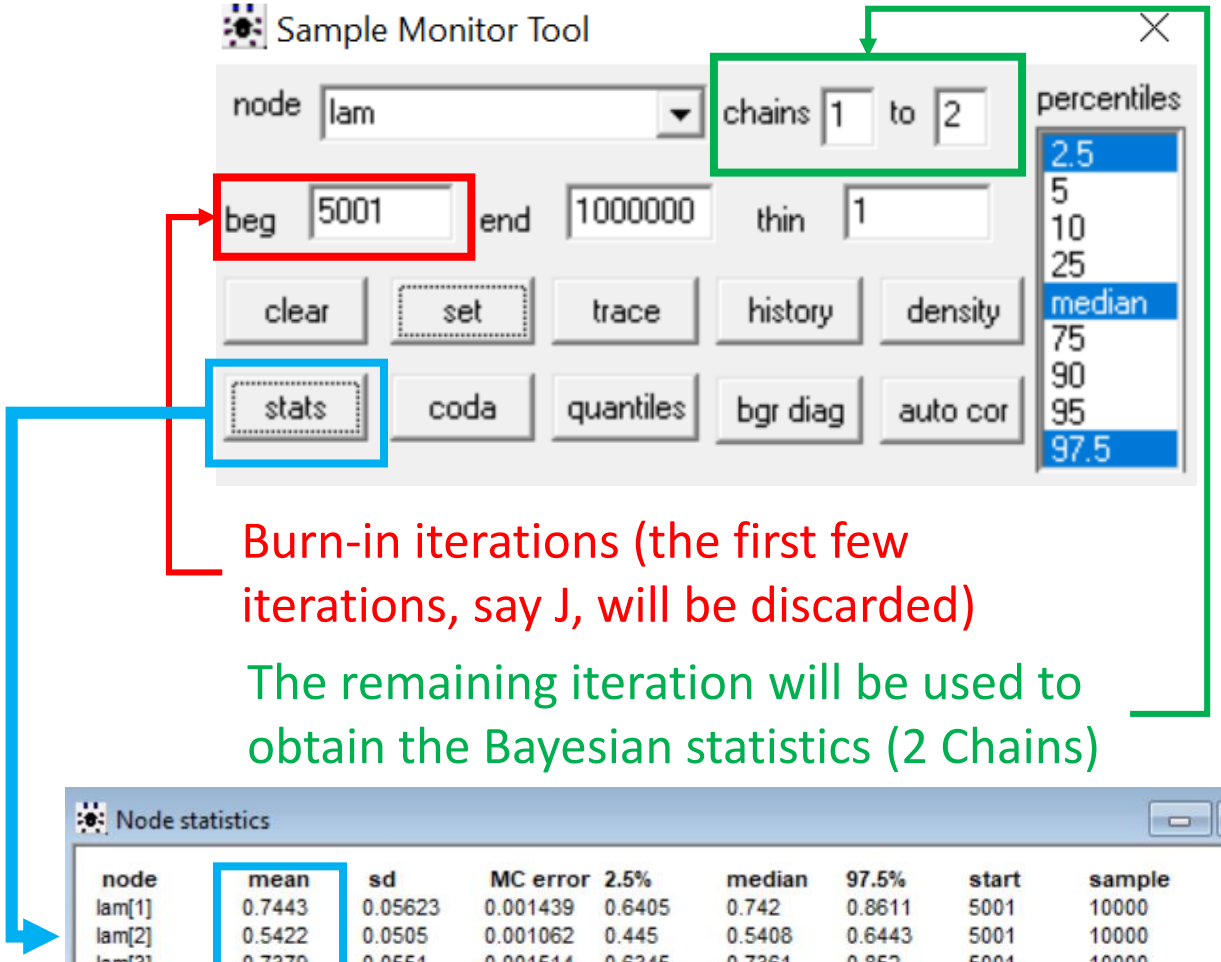
Chains 1 (2) to 1 (2)



a graphics window showing
the sample trace will appear

WinBUGS

Obtained Bayesian statistics



Burn-in iterations (the first few iterations, say J, will be discarded)

The remaining iteration will be used to obtain the Bayesian statistics (2 Chains)

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
lam[1]	0.7443	0.05623	0.001439	0.6405	0.742	0.8611	5001	10000
lam[2]	0.5422	0.0505	0.001062	0.445	0.5408	0.6443	5001	10000
lam[3]	0.7379	0.0551	0.001514	0.6345	0.7361	0.852	5001	10000
lam[4]	0.7318	0.05783	0.00158	0.6224	0.7301	0.8481	5001	10000
lam[5]	0.7544	0.0568	0.001501	0.6497	0.7522	0.8705	5001	10000
lam[6]	-0.7924	0.05795	0.001638	-0.9124	-0.7905	-0.6838	5001	10000

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%
gam[1]	0.561	0.06832	0.001719	0.4324	0.5596	0.702
gam[2]	0.3713	0.06058	0.001426	0.2552	0.3704	0.491

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%
u[1]	-0.9931	0.06593	0.002266	-1.124	-0.9923	-0.86
u[2]	-0.9683	0.05436	0.001708	-1.076	-0.9674	-0.862
u[3]	-0.9588	0.04852	0.001311	-1.055	-0.9584	-0.864
u[4]	-0.02439	0.066	0.002054	-0.1553	-0.02441	0.107
u[5]	0.03716	0.05375	0.001614	-0.07013	0.03749	0.142
u[6]	0.08038	0.05578	0.001604	-0.02972	0.08039	0.190
u[7]	1.029	0.06766	0.002272	0.8976	1.028	1.164
u[8]	1.044	0.05698	0.001712	0.9354	1.043	1.158
u[9]	0.9743	0.05894	0.001833	0.8553	0.975	1.089

WinBUGS

Obtained Bayesian statistics

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
phx[1,1]	0.9367	0.1145	0.002893	0.7312	0.9312	1.179	5001	10000
phx[1,2]	0.2663	0.07061	0.001045	0.1347	0.264	0.4119	5001	10000
phx[2,1]	0.2663	0.07061	0.001045	0.1347	0.264	0.4119	5001	10000
phx[2,2]	1.01	0.1242	0.003395	0.7858	1.005	1.268	5001	10000

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
sgm[1]	0.409	0.05775	0.001435	0.3033	0.4064	0.53	5001	10000
sgm[2]	0.3942	0.04331	8.045E-4	0.3147	0.3922	0.4855	5001	10000
sgm[3]	0.4309	0.03911	5.449E-4	0.3597	0.4294	0.5139	5001	10000
sgm[4]	0.4046	0.05442	0.001459	0.3053	0.4025	0.5174	5001	10000
sgm[5]	0.3662	0.03931	7.656E-4	0.2935	0.3647	0.4485	5001	10000
sgm[6]	0.4309	0.04416	8.281E-4	0.3522	0.4289	0.5253	5001	10000
sgm[7]	0.4117	0.0586	0.00168	0.3037	0.409	0.5361	5001	10000
sgm[8]	0.4243	0.0443	7.648E-4	0.3434	0.4215	0.5181	5001	10000
sgm[9]	0.4308	0.0473	9.095E-4	0.3423	0.4286	0.5284	5001	10000

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
sgd	0.4521	0.06512	0.001716	0.3347	0.449	0.5901	5001	10000

WinBUGS

R2WinBUGS Package

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

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.

```
library(R2WinBUGS) #Load R2WinBUGS package

N=500                              #Sample size
BZ=numeric(N)                      #Fixed covariate in structural equation
XI=matrix(NA, nrow=N, ncol=2)      #Explanatory latent variables
Eta=numeric(N)                     #Outcome latent variables
Y=matrix(NA, nrow=N, ncol=10)       #Observed variables

#The covariance matrix of xi
phi=matrix(c(1, 0.3, 0.3, 1), nrow=2)
```

Save the result

```
#Estimates and standard error estimates
Eu=matrix(NA, nrow=100, ncol=10);    SEu=matrix(NA, nrow=100, ncol=10)
Elam=matrix(NA, nrow=100, ncol=7);   SElam=matrix(NA, nrow=100, ncol=7)
Eb=numeric(100);                     SEb=numeric(100)
Egam=matrix(NA, nrow=100, ncol=5);   SEgam=matrix(NA, nrow=100, ncol=5)
Esgm=matrix(NA, nrow=100, ncol=10);  SEsgm=matrix(NA, nrow=100, ncol=10)
Esgd=numeric(100);                   SEsgd=numeric(100)
Ephx=matrix(NA, nrow=100, ncol=3);   SEphx=matrix(NA, nrow=100, ncol=3)

R=matrix(c(1.0, 0.3, 0.3, 1.0), nrow=2)

parameters=c("u", "lam", "b", "gam", "sgm", "sgd", "phx")
```

Initial values

```
init1=list(u=rep(0,10), lam=rep(0,7), b=0, gam=rep(0,5), psi=rep(1,10),
           psd=1, phi=matrix(c(1, 0, 0, 1), nrow=2))

init2=list(u=rep(1,10), lam=rep(1,7), b=1, gam=rep(1,5), psi=rep(2,10),
           psd=2, phi=matrix(c(2, 0, 0, 2), nrow=2))

inits=list(init1, init2)
```

WinBUGS

R2WinBUGS Package

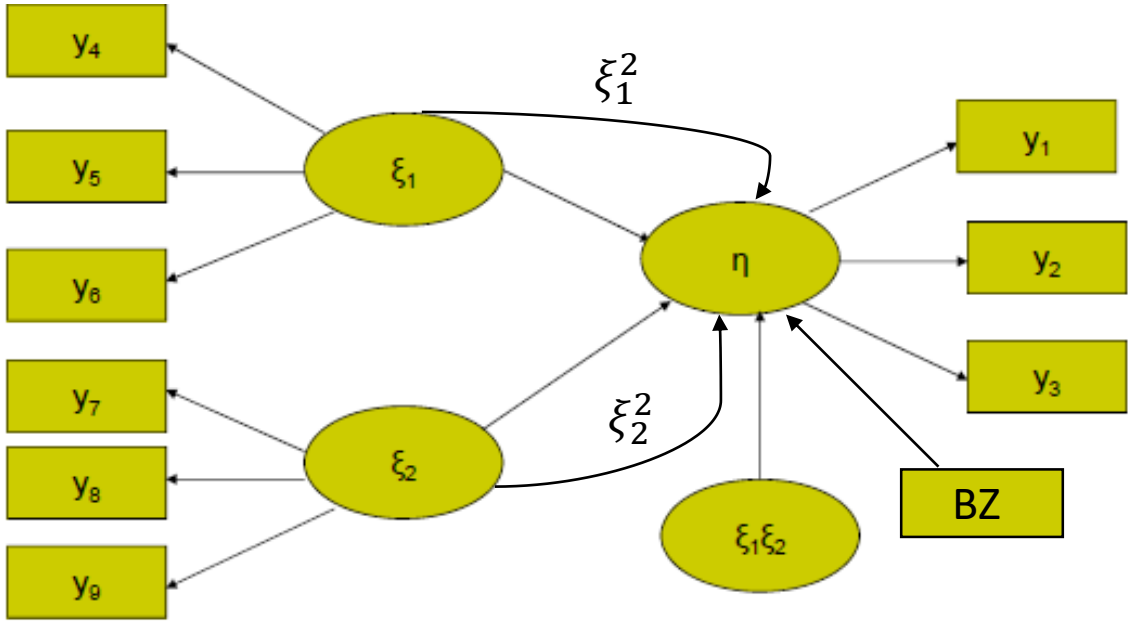
WinBUGS is an interactive program, and it is not convenient to use it to do simulation directl

For $i = 1, \ldots, n$

$$\mathbf{y}_i = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i$$

Non linear $$\boldsymbol{\eta}_i = \mathbf{B}\mathbf{d}_i + \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i) + \boldsymbol{\delta}_i$$

$$\mathbf{F}(\xi_i) = \Gamma_1\xi_{i1} + \Gamma_2\xi_{i2} + \Gamma_4\xi_{i1}\xi_{i2} + \Gamma_5\xi_{i1}^2 + \Gamma_6\xi_{i2}^2$$




```
eps=numeric(10)

for (t in 1:100){
    #Generate Data
    for (i in 1:N) {
    BZ[i]=rt(1, 5)

    XI[i,]=rmvnorm(1, c(0,0), phi)

    delta=rnorm(1, 0, sqrt(0.36))
    Eta[i]=0.5*BZ[i]+0.4*XI[i,1]+0.4*XI[i,2]+0.3*XI[i,1]*XI[i,2]
           +0.2*XI[i,1]*XI[i,1]+0.5*XI[i,2]*XI[i,2]+delta

    eps[1:3]=rnorm(3, 0, sqrt(0.3))
    eps[4:7]=rnorm(4, 0, sqrt(0.5))
    eps[8:10]=rnorm(3, 0, sqrt(0.4))
    Y[i,1]=Eta[i]+eps[1]
    Y[i,2]=0.9*Eta[i]+eps[2]
    Y[i,3]=0.7*Eta[i]+eps[3]
    Y[i,4]=XI[i,1]+eps[4]
    Y[i,5]=0.9*XI[i,1]+eps[5]
    Y[i,6]=0.7*XI[i,1]+eps[6]
    Y[i,7]=0.5*XI[i,1]+eps[7]
    Y[i,8]=XI[i,2]+eps[8]
    Y[i,9]=0.9*XI[i,2]+eps[9]
    Y[i,10]=0.7*XI[i,2]+eps[10]
}
```

Iteration 100 times

Generate data

WinBUGS

R2WinBUGS Package

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

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. A WinBUGS window will pop up and R will freeze up. The model will now run in WinBUGS. 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.

```
#Run WinBUGS
data=list(N=500, zero=c(0,0), z=BZ, R=R, y=Y)

model<-bugs(data,inits,parameters,
            model.file="D:/model.txt",
            n.chains=2,n.iter=10000,n.burnin=4000,n.thin=1,
            bugs.directory="D:/WinBUGS14")

#Save Estimates
Eu[t,]=model$mean$u;                    SEu[t,]=model$sd$u
Elam[t,]=model$mean$lam;                SElam[t,]=model$sd$lam
Eb[t]=model$mean$b;                     SEb[t]=model$sd$b
Egam[t,]=model$mean$gam;                SEgam[t,]=model$sd$gam
Esgm[t,]=model$mean$sgm;                SEsgm[t,]=model$sd$sgm
Esgd[t]=model$mean$sgd;                 SEsgd[t]=model$sd$sgd
Ephx[t,1]=model$mean$phx[1,1];          SEphx[t,1]=model$sd$phx[1,1
Ephx[t,2]=model$mean$phx[1,2];          SEphx[t,2]=model$sd$phx[1,2
Ephx[t,3]=model$mean$phx[2,2];          SEphx[t,3]=model$sd$phx[2,2
}
```

Input data

WinBUGS

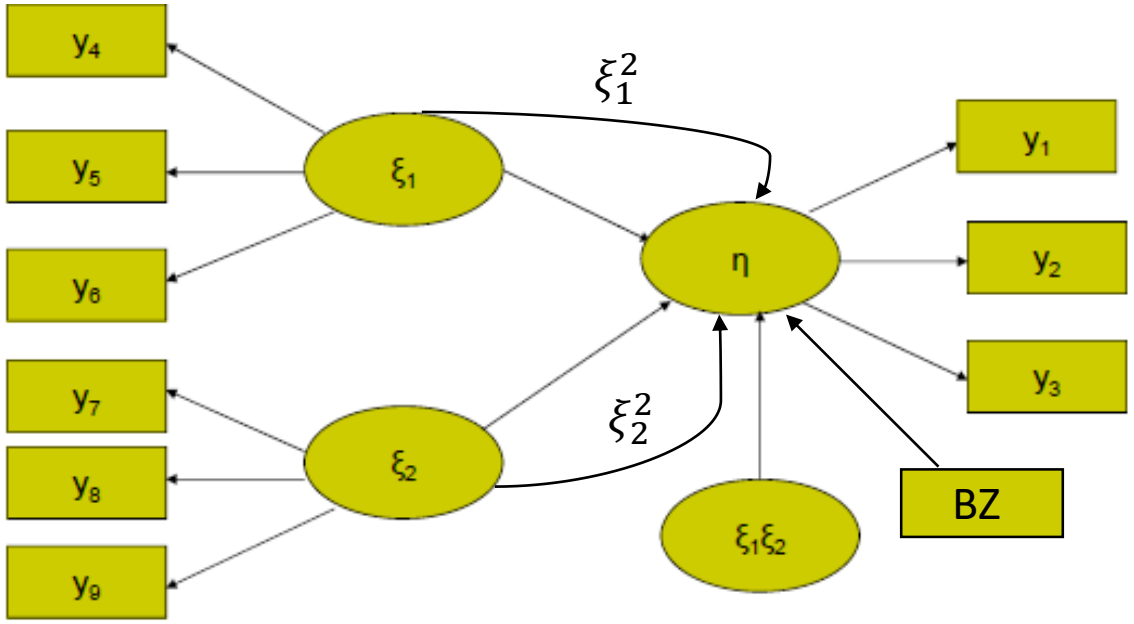
R2WinBUGS Package

For $i = 1, \ldots, n$

$$\mathbf{y}_i = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i$$

Non linear $\quad \boldsymbol{\eta}_i = \mathbf{B}\mathbf{d}_i + \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i) + \boldsymbol{\delta}_i$

$$\mathbf{F}(\xi_i) = \Gamma_1\xi_{i1} + \Gamma_2\xi_{i2} + \Gamma_4\xi_{i1}\xi_{i2} + \Gamma_5\xi_{i1}^2 + \Gamma_6\xi_{i2}^2$$




```
model {
    for (i in 1:N) {
        for (j in 1:10) { 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,5]<-u[5]+lam[3]*xi[i,1]
        mu[i,6]<-u[6]+lam[4]*xi[i,1]
        mu[i,7]<-u[7]+lam[5]*xi[i,1]
        mu[i,8]<-u[8]+xi[i,2]
        mu[i,9]<-u[9]+lam[6]*xi[i,2]
        mu[i,10]<-u[10]+lam[7]*xi[i,2]

        #structural equation
        eta[i]~dnorm(nu[i], psd)

        nu[i]<-b*z[i]+gam[1]*xi[i,1]+gam[2]*xi[i,2]+gam[3]*xi[i,1]*xi[i,2]
               +gam[4]*xi[i,1]*xi[i,1]+gam[5]*xi[i,2]*xi[i,2]

        xi[i,1:2]~dmnorm(zero[1:2], phi[1:2,1:2])
    }   #end of i

    #prior distribution
    lam[1]~dnorm(0.9,psi[2])    lam[2]~dnorm(0.7,psi[3])
    lam[3]~dnorm(0.9,psi[5])    lam[4]~dnorm(0.7,psi[6])
    lam[5]~dnorm(0.5,psi[7])    lam[6]~dnorm(0.9,psi[9])
    lam[7]~dnorm(0.7,psi[10])

    b~dnorm(0.5, psd)           gam[1]~dnorm(0.4,psd)
    gam[2]~dnorm(0.4,psd)       gam[3]~dnorm(0.3,psd)
    gam[4]~dnorm(0.2,psd)       gam[5]~dnorm(0.5,psd)

    for (j in 1:10) {
        psi[j]~dgamma(9,4)      sgm[j]<-1/psi[j]
        u[j]~dnorm(0,1)
    }

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

    phi[1:2,1:2]~dwish(R[1:2,1:2], 4)
    phx[1:2,1:2]<-inverse(phi[1:2,1:2])
} #end of model
```

The Metropolis–Hastings algorithm

For $i = 1, \ldots, n$

$$\mathbf{y}_i = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i$$

Non linear $\quad \boldsymbol{\eta}_i = \mathbf{B}\mathbf{d}_i + \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i) + \boldsymbol{\delta}_i$

$$\mathbf{F}(\xi_i) = \Gamma_1\xi_{i1} + \Gamma_2\xi_{i2} + \Gamma_4\xi_{i1}\xi_{i2} + \Gamma_5\xi_{i1}^2 + \Gamma_6\xi_{i2}^2$$

$$p(\Omega|\mathbf{Y},\theta) = \prod_{i=1}^{n} p(\omega_i|\mathbf{y}_i,\theta) \propto \prod_{i=1}^{n} p(\mathbf{y}_i|\omega_i,\theta)p(\eta_i|\xi_i,\theta)p(\xi_i|\theta$$

$$[\boldsymbol{\omega}_i|\mathbf{y}_i,\boldsymbol{\theta}] \stackrel{D}{=} N\left[\boldsymbol{\Sigma}^{*-1}\boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}(\mathbf{y}_i - \boldsymbol{\mu}) + \boldsymbol{\Sigma}^{*^{-1}}\boldsymbol{\Sigma}_\omega^{-1}\begin{pmatrix}\boldsymbol{\Pi}_0^{-1}\mathbf{B}\mathbf{d}_i \\ \mathbf{0}\end{pmatrix}, \boldsymbol{\Sigma}^{*-1}\right]$$

$$\boldsymbol{\omega}_i = (\boldsymbol{\eta}_i^T, \boldsymbol{\xi}_i^T)^T$$

$$\boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}_\omega^{-1} + \boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\boldsymbol{\Lambda}$$

$$[\boldsymbol{\omega}_i|\boldsymbol{\theta}] \stackrel{D}{=} N\left[\begin{pmatrix}\boldsymbol{\Pi}_0^{-1}\mathbf{B}\mathbf{d}_i \\ \mathbf{0}\end{pmatrix}, \boldsymbol{\Sigma}_\omega\right] \quad \boldsymbol{\Sigma}_\omega = \begin{bmatrix}\boldsymbol{\Pi}_0^{-1}(\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Gamma}^T + \boldsymbol{\Psi}_\delta)\boldsymbol{\Pi}_0^{-T} & \boldsymbol{\Pi}_0^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi} \\ \boldsymbol{\Phi}\boldsymbol{\Gamma}^T\boldsymbol{\Pi}_0^{-T} & \boldsymbol{\Phi}\end{bmatrix}$$

$$\boldsymbol{\Pi}_0 = \mathbf{I} - \boldsymbol{\Pi}$$

The Metropolis–Hastings algorithm

For $i = 1, \ldots, n$

$$\mathbf{y}_i = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i$$

Non linear $$\boldsymbol{\eta}_i = \mathbf{B}\mathbf{d}_i + \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i) + \boldsymbol{\delta}_i$$

$$\mathbf{F}(\xi_i) = \Gamma_1\xi_{i1} + \Gamma_2\xi_{i2} + \Gamma_4\xi_{i1}\xi_{i2} + \Gamma_5\xi_{i1}^2 + \Gamma_6\xi_{i2}^2$$

$$p(\boldsymbol{\Omega}|\mathbf{Y}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(\boldsymbol{\omega}_i|\mathbf{y}_i, \boldsymbol{\theta}) \propto \prod_{i=1}^{n} p(\mathbf{y}_i|\boldsymbol{\omega}_i, \boldsymbol{\theta}) p(\boldsymbol{\eta}_i|\boldsymbol{\xi}_i, \boldsymbol{\theta}) p(\boldsymbol{\xi}_i|\boldsymbol{\theta}$$

Once $\boldsymbol{\Omega}$ latent variabel is given, the nonlinear SEM becomes the familiar regression model.

$$p(\boldsymbol{\omega}_i|\mathbf{y}_i, \boldsymbol{\theta}) = \exp\left\{ -\frac{1}{2}\boldsymbol{\xi}_i^T \boldsymbol{\Phi}^{-1}\boldsymbol{\xi}_i - \frac{1}{2}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i)^T \boldsymbol{\Psi}_\epsilon^{-1}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i) \right.$$

This distribution is non-standard and complex.

$$\left. -\frac{1}{2}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i))^T \boldsymbol{\Psi}_\delta^{-1}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i) \right.$$

The **MH algorithm** is used to generate observations from the target density.

The Metropolis–Hastings algorithm

Implementation

1. Give an initial value: $\boldsymbol{\omega}_i^{(0)}$
2. At the l^{th} MH iteration with a current value $\boldsymbol{\omega}_i^{(l)}$, a new candidate $\boldsymbol{\omega}_i^{(can)}$ is generated from **proposal distribution**: $Normal(\boldsymbol{\omega}_i^{(l)}, \sigma^2\boldsymbol{\Sigma}_\omega)$
3. $\boldsymbol{\omega}_i^{(can)}$ is accepted with **acceptance probability p**:

(Generate $u \sim U(0,1)$ if $u \leq p$: $\boldsymbol{\omega}_i^{(l+1)} = \boldsymbol{\omega}_i^{(can)}$; if $u > p$: $\boldsymbol{\omega}_i^{(l+1)} = \boldsymbol{\omega}_i^{(l)}$)

$$p = \min\left\{1, \frac{p\left(\boldsymbol{\omega}_i^{(can)}|y_i,\theta\right)}{p(\boldsymbol{\omega}_i^{(l)}|y_i,\theta)}\right\}$$

$$p(\boldsymbol{\omega}_i|\mathbf{y}_i,\boldsymbol{\theta}) = \exp\left\{ -\frac{1}{2}\boldsymbol{\xi}_i^T\boldsymbol{\Phi}^{-1}\boldsymbol{\xi}_i - \frac{1}{2}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i)^T\boldsymbol{\Psi}_\epsilon^{-1}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i)\right.$$

This distribution is non-standard and complex.

$$\left. -\frac{1}{2}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i))^T\boldsymbol{\Psi}_\delta^{-1}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i)\right.$$

The **MH algorithm** is used to generate observations from the target density.

The Metropolis–Hastings algorithm

Implementation

1. Give an initial value: $\boldsymbol{\omega}_i^{(0)}$
2. At the l^{th} MH iteration with a current value $\boldsymbol{\omega}_i^{(l)}$, a new candidate $\boldsymbol{\omega}_i^{(can)}$ is generated from **proposal distribution**: $Normal(\boldsymbol{\omega}_i^{(l)}, \sigma^2 \boldsymbol{\Sigma}_\omega)$
3. $\boldsymbol{\omega}_i^{(can)}$ is accepted with **acceptance probability** $\boldsymbol{p}$

Theoretically, the proposal distribution can be arbitrary; however, it is wise to choose a standard distribution which

(i) approximates to the target distribution;

(ii) can be sampled from easily.

The tuning parameter σ^2 is chosen such that controls the acceptance rate is approximately 0.25 or more.

The optimal acceptance rate is chosen on the problem-by-problem basis

$\boldsymbol{\Sigma}_\omega$ is the Hessian matrix of $-\log p(\boldsymbol{\omega}_i|y_i,\theta)$ with $\boldsymbol{\omega}_i =$

$$\boldsymbol{\Sigma}_\omega^{-1} = \boldsymbol{\Sigma}_\delta^{-1} + \boldsymbol{\Lambda}^T \boldsymbol{\Psi}_\epsilon^{-1} \boldsymbol{\Lambda} \qquad \boldsymbol{\Delta} = [\partial \mathbf{F}(\boldsymbol{\xi}_i)/\partial \boldsymbol{\xi}_i]^T|_{\xi_i=0}$$

$$\boldsymbol{\Sigma}_\delta^{-1} = \begin{bmatrix} \boldsymbol{\Psi}_\delta^{-1} & -\boldsymbol{\Psi}_\delta^{-1}\boldsymbol{\Gamma}\boldsymbol{\Delta} \\ -\boldsymbol{\Delta}^T\boldsymbol{\Gamma}^T\boldsymbol{\Psi}_\delta^{-1} & \boldsymbol{\Phi}^{-1} + \boldsymbol{\Delta}^T\boldsymbol{\Gamma}^T\boldsymbol{\Psi}_\delta^{-1}\boldsymbol{\Gamma}\boldsymbol{\Delta} \end{bmatrix}$$

Section 2: Estimating with “R”

CFA Model

$\mathbf{y}_i$ is a $p \times 1$ observed random vector

For $i = 1, \ldots, n$ $\qquad \mathbf{y}_i = \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i$

$\boldsymbol{\omega}_i (q \times 1)$ is distributed as $N[\mathbf{0}, \boldsymbol{\Phi}]$ $\qquad$ $\boldsymbol{\epsilon}_i$ is distributed as $N[\mathbf{0}, \boldsymbol{\Psi}_\epsilon]$

Gibbs Sampler

At the j^{th} iteration with current values

$$
\begin{aligned}
\boldsymbol{\theta}_1^{(j+1)} &\text{ from } p(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2^{(j)}, \ldots, \boldsymbol{\theta}_a^{(j)}, \boldsymbol{\Omega}^{(j)}, \mathbf{Y}), \\
\boldsymbol{\theta}_2^{(j+1)} &\text{ from } p(\boldsymbol{\theta}_2 | \boldsymbol{\theta}_1^{(j+1)}, \ldots, \boldsymbol{\theta}_a^{(j)}, \boldsymbol{\Omega}^{(j)}, \mathbf{Y}), \\
&\vdots \qquad\qquad \vdots \\
\boldsymbol{\theta}_a^{(j+1)} &\text{ from } p(\boldsymbol{\theta}_a | \boldsymbol{\theta}_1^{(j+1)}, \ldots, \boldsymbol{\theta}_{a-1}^{(j+1)}, \boldsymbol{\Omega}^{(j)}, \mathbf{Y}), \\
\boldsymbol{\Omega}_1^{(j+1)} &\text{ from } p(\boldsymbol{\Omega}_1 | \boldsymbol{\theta}^{(j+1)}, \boldsymbol{\Omega}_2^{(j)}, \cdots, \boldsymbol{\Omega}_b^{(j)}, \mathbf{Y}), \\
\boldsymbol{\Omega}_2^{(j+1)} &\text{ from } p(\boldsymbol{\Omega}_2 | \boldsymbol{\theta}^{(j+1)}, \boldsymbol{\Omega}_1^{(j+1)}, \ldots, \boldsymbol{\Omega}_b^{(j)}, \mathbf{Y}), \\
&\vdots \qquad\qquad \vdots \\
\boldsymbol{\Omega}_b^{(j+1)} &\text{ from } p(\boldsymbol{\Omega}_b | \boldsymbol{\theta}^{(j+1)}, \boldsymbol{\Omega}_1^{(j+1)}, \ldots, \boldsymbol{\Omega}_{b-1}^{(j+1)}, \mathbf{Y}).
\end{aligned}
$$

CFA Model

$\mathbf{y}_i$ is a $p \times 1$ observed random vector

$$\text{For } i = 1, \ldots, n \qquad \mathbf{y}_i = \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i$$

$$\boldsymbol{\omega}_i(q \times 1) \text{ is distributed as } N[\mathbf{0}, \boldsymbol{\Phi}] \qquad \boldsymbol{\epsilon}_i \text{ is distributed as } N[\mathbf{0}, \boldsymbol{\Psi}_\epsilon]$$

STEP A: Generate a random variate $\Omega^{(j+1)}$ from the conditional distribution $[\Omega|\mathbf{Y}, \theta^{(j)}]$.

STEP B: Generate a random variate $\theta^{(j+1)}$ from the conditional distributi $[\theta|\mathbf{Y}, \Omega^{(j+1)}]$, and return to 'Step a' if necessary.

$$[\boldsymbol{\omega}_i|\mathbf{y}_i, \boldsymbol{\theta}] \overset{D}{=} N\left[(\boldsymbol{\Phi}^{-1} + \boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\mathbf{y}_i, (\boldsymbol{\Phi}^{-1} + \boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\boldsymbol{\Lambda})^{-1}\right]$$

$$IW_q[(\boldsymbol{\Omega}\boldsymbol{\Omega}^T + \mathbf{R}_0^{-1}),\ n + \rho_0]$$

$$[\psi_{\epsilon k}^{-1}|\mathbf{Y}, \boldsymbol{\Omega}] \overset{D}{=} Gamma\,[n/2 + \alpha_{0\epsilon k},\ \beta_{\epsilon k}] \quad \beta_{\epsilon k} = \beta_{0\epsilon k} + \tfrac{1}{2}(\mathbf{Y}_k^{*T}\mathbf{Y}_k^* - \mathbf{a}_k^{*T}\mathbf{A}_k^{*-1}\mathbf{a}_k^* + \boldsymbol{\Lambda}_{0k}^{*T}\mathbf{H}_{0yk}^{*-1}\boldsymbol{\Lambda}_{0}^{*}$$

$$[\boldsymbol{\Lambda}_k^*|\mathbf{Y}, \boldsymbol{\Omega}, \psi_{\epsilon k}^{-1}] \overset{D}{=} N[\mathbf{a}_k^*, \psi_{\epsilon k}\mathbf{A}_k^*] \quad \mathbf{A}_k^* = (\mathbf{H}_{0yk}^{*-1} + \boldsymbol{\Omega}_k^*\boldsymbol{\Omega}_k^{*T})^{-1} \quad \mathbf{a}_k^* = \mathbf{A}_k^*(\mathbf{H}_{0yk}^{*-1}\boldsymbol{\Lambda}_{0k}^* + \boldsymbol{\Omega}_k^*\mathbf{Y}$$

CFA Model

```
##################################MCMC#####################################
for (g in 1:n.mcmc) {

  # update ω_i
  SIG <- array(0, dim = c(q, q))
  SIG <- PHI                          # Σ_ω = Φ
  iv.SIG <- chol2inv(chol(SIG))  # Σ_ω ^ -1   solve(SIG)
  # Σ*^-1 = (Σ_ω^-1 + Λ'·Ψ_ϵ^-1·Λ)^-1
  ISG <- chol2inv(chol(iv.SIG + crossprod(iv.sqrt.PSX * LAM)))
  ##t(LAM)%*% diag(iv.PSX) %*%LAM ##t(LAM)%*% diag(iv.PSX) %*%LAM   corssprod(X)=t(X)%*%X
  # ω_i|· ~ N[Σ*^-1 · (Λ'·Ψ_ϵ^-1·y_i, Σ*^-1]
  omg.me <- omg <- ISG %*% t(LAM * iv.PSX) %*% (Y) + t(mvrnorm(n, mu = rep(0, q), Sigma = ISG))
```

$$[\boldsymbol{\omega}_i|\mathbf{y}_i,\boldsymbol{\theta}] \overset{D}{=} N\left[(\boldsymbol{\Phi}^{-1}+\boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\mathbf{y}_i,(\boldsymbol{\Phi}^{-1}+\boldsymbol{\Lambda}^T\boldsymbol{\Psi}_\epsilon^{-1}\boldsymbol{\Lambda})^{-1}\right]$$

CFA Model

```
count.LAM  <- 1
for (k in 1:p) {
  free <- Id.me[k, ]
  len <- n.me.row[k]
  Ycen <- L.me[k, !free, drop = F] %*% omg.me[!free, , drop = F]
  Uk <- omg.me[free, , drop = F]
  PSiginv <- diag(sigly, len)
  L0yk <- cbind(L0.me[k, free])
  # update Λ_y and Ψ_ϵ correponding to normal
  Yk.star <- Y[k, ] - as.vector(Ycen)
  alpha.star <- alpha.x + 0.5 * n
  beta.star <- beta.x + 0.5 * sum(Yk.star^2)
  if (len > 0) {
    # A_yk = (H_0yk^-1 + U_k·U_k')^-1, H_0yk is chosen to be diagonal
    Ayk <- chol2inv(chol(PSiginv + tcrossprod(Uk)))  #
    temp <- PSiginv %*% L0yk + Uk %*% Yk.star  # A_yk^-1·a_yk
    # a_yk = A_yk · (H_0yk^-1·Λ_0yk + U_k·Y*_k)
    ayk <- Ayk %*% temp
    # β_ϵk = β_0ϵk + 0.5(Y*k'·Y*k + Λ_0yk'·H_0yk^-1·Λ_0yk - a_yk'·A_yk^-1·a_yk)   t(x) %*%
    beta.star <- beta.star + 0.5 * (crossprod(crossprod(PSiginv, L0yk), L0yk) - crossprod(ayk , temp))
  }
```

Let $\boldsymbol{c}_k$ be the corresponding $1 \times q$ row vector

$c_{kj} = 0$ if λ_{kj} is a **fixed parameter**

$c_{kj} = 1$ if λ_{kj} is an **unknown parameter**

$r_k = c_{k1} + \cdots + c_{kq}$ Number of unknown parameter i

$$y^*_{ik} = y_{ik} - \sum_{j=1}^{q} \lambda_{kj}\omega_{ij}(1 - c_{kj})$$

$\mathbf{Y}^{*T}_k$ be the $1 \times n$ vector $(y^*_{1k}, \cdots, y^*_{n}$

$$[\psi^{-1}_{\epsilon k}|\mathbf{Y}, \boldsymbol{\Omega}] \stackrel{D}{=} Gamma\,[n/2 + \alpha_{0\epsilon k},\ \beta_{\epsilon k}] \quad \beta_{\epsilon k} = \beta_{0\epsilon k} + \tfrac{1}{2}(\mathbf{Y}^{*T}_k\mathbf{Y}^*_k - \mathbf{a}^{*T}_k\mathbf{A}^{*-1}_k\mathbf{a}^*_k + \boldsymbol{\Lambda}^{*T}_{0k}\mathbf{H}^{*-1}_{0yk}\boldsymbol{\Lambda}^*_{0}$$

$$[\boldsymbol{\Lambda}^*_k|\mathbf{Y}, \boldsymbol{\Omega}, \psi^{-1}_{\epsilon k}] \stackrel{D}{=} N\,[\mathbf{a}^*_k, \psi_{\epsilon k}\mathbf{A}^*_k] \quad \mathbf{A}^*_k = (\mathbf{H}^{*-1}_{0yk} + \boldsymbol{\Omega}^*_k\boldsymbol{\Omega}^{*T}_k)^{-1} \quad \mathbf{a}^*_k = \mathbf{A}^*_k(\mathbf{H}^{*-1}_{0yk}\boldsymbol{\Lambda}^*_{0k} + \boldsymbol{\Omega}^*_k\mathbf{Y}$$

CFA Model

```
    # ψ_ϵk|· ~ Inv-Gamma(n/2 + α_0ϵk, β_ϵk)
    iv.PSX[k]<-rgamma(1, shape = alpha.star, rate = beta.star)
    PSX[k] <- 1 / iv.PSX[k]
    iv.sqrt.PSX[k] <- sqrt(iv.PSX[k])

    if (len > 0){
      # Λ_yk|· ~ N[a_yk, ψ_ϵk·A_yk]
      L.me[k, free] <- mvrnorm(1, ayk, Sig = PSX[k] * Ayk)
      LAM[k, ] <- L.me[k, ]
      if (n.LAM.row[k] > 0)
        data.LAM[g, count.LAM:(count.LAM + n.LAM.row[k] - 1)]<-LAM[k, Id.LAM[k, ]]

      count.LAM <- count.LAM + n.LAM.row[k]
    }
  }
```

$$[\psi_{\epsilon k}^{-1}|\mathbf{Y},\boldsymbol{\Omega}] \stackrel{D}{=} Gamma\,[n/2+\alpha_{0\epsilon k},\ \beta_{\epsilon k}] \quad \beta_{\epsilon k} = \beta_{0\epsilon k} + \tfrac{1}{2}(\mathbf{Y}_k^{*T}\mathbf{Y}_k^{*} - \mathbf{a}_k^{*T}\mathbf{A}_k^{*-1}\mathbf{a}_k^{*} + \boldsymbol{\Lambda}_{0k}^{*T}\mathbf{H}_{0yk}^{*-1}\boldsymbol{\Lambda}_{0}$$

$$[\boldsymbol{\Lambda}_k^{*}|\mathbf{Y},\boldsymbol{\Omega},\psi_{\epsilon k}^{-1}] \stackrel{D}{=} N\,[\mathbf{a}_k^{*},\psi_{\epsilon k}\mathbf{A}_k^{*}] \quad \mathbf{A}_k^{*} = (\mathbf{H}_{0yk}^{*-1} + \boldsymbol{\Omega}_k^{*}\boldsymbol{\Omega}_k^{*T})^{-1} \quad \mathbf{a}_k^{*} = \mathbf{A}_k^{*}(\mathbf{H}_{0yk}^{*-1}\boldsymbol{\Lambda}_{0k}^{*} + \boldsymbol{\Omega}_k^{*}\mathbf{Y}$$

CFA Model

```
# update Φ
# Φ|· ~ IW_q2[Ω_2·Ω_2' + R_0^-1, n + ρ_0]
if (q2 > 0) {
  iv.PHI <- rWishart(1, rho0 + n, chol2inv(chol(tcrossprod(omg) + R0)))[, , 1]
  c.iv.PHI <- chol(iv.PHI)
  PHI <- chol2inv(c.iv.PHI)
  data.PHI[g, ] <- as.vector(PHI)
}
```

$$IW_q[(\mathbf{\Omega}\mathbf{\Omega}^T + \mathbf{R}_0^{-1}),\ n + \rho_0]$$

CFA Model

With Mean structure and Covariates

$\mathbf{y}_i$ is a $p \times 1$ observed random vector

For $i = 1, \ldots, n$ $\quad \mathbf{y}_i = \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i \longrightarrow \boldsymbol{y_i} = \boldsymbol{\mu} + \boldsymbol{A c_i} + \boldsymbol{\Lambda \omega_i} + \boldsymbol{\epsilon_i}$

$\boldsymbol{\omega}_i (q \times 1)$ is distributed as $N[\mathbf{0}, \boldsymbol{\Phi}]$ $\quad \boldsymbol{\epsilon}_i$ is distributed as $N[\mathbf{0}, \boldsymbol{\Psi}_\epsilon]$

```
# update ω_i
SIG <- array(0, dim = c(q, q))
SIG <- PHI                          # Σ_ω = Φ
iv.SIG <- chol2inv(chol(SIG))  # Σ_ω ^ -1   solve(SIG)
# Σ*^-1 = (Σ_ω^-1 + Λ'·Ψ_ϵ^-1·Λ)^-1
ISG <- chol2inv(chol(iv.SIG + crossprod(iv.sqrt.PSX * LAM))) ##t(LAM)%*% diag(iv.PSX) %*%LAM
# ω_i|· ~ N[Σ*^-1 · (Λ'·Ψ_ϵ^-1·(y_i - μ - A·c_i), Σ*^-1]
omg.me[(2 + r1):(1 + r1 + q), ] <- omg <- ISG %*% t(LAM * iv.PSX)%*% (Y - mu - A %*% c) +
  t(mvrnorm(n, mu = rep(0, q), Sigma = ISG))
```

$$[\boldsymbol{\omega}_i | \mathbf{y}_i, \boldsymbol{\theta}] \overset{D}{=} N\left[(\boldsymbol{\Phi}^{-1} + \boldsymbol{\Lambda}^T \boldsymbol{\Psi}_\epsilon^{-1} \boldsymbol{\Lambda})^{-1} \boldsymbol{\Lambda}^T \boldsymbol{\Psi}_\epsilon^{-1} \mathbf{y}_i, (\boldsymbol{\Phi}^{-1} + \boldsymbol{\Lambda}^T \boldsymbol{\Psi}_\epsilon^{-1} \boldsymbol{\Lambda})^{-1}\right]$$

$$\boldsymbol{y_i} - \boldsymbol{\mu} - \boldsymbol{A c_i}$$

CFA Model With Mean structure and Covariates

```
omg.me <- rbind(rep(1, n), c, omg)   # u = (1', c', ω')'

count.LAM <- count.A <- 1
for (k in 1:p) {
  free <- Id.me[k, ]
  len <- n.me.row[k]
  Ycen <- L.me[k, !free, drop = F] %*% omg.me[!free, , drop = F]
  Uk <- omg.me[free, , drop = F]
  PSiginv <- diag(sigly, len)
  LOyk <- cbind(LO.me[k, free])
  # update Λ_y and Ψ_ϵ correponding to normal
  Yk.star <- Y[k, ] - as.vector(Ycen)
  alpha.star <- alpha.x + 0.5 * n
  beta.star <- beta.x + 0.5 * sum(Yk.star^2)
  if (len > 0) {
    # A_yk = (H_0yk^-1 + U_k·U_k')^-1, H_0yk is chosen to be di
    Ayk <- chol2inv(chol(PSiginv + tcrossprod(Uk)))  #
    temp <- PSiginv %*% LOyk + Uk %*% Yk.star  # A_yk^-1·a_yk
    # a_yk = A_yk · (H_0yk^-1·Λ_0yk + U_k·Y*_k)
    ayk <- Ayk %*% temp
```

$$\boldsymbol{y_i = \mu + Ac_i + \Lambda\omega_i + \epsilon_i}$$

$$\boldsymbol{u_i = \left(1^T, c_i^T, \omega_i^T\right)^T \quad \Lambda_y = (\mu, A, \Lambda}$$

$$\boldsymbol{y_i = \Lambda_y u_i + \epsilon_i}$$

$$\beta_{\epsilon k} =$$

$$\beta_{0\epsilon k} + \frac{1}{2}(\mathbf{Y}_k^{*T}\mathbf{Y}_k^* - \mathbf{a}_k^{*T}\mathbf{A}_k^{*-1}\mathbf{a}_k^* + \mathbf{\Lambda}_{0k}^{*T}\mathbf{H}_{0yk}^{*-1}\mathbf{\Lambda}_{0k}^*$$

$$\mathbf{A}_k^* = (\mathbf{H}_{0yk}^{*-1} + \mathbf{\Omega}_k^*\mathbf{\Omega}_k^{*T})^{-1}$$

$$\mathbf{a}_k^* = \mathbf{A}_k^*(\mathbf{H}_{0yk}^{*-1}\mathbf{\Lambda}_{0k}^* + \mathbf{\Omega}_k^*\mathbf{Y}_k^*)$$

SEM Model

For $i = 1, \ldots, n$

$$\boldsymbol{y_i} = \boldsymbol{\mu} + \boldsymbol{A c_i} + \boldsymbol{\Lambda \omega_i} + \boldsymbol{\epsilon_i}$$

$$\boldsymbol{\eta}_i = \mathbf{B}\mathbf{d}_i + \boldsymbol{\Pi}\boldsymbol{\eta}_i + \boldsymbol{\Gamma}\boldsymbol{\xi}_i + \boldsymbol{\delta}_i$$

$$[\boldsymbol{\omega}_i | \mathbf{y}_i, \boldsymbol{\theta}] \stackrel{D}{=} N\left[\boldsymbol{\Sigma}^{*-1}\boldsymbol{\Lambda}^T\boldsymbol{\Psi}_{\epsilon}^{-1}(\boldsymbol{y_i} - \boldsymbol{\mu} - \boldsymbol{A c_i}) + \boldsymbol{\Sigma}^{*^{-1}}\boldsymbol{\Sigma}_{\omega}^{-1}\begin{pmatrix}\boldsymbol{\Pi}_0^{-1}\mathbf{B}\mathbf{d}_i \\ \mathbf{0}\end{pmatrix}, \boldsymbol{\Sigma}^{*-1}\right]$$

$$IW_q[(\boldsymbol{\Omega}\boldsymbol{\Omega}^T + \mathbf{R}_0^{-1}),\ n + \rho_0]$$

$$\boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}_{\omega}^{-1} + \boldsymbol{\Lambda}^T\boldsymbol{\Psi}_{\epsilon}^{-1}\boldsymbol{\Lambda}$$

Normal – (Inverted) gamma distribution

$$[\psi_{\epsilon k}^{-1} | \mathbf{Y}, \boldsymbol{\Omega}] \stackrel{D}{=} Gamma\,[n/2 + \alpha_{0\epsilon k},\ \beta_{\epsilon k}]$$

$$[\psi_{\delta k}^{-1} | \boldsymbol{\Omega}] \stackrel{D}{=} \text{Gamma}[n/2 + \alpha_{0\delta k},\ \beta_{\delta k}]$$

$$[\boldsymbol{\Lambda}_k^* | \mathbf{Y}, \boldsymbol{\Omega}, \psi_{\epsilon k}^{-1}] \stackrel{D}{=} N\,[\mathbf{a}_k^*, \psi_{\epsilon k}\mathbf{A}_k^*]$$

$$[\boldsymbol{\Lambda}_{\omega k} | \boldsymbol{\Omega}, \psi_{\delta k}^{-1}] \stackrel{D}{=} N[\mathbf{a}_{\omega k}, \psi_{\delta k}\mathbf{A}_{\omega k}]$$

SEM Model

$$N\left[\boldsymbol{\Sigma}^{*-1}\boldsymbol{\Lambda}^{T}\boldsymbol{\Psi}_{\epsilon}^{-1}(\mathbf{y}_i-\mathbf{A}\mathbf{c}_i)+\boldsymbol{\Sigma}^{*^{-1}}\boldsymbol{\Sigma}_{\omega}^{-1}\begin{pmatrix}\boldsymbol{\Pi}_0^{-1}\mathbf{B}\mathbf{d}_i\\ \mathbf{0}\end{pmatrix},\ \boldsymbol{\Sigma}^{*-1}\right.$$

$$\boldsymbol{\Sigma}^{*}=\boldsymbol{\Sigma}_{\omega}^{-1}+\boldsymbol{\Lambda}^{T}\boldsymbol{\Psi}_{\epsilon}^{-1}\boldsymbol{\Lambda}$$

```
# update ω_i
SIG.omg <- array(0, dim = c(q, q))
# Σ_ω = [Π_0^-1 · (Γ·Φ·Γ' + Ψ_δ) · Π_0^-1'   Π_0^-1·Γ·Φ]
#       [Φ·Γ'·Π_0^-1'                        Φ         ]
if (q2 > 0)
  SIG.omg[(q1 + 1):q, (q1 + 1):q] <- PHI
if (q1 > 0) {
  iv.PI0 <- solve(diag(1, q1) - PI)
  SIG.omg[1:q1, 1:q1] <- iv.PI0 %*% (GA %*% PHI %*% t(GA) + diag(PSD, nrow = q1)) %*% t(iv.PI0)
  SIG.omg[1:q1, (q1 + 1):q] <- iv.PI0 %*% GA %*% PHI
  SIG.omg[(q1 + 1):q, 1:q1] <- t(SIG.omg[1:q1, (q1 + 1):q])
}
iv.SIG.omg <- chol2inv(chol(SIG.omg))  # # Σ_ω^-1   solve(SIG.omg)
# Σ*^-1 = (Σ_ω^-1 + Λ'·Ψ_ϵ^-1·Λ)^-1
ISG <- chol2inv(chol(iv.SIG.omg+ crossprod(iv.sqrt.PSX * LAM))) ##t(LAM)%*% diag(iv.PSX) %*%LAM   corssprod(X)=t(X)%*

# ω_i|· ~ N[Σ*^-1 · (Λ'·Ψ_ϵ^-1·(y_i- μ -A·c_i))+ Σ*^-1 ·Σ_ω^-1 · (Π_0^-1·B·d_i), Σ*^-1]
#          [                                                    (     0      )       ]

omg.me[(2 + r1):(1 + r1 + q), ] <- omg.se[(r2 + 1):n.se, ] <-
omg <- ISG %*% t(LAM * iv.PSX) %*% (Y - mu - A %*% c) +
  ISG %*% iv.SIG.omg %*% rbind(iv.PI0 %*% (B %*% d), array(0, dim = c(q2, n)))+t(mvrnorm(n, mu = rep(0, q), Sigma = I
```

SEM Model

```
# update Λ_ω and Ψ_δ
count.B <- count.PI <- count.GA <- 1
for (k in 1:q1) {
  free <- Id.se[k, ]
  len <- n.se.row[k]
  etacen <- L.se[k, !free, drop = F] %*% omg.se[!free, , drop = F]
  Vk <- omg.se[free, , drop = F]
  iH0wk <- diag(sigbi, len)
  L0wk <- cbind(L0.se[k, free])

  Eta.k <- omg[k, ] - as.vector(etacen)
  alpha.star <- alpha.d + 0.5 * n
  beta.star <- beta.d + 0.5 * sum(Eta.k^2)

  if (len > 0) {
    # A_ωk = (H_0ωk^-1 + V_k·V_k')^-1, H_0ωk is chosen to be diagonal
    Awk <- chol2inv(chol(iH0wk + tcrossprod(Vk)))
    temp <- iH0wk %*% L0wk + Vk %*% Eta.k  # A_ωk^-1·a_ωk
    # a_ωk = A_ωk · (H_0ωk^-1·Λ_0ωk + V_k·Ξ_k)
    awk <- Awk %*% temp
    # β_δk = β_0δk + 0.5(Ξ_k'·Ξ_k + Λ_0ωk'·H_0ωk^-1·Λ_0ωk - a_ωk'·A_ωk^-1·a_ωk)
    beta.star <- beta.star + 0.5 * (crossprod(crossprod(iH0wk, L0wk), L0wk) - crossprod(awk, temp))
  }
```

$$\beta_{\delta k} = \beta_{0\delta k} + \frac{1}{2}\left(\boldsymbol{\Xi}_k^T \boldsymbol{\Xi}_k - \mathbf{a}_{\omega k}^T \mathbf{A}_{\omega k}^{-1} \mathbf{a}_{\omega k} + \boldsymbol{\Lambda}_{0\omega k}^T \mathbf{H}_{0\omega k}^{-1} \boldsymbol{\Lambda}_{0\omega}\right.$$

$$\mathbf{A}_{\omega k} = (\mathbf{H}_{0\omega k}^{-1} + \mathbf{V}_k \mathbf{V}_k^T)^{-1}$$

$$\mathbf{a}_{\omega k} = \mathbf{A}_{\omega k}(\mathbf{H}_{0\omega k}^{-1} \boldsymbol{\Lambda}_{0\omega k} + \mathbf{V}_k \boldsymbol{\Xi}_k)$$

SEM Model

```
if (len > 0) {
  # A_ωk = (H_0ωk^-1 + V_k·V_k')^-1, H_0ωk is chosen to be diagonal
  Awk <- chol2inv(chol(iH0wk + tcrossprod(Vk)))
  temp <- iH0wk %*% L0wk + Vk %*% Eta.k  # A_ωk^-1·a_ωk
  # a_ωk = A_ωk · (H_0ωk^-1·Λ_0ωk + V_k·Ξ_k)
  awk <- Awk %*% temp
  # β_δk = β_0δk + 0.5(Ξ_k'·Ξ_k + Λ_0ωk'·H_0ωk^-1·Λ_0ωk - a_ωk'·A_ωk^-1·a_ωk)
  beta.star <- beta.star + 0.5 * (crossprod(crossprod(iH0wk, L0wk), L0wk) - crossprod(awk, temp))
}

iv.PSD[k] <- rgamma(1, shape = alpha.star, rate = beta.star)
PSD[k] <- 1 / iv.PSD[k]
iv.sqrt.PSD[k] <- sqrt(iv.PSD[k])

if (len > 0) {
  # Λ_ωk|· ~ N[a_ωk, ψ_δk·A_ωk]
  L.se[k, free] <- mvrnorm(1, awk, Sig = PSD[k] * Awk)
  if (r2 > 0)
    B[k, ] <- L.se[k, 1:r2]
  if (q1 > 0)
    PI[k, ] <- L.se[k, (r2 + 1):(r2 + q1)]
  if (q2 > 0)
    GA[k, ] <- L.se[k, (r2 + q1 + 1):n.se]
```

$$[\psi_{\delta k}^{-1}|\mathbf{\Omega}] \stackrel{D}{=} \text{Gamma}[n/2 + \alpha_{0\delta k}, \beta_{\delta k}]$$

$$[\mathbf{\Lambda}_{\omega k}|\mathbf{\Omega}, \psi_{\delta k}^{-1}] \stackrel{D}{=} N[\mathbf{a}_{\omega k}, \psi_{\delta k}\mathbf{A}_{\omega k}]$$

SEM Model

```
# update Φ
# Φ|· ~ IW_q2[Ω_2·Ω_2' + R_0^-1, n + ρ_0]
if (q2 > 0) {
  iv.PHI <- rWishart(1, rho0 + n, chol2inv(chol(tcrossprod(omg[(q1 + 1):q, ,drop = F]) + R0)))[, , 1]
  c.iv.PHI <- chol(iv.PHI)
  PHI <- chol2inv(c.iv.PHI)
  data.PHI[g, ] <- as.vector(PHI)
}
```

$$IW_q[(\mathbf{\Omega}_2\mathbf{\Omega}_2^T + \mathbf{R}_0^{-1}),\ n + \rho_0]$$

$$\mathbf{\Omega}_2 = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n)$$

Nonlinear SEM Model

```r
is.linear <- "F"        # T -- linear SEM, F -- non-linear SEM
if(is.linear=="T"){
  t <- q2
}
##################define nonlinear SE################
if(is.linear=="F"){
  t <- 5                # dim of F(ξ), the non-linear term in SE (including linear term)
  Non.lin <- function(xi) {  # F(ξ)
    fxi <- array(0, dim = c(t, n))
    fxi <- rbind(xi,xi^2,xi[1,]*xi[2,])
    return(fxi)
  }
  dFdxi <- function(x, t, q2) {  # Δ = [∂F(ξ_i)/∂ξ_i]'|(ξ_i=0)
    delt <- array(0, dim = c(t, q2))
    delt[1:q2,1:q2] <- diag(1, q2)
    return(delt)
  }
  # tuning parameters in proposal distribution in MH algorith
  sig2.mh <- 1          # control acceptance rate for ξ
}
```

```r
#define_constant
n <- dim(Y)[2]          # sample size
p <- dim(Y)[1]          # dimension of Y, the response variables
q <- q1 + q2            # dim of ω = (η, ξ), the latent variables
r1 <- dim(c)[1]         # dim of c, covariates in ME
r2 <- dim(d)[1]         # dim of d, covariates in SE
n.se <- r2 + q1 + t     # dim of linear terms in SE

xi <- array(dim = c(q2, n))           # ξ -- explanatory latent varia
eta <- array(dim = c(q1, n))          # η -- outcome latent variables
omg <- array(dim = c(q, n))           # ω = (η, ξ)
if (q1 > 0) {
  omg.se <- array(dim = c(n.se, n))   # G(ω) = (d, η, F(ξ))
  fxi <- array(0, dim = c(t, n))      # F(ξ) -- non-linear terms
}
```

Nonlinear SEM Model

proposal distribution: $Normal(\boldsymbol{\omega}_i^{(l)}, \sigma^2 \boldsymbol{\Sigma}_\omega)$

```
} else {  # for non-linear SEM, MH algorithm
    ISG <- crossprod(iv.sqrt.PSX * LAM)   #Λ'·Ψ_ϵ^-1·Λ
    if (q2 > 0)
      ISG[(q1 + 1):q, (q1 + 1):q] <- ISG[(q1 + 1):q, (q1 + 1):q] + iv.PHI    #Phi Φ^-1
    if (q1 > 0) {
      PI0 <- diag(1, q1) - PI       # Π_0=I-Π
      Delt <- dFdxi(rep(0, q2), t, q2)
      sqrt.ISG <- array(0, dim = c(q1, q))
      sqrt.ISG[, 1:q1] <- PI0
      sqrt.ISG[, (q1 + 1):q] <- -GA %*% Delt     # Γ*Delta
      ISG <- ISG + crossprod(iv.sqrt.PSD * sqrt.ISG)
    }  # Σ_ω^-1 = [Π_0' *Ψ_δ ^ -1 * Π_0    - Π_0'*Ψ_δ^-1*Γ*Delta            ] + Λ'·Ψ_ϵ^-1·Λ
    #            [-Delta'*Γ'*Ψ_δ^-1*Π_0   Φ^-1 + Delta'*Γ'*Ψ_δ^-1*Γ*Delta  ]

    SIG <- chol2inv(chol(ISG))  # Σ_ω
    SIG <- sig2.mh * SIG        # sigma^2 * Σ_ω
    # Proposal distribution NORMAL
    omg.new <- omg + t(mvrnorm(n, mu = rep(0, q), Sigma = SIG))
```

$$\boldsymbol{\Sigma}_\omega^{-1} = \boldsymbol{\Sigma}_\delta^{-1} + \boldsymbol{\Lambda}^T \boldsymbol{\Psi}_\epsilon^{-1} \boldsymbol{\Lambda}$$

$$\boldsymbol{\Sigma}_\delta^{-1} = \begin{bmatrix} \boldsymbol{\Pi}_0^T \boldsymbol{\Psi}_\delta^{-1} \boldsymbol{\Pi}_0 & -\boldsymbol{\Pi}_0^T \boldsymbol{\Psi}_\delta^{-1} \boldsymbol{\Gamma} \boldsymbol{\Delta} \\ -\boldsymbol{\Delta}^T \boldsymbol{\Gamma}^T \boldsymbol{\Psi}_\delta^{-1} \boldsymbol{\Pi}_0 & \boldsymbol{\Phi}^{-1} + \boldsymbol{\Delta}^T \boldsymbol{\Gamma}^T \boldsymbol{\Psi}_\delta^{-1} \boldsymbol{\Gamma} \end{bmatrix}$$

Nonlinear SEM Model

```
# likelihood of new candidate ω_i
# log.like1 store -2log p(Y_i|.) = (y_i - A·c_i - Λ·ω_i)' · Ψ_ε^-1 · (y_i - A·c_i - Λ·ω_i),
log.like1 <- colSums((iv.sqrt.PSX * (Y - mu - A %*% c - LAM %*% omg.new))^2)
# log.like2 store -2log p(η_i|.) = (η_i - Λ_ω·G(ω_i))' · Ψ_δ^-1 · (η_i - Λ_ω·G(ω_i)), i = 1
if (q1 > 0) {
  temp <- PI0 %*% omg.new[1:q1, ] - GA %*% (Non.lin(omg.new[(q1 + 1):q, ]))
  if (r2 > 0)
    temp <- temp - B %*% d
  temp <- iv.sqrt.PSD * temp
  log.like2 <- colSums(temp^2)
}
# log.like3 store -2log p(ξ_i|.) = ξ_i' · Φ^-1 · ξ_i, i = 1, ..., n.
if (q2 > 0)
  log.like3 <- colSums((c.iv.PHI %*% omg.new[(q1 + 1):q, ])^2)
# return log(P(ω_i|.))
ll2 <- (-0.5 * (log.like1 + log.like2 + log.like3))
```

$$p(\boldsymbol{\omega}_i|\mathbf{y}_i,\boldsymbol{\theta}) = \exp\left\{ -\frac{1}{2}\boldsymbol{\xi}_i^T\boldsymbol{\Phi}^{-1}\boldsymbol{\xi}_i - \frac{1}{2}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i)^T\boldsymbol{\Psi}_\epsilon^{-1}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i) \right.$$
$$\left. - \frac{1}{2}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i))^T\boldsymbol{\Psi}_\delta^{-1}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i)) \right\}$$

Nonlinear SEM Model

```
# likelihood of old ω_i
# log.like1 store -2log p(Y_i|.) = (y_i - A·c_i - Λ·ω_i)' · Ψ_ε^-1 · (y_i - A·c_i - Λ·ω_i),
log.like1 <- colSums((iv.sqrt.PSX * (Y - mu - A %*% c - LAM %*% omg))^2)
# log.like2 store -2log p(η_i|.) = (η_i - Λ_ω·G(ω_i))' · Ψ_δ^-1 · (η_i - Λ_ω·G(ω_i)), i = 1
if (q1 > 0) {
  temp <- PI0 %*% omg[1:q1, ] - GA %*% (Non.lin(omg[(q1 + 1):q, ]))
  if (r2 > 0)
    temp <- temp - B %*% d
  temp <- iv.sqrt.PSD * temp
  log.like2 <- colSums(temp^2)
}
# log.like3 store -2log p(ξ_i|.) = ξ_i' · Φ^-1 · ξ_i, i = 1, ..., n.
if (q2 > 0)
  log.like3 <- colSums((c.iv.PHI %*% omg[(q1 + 1):q, ])^2)
# return log(P(ω_i|.))
ll1 <- (-0.5 * (log.like1 + log.like2 + log.like3))
```

$$p(\boldsymbol{\omega}_i|\mathbf{y}_i,\boldsymbol{\theta}) = \exp\left\{ -\frac{1}{2}\boldsymbol{\xi}_i^T\boldsymbol{\Phi}^{-1}\boldsymbol{\xi}_i - \frac{1}{2}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i)^T\boldsymbol{\Psi}_\epsilon^{-1}(\mathbf{y}_i - \mathbf{A}\mathbf{c}_i - \boldsymbol{\Lambda}\boldsymbol{\omega}_i) \right.$$
$$\left. - \frac{1}{2}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i))^T\boldsymbol{\Psi}_\delta^{-1}(\boldsymbol{\eta}_i - \mathbf{B}\mathbf{d}_i - \boldsymbol{\Pi}\boldsymbol{\eta}_i - \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}_i)) \right\}$$

Nonlinear SEM Model

$$p = \min\left\{1, \frac{p\left(\boldsymbol{\omega}_i^{(can)} | y_i, \theta\right)}{p(\boldsymbol{\omega}_i^{(l)} | y_i, \theta)}\right\}$$

```
# acceptance ratio for ω_i
p.acpt <- exp(ll2 - ll1)
acpt <- (runif(n) < p.acpt)  # whether new ω_i accepted
# update ω_i
omg.me[(2 + r1):(1 + r1 + q), acpt] <- omg[, acpt] <- omg.new[, acpt]
if (q1 > 0) {
  omg.se[(r2 + 1):(r2 + q1), acpt] <- omg[1:q1, acpt]
  omg.se[(r2 + q1 + 1):n.se, acpt] <- Non.lin(omg[(q1 + 1):q, ])[, acpt]
}
n.acpt.omg <- n.acpt.omg + acpt
```

End of Supplementary