Homework 2

Xiaocheng Zhou (1155184323)

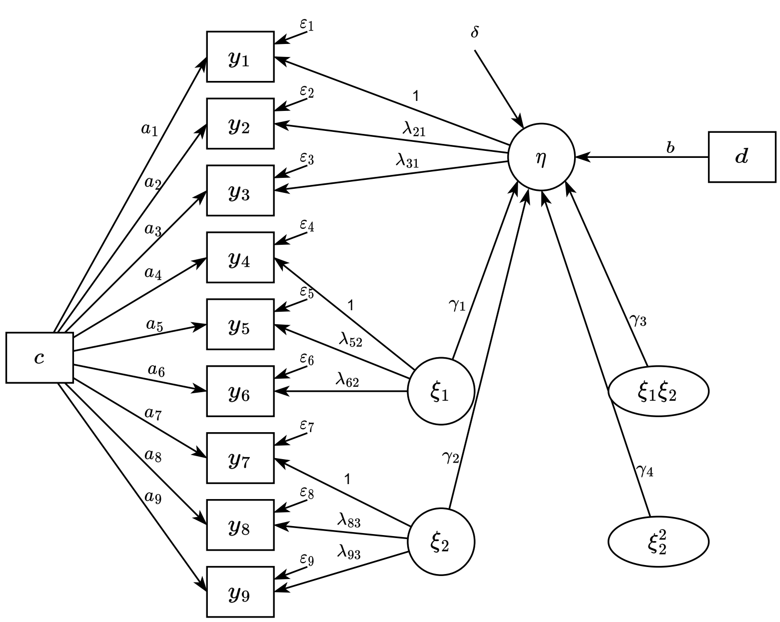
March 22, 2023

Answers

1. Consider a non-linear SEM defined as follows (matrix form)

$$\begin{bmatrix} y_{i1} \\ y_{i2} \\ y_{i3} \\ y_{i4} \\ y_{i5} \\ y_{i6} \\ y_{i7} \\ y_{i8} \\ y_{i9} \end{bmatrix} = \begin{bmatrix} \mu_1 & a_1 \\ \mu_2 & a_2 \\ \mu_3 & a_3 \\ \mu_4 & a_4 \\ \mu_5 & a_5 \\ \mu_6 & a_6 \\ \mu_7 & a_7 \\ \mu_8 & a_8 \\ \mu_9 & a_9 \end{bmatrix} \begin{bmatrix} 1 \\ c_i \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \lambda_{83} \\ 0 & 0 & \lambda_{93} \end{bmatrix} \begin{bmatrix} \eta_i \\ \xi_{i1} \\ \xi_{i2} \end{bmatrix} + \begin{bmatrix} \varepsilon_{i1} \\ \varepsilon_{i2} \\ \varepsilon_{i3} \\ \varepsilon_{i4} \\ \varepsilon_{i5} \\ \varepsilon_{i6} \\ \varepsilon_{i7} \\ \varepsilon_{i8} \\ \varepsilon_{i9} \end{bmatrix}$$

$$\eta_i = bd_i + \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \end{bmatrix} \begin{bmatrix} \xi_{i1} \\ \xi_{i2} \\ \xi_{i1}\xi_{i2} \\ \xi_{i2}^2 \end{bmatrix} + \delta_i \tag{1}$$



(a) Set true values for the model parameters. Generate data from the model and conduct Bayesian analysis on the basis of 10 replications.
The true values of parameters set for this question are listed as follow, and 10 data sets are generated based on the true parameters. The script of data generating and Bayesian analysis with `WinBUGS` is attached as Appendix.

$$\begin{aligned}
\boldsymbol{\mu}_{1:9} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{a}_{1:9} &= \begin{bmatrix} 0.2 & -0.2 & 0.4 & 0.3 & -0.2 & 0.4 & 0.5 & -0.4 & 0.3 \end{bmatrix} \\
\boldsymbol{\lambda}_{\{21,31,52,62,83,93\}} &= \begin{bmatrix} 0.9 & 0.6 & 0.7 & 0.9 & 0.8 & 0.6 \end{bmatrix} \\
b &= 0.5 \\
\boldsymbol{\gamma}_{1:4} &= \begin{bmatrix} 0.4 & 0.3 & -0.5 & 0.1 \end{bmatrix} \\
\boldsymbol{\Phi} &= \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix} \ (\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Phi})) \\
\boldsymbol{\psi}_{\varepsilon 1:9} &= \begin{bmatrix} 0.3 & 0.3 & 0.3 & 0.4 & 0.4 & 0.4 & 0.5 & 0.5 & 0.5 \end{bmatrix} \ (\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\psi}_\varepsilon))) \\
\psi_\delta &= 0.36 \ (\delta \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\psi}_\delta)))
\end{aligned}$$

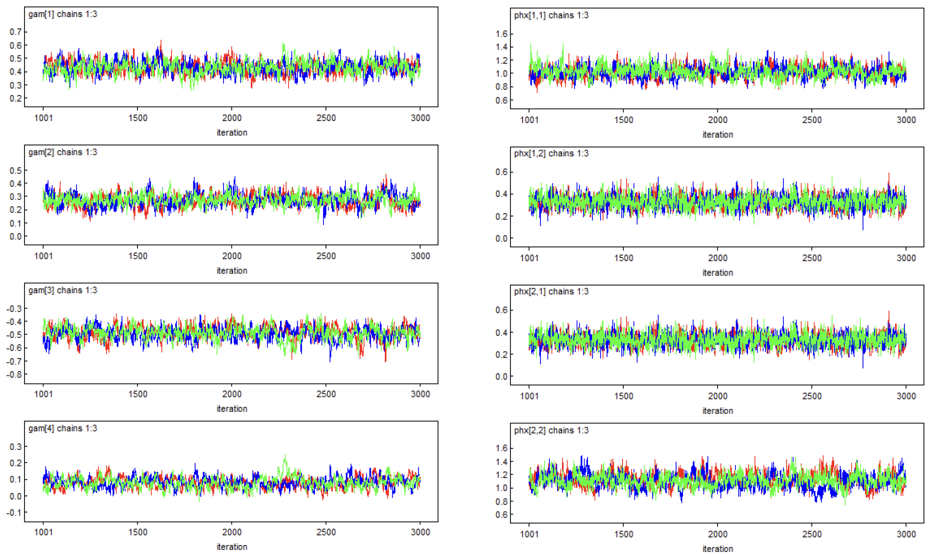
Table 1: Three sets of initial values are set for iterative estimation

Parameters	Set 1	Set 2	Set 3
$\boldsymbol{\mu}_{1:9}^{(0)}$	$\mathbf{0}$	$\mathbf{1}$	$-\mathbf{1}$
$\boldsymbol{a}_{1:9}^{(0)}$	$\mathbf{0}$	$\mathbf{1}$	$-\mathbf{1}$
$\boldsymbol{\lambda}_{\{21,31,52,62,83,93\}}^{(0)}$	$\mathbf{0}$	$\mathbf{1}$	$-\mathbf{1}$
$b^{(0)}$	0	1	-1
$\boldsymbol{\gamma}_{1:4}^{(0)}$	$\mathbf{0}$	$\mathbf{1}$	$-\mathbf{1}$
$\boldsymbol{\Phi}^{(0)}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
$\boldsymbol{\psi}_{\varepsilon 1:9}^{(0)}$	$\mathbf{1}$	$2 \times \mathbf{1}$	$0.5 \times \mathbf{1}$
$\psi_\delta^{(0)}$	1	2	0.5

(b) Demonstrate how to check convergence of the model.

- **Method 1**: check the plots of estimation process. If the curves starting from different initial values meet together, then the model converges well. Figure 1 shows two illustration of convergence of estimates, suggesting our estimation converges.
- **Method 2**: check the `Rhat` column, potential scale reduction factor (or EPSR introduced in Lecture slides), reported by `WinBUGS` summary. If it is very close to 1, then the model converges well. Our results are very close to 1, also suggesting the good convergence.

(c) Use Bias and RMSE to summarize the estimation results.
In the results of `WinBUGS`, we regard the mean of burn-in estimates as the output estimate $\hat{\theta}$, then the bias ($\frac{1}{R}\sum_{r=1}^{R}\hat{\theta}_r - \theta$) and RMSE ($\sqrt{\frac{1}{R}\sum_{r=1}^{R}(\hat{\theta}_r - \theta)^2}$) are reported as follow



(a) Estimates for $\boldsymbol{\gamma}_{1:4}$

(b) Estimates for $\boldsymbol{\Phi}$

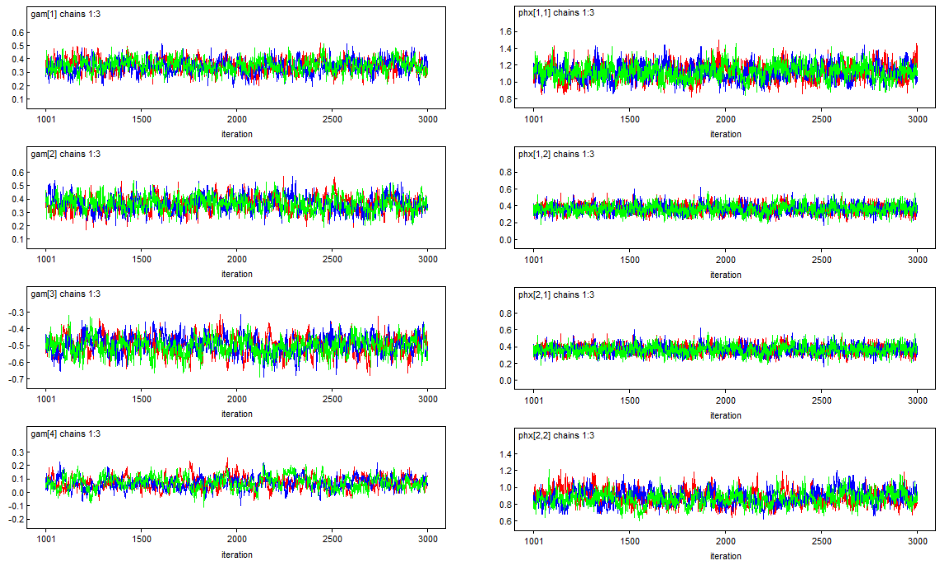
Figure 1: Some example estimates with Prior 1 from iterations 1001 – 3000

Table 2: Bias and RMSE of the above 10 replicated estimates

parameters	evaluation (top Bias, bottom RMSE)
$\hat{\boldsymbol{\mu}}_{1:9}$	bias: $\begin{bmatrix} 0.032 & 0.019 & 0.003 & 0.003 & 0.023 & -0.008 & 0.002 & 0.003 & -0.006 & -0.006 \end{bmatrix}$ RMSE: $\begin{bmatrix} 0.058 & 0.053 & 0.033 & 0.046 & 0.042 & 0.027 & 0.049 & 0.050 & 0.036 \end{bmatrix}$
$\hat{\boldsymbol{a}}_{1:9}$	bias: $\begin{bmatrix} 0.022 & 0.019 & 0.008 & 0.032 & 0.019 & 0.024 & 0.005 & -0.003 & -0.006 \end{bmatrix}$ RMSE: $\begin{bmatrix} 0.034 & 0.031 & 0.020 & 0.047 & 0.026 & 0.034 & 0.041 & 0.027 & 0.024 \end{bmatrix}$
$\hat{\boldsymbol{\lambda}}_{\{21,31,52,62,83,93\}}$	bias: $\begin{bmatrix} -0.003 & 0.002 & 0.009 & -0.010 & 0.014 & -0.011 \end{bmatrix}$ RMSE: $\begin{bmatrix} 0.050 & 0.028 & 0.037 & 0.038 & 0.064 & 0.047 \end{bmatrix}$
$\hat{b}$	bias: -0.005 RMSE: 0.002
$\hat{\boldsymbol{\gamma}}_{1:4}$	bias: $\begin{bmatrix} -0.004 & -0.001 & 0.013 & -0.008 \end{bmatrix}$ RMSE: $\begin{bmatrix} 0.051 & 0.046 & 0.051 & 0.050 \end{bmatrix}$
$\hat{\boldsymbol{\Phi}}$	bias: $\begin{bmatrix} 0.028 & 0.019 \\ * & -0.046 \end{bmatrix}$ RMSE: $\begin{bmatrix} 0.090 & 0.067 \\ * & 0.083 \end{bmatrix}$
$\hat{\boldsymbol{\psi}}_{\varepsilon\ 1:9}$	bias: $\begin{bmatrix} 0.002 & 0.022 & -0.004 & 0.002 & 0.010 & -0.026 & -0.006 & -0.012 & -0.001 \end{bmatrix}$ RMSE: $\begin{bmatrix} 0.031 & 0.027 & 0.019 & 0.021 & 0.033 & 0.034 & 0.074 & 0.036 & 0.029 \end{bmatrix}$
$\hat{\psi}_{\delta}$	bias: 0.009 RMSE: 0.002

(d) Show your prior inputs and check whether the Bayesian analysis is sensitive to the inputs

My prior parameters used in the above process are listed in the Table 3 Prior 1. Now, consider the Prior 2, which is with more divergence and variance, and repeat the process. We found both of the estimates plot (Figure 2) and potential scale reduction factors suggest good convergences of this model. Moreover, the bias and RMSE also nearly do not change.



(a) Estimates for $\boldsymbol{\gamma}_{1:4}$ (b) Estimates for $\boldsymbol{\Phi}$

Figure 2: Some example estimates with Prior 2 from iterations 1001 – 3000

Table 3: Two sets of prior distributions are set for sensitivity analysis

Parameters	Prior 1	Prior 2
μ_k	$\mathcal{N}(0,1)$	$\mathcal{N}(1,2)$
$[a_k \vert \psi_{\varepsilon k}]$	$\mathcal{N}(0.3,\psi_{\varepsilon k})$	$\mathcal{N}(1,\psi_{\varepsilon k})$
$[\lambda_{kj} \vert \psi_{\varepsilon k}]$	$\mathcal{N}(0.5,\psi_{\varepsilon k})$	$\mathcal{N}(1,\psi_{\varepsilon k})$
$[b \vert \psi_\delta]$	$\mathcal{N}(0.5,\psi_\delta)$	$\mathcal{N}(1,\psi_\delta)$
$[\boldsymbol{\gamma} \vert \psi_\delta]$	$\mathcal{N}(\begin{bmatrix}0.4 & 0.3 & 0.5 & 0.5\end{bmatrix}^\intercal, \psi_\delta \mathbf{I})$	$\mathcal{N}(\mathbf{1}, \psi_\delta \mathbf{I})$
$\boldsymbol{\Phi}^{-1}$	$\text{Wishart}(\begin{bmatrix}1 & 0.3\\ 0.3 & 1\end{bmatrix}, 4)$	$\text{Wishart}(\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}, 4)$
$\psi_{\varepsilon k}^{-1}$	$\text{Gamma}(9,4)$	$\text{Gamma}(6,10)$
ψ_δ^{-1}	$\text{Gamma}(9,4)$	$\text{Gamma}(6,10)$

2. Continue to Q1, use Bayesian model comparison statistics, including Bayes factor and DIC, and the 10 datasets generated in Q1 to answer the following questions:
   (a) Compare the non-linear SEM in Q1 with its linear SEM counterpart.

$$\eta_i = bd_i + \begin{bmatrix}\gamma_1 & \gamma_2\end{bmatrix}\begin{bmatrix}\xi_{i1}\\ \xi_{i2}\end{bmatrix} + \delta_i \tag{2}$$

   (b) Consider a new non-linear SEM by modifying the structural equation in Q1 as follow. Compare the non-linear SEM in Q1 with this new model.

$$\eta_i = bd_i + \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 \end{bmatrix} \begin{bmatrix} \xi_{i1} \\ \xi_{i2} \\ \xi_{i1}\xi_{i2} \\ \xi_{i1}^2 \\ \xi_{i2}^2 \end{bmatrix} + \delta_i \tag{3}$$

The BFs of $\frac{P(\mathbf{Y}|\text{SEM}(1))}{P(\mathbf{Y}|\text{SEM}(2))}$ and $\frac{P(\mathbf{Y}|\text{SEM}(3))}{P(\mathbf{Y}|\text{SEM}(1))}$ and DICs of SEM(1), SEM(2), and SEM(3) for the 10 datasets above are listed in Table 4.

The average bayes factor of true model vs linear model is always greater than 20 and the true model always has smaller DIC than the linear model, suggesting the true model is preferred when compared with the linear one. The bayes factors of the alternative non-linear model vs true model always negative and the true model but has larger DICs than the alternative non-linear model at most of time. It is worth noting that the alternative non-linear model and true model are very similar under DIC, so the decision only based on DIC could cause problem. We should collect multiple model selection criteria for a comprehensive comparison.

Table 4: Bayes Factors and DIC for model comparison

Dataset No.	$\log \text{BF}_{12}$	$\log \text{BF}_{31}$	DIC_1	DIC_2	DIC_3
1	44.16	−3.21	9669.49	9719.17	9659.78
2	36.34	−3.33	9649.49	9705.57	9638.87
3	47.56	−3.24	9530.99	9586.83	9519.41
4	22.78	−2.29	9616.88	9653.28	9614.9
5	28.68	−0.68	9797.67	9840.05	9789.4
6	30.74	−1.96	9702.01	9747.22	9699.86
7	43.54	−3.28	9724.36	9798.65	9715.72
8	36.10	−2.45	9513.45	9596.35	9512.15
9	51.27	−3.13	9707.44	9785.75	9711.02
10	47.29	−3.17	9677.17	9744.56	9668.17
Mean	38.84	−2.67	9658.90	9717.74	9652.93
SD	38.85	0.85	86.65	84.26	86.74

Appendix

Data generate and parameters estimation

```
library(mvtnorm)
library(R2WinBUGS)

timestamp = strftime(Sys.time(), "%Y%m%d-%H")
winBUGS.path = "D:/pkgs/WinBUGS14/"

iter = 10
NY = 9  # dimension of Y
Neta = 1  # dimension of eta
Nxi = 2  # dimension of xi
Ngam = 4  # dimension of gamma

N = 500
BD = numeric(N)
BC = numeric(N)
XI = matrix(NA, nrow = N, ncol = 2)
Eta = numeric(N)
Y = matrix(NA, nrow = N, ncol = NY)

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

# Estimates and standard error estimates
# store a set of generated parameters from prior, true parameters
Eu = matrix(NA, nrow = iter, ncol = NY)
SEu = matrix(NA, nrow = iter, ncol = NY)
Elam = matrix(NA, nrow = iter, ncol = NY - Neta - Nxi)
SElam = matrix(NA, nrow = iter, ncol = NY - Neta - Nxi)
Eb = numeric(iter)
SEb = numeric(iter)
Ea = matrix(NA, nrow = iter, ncol = NY)
SEa = matrix(NA, nrow = iter, ncol = NY)
Egam = matrix(NA, nrow = iter, ncol = Ngam)
SEgam = matrix(NA, nrow = iter, ncol = Ngam)
Esgm = matrix(NA, nrow = iter, ncol = NY)
SEsgm = matrix(NA, nrow = iter, ncol = NY)
Esgd = numeric(iter)
SEsgd = numeric(iter)
Ephx = matrix(NA, nrow = iter, ncol = 3)
SEphx = matrix(NA, nrow = iter, ncol = 3)

R = matrix(c(1, 0.3, 0.3, 1), nrow = 2)

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

init1 = list(u = rep(0, NY), lam = rep(0, NY - Neta - Nxi), b = 0,
             a = rep(0, NY), gam = rep(0, Ngam), psi = rep(1, NY),
             psd = 1, phi = matrix(c(1, 0, 0, 1), nrow = 2))

init2 = list(u = rep(1, NY), lam = rep(1, NY - Neta - Nxi), b = 1,
             a = rep(1, NY), gam = rep(1, Ngam), psi = rep(2, NY),
             psd = 2, phi = matrix(c(2, 0, 0, 2), nrow = 2))

init3 = list(u = rep(-1, NY), lam = rep(-1, NY - Neta - Nxi), b = -1,
             a = rep(-1, NY), gam = rep(-1, Ngam), psi = rep(0.5, NY),
             psd = 0.5, phi = matrix(c(0.5, 0, 0, 0.5), nrow = 2))
# psi is sgm, psd is sgd, phi is phx

inits = list(init1, init2, init3)

eps = numeric(NY)

datapath = paste0(getwd(),'/data')
dir.create(datapath, showWarnings = FALSE, recursive = TRUE)

for (t in 1:iter) {
```

```
  iterpath = paste0(getwd(),"/rep", t)
  dir.create(iterpath, showWarnings = FALSE, recursive = TRUE)
  # generate data
  for (i in 1:N) {
    BD[i] = rt(1, 5)
    BC[i] = rt(1, 5)

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

    delta = rnorm(1, 0, sqrt(0.36))

    Eta[i] = 0.5 * BD[i] + 0.4 * XI[i, 1] + 0.3 * XI[i, 2] - 0.5 * XI[i, 1] * XI[i, 2] + 0.1 *
        XI[i, 2] * XI[i, 2] + delta

    eps[1:3] = rnorm(3, 0, sqrt(0.3))
    eps[4:6] = rnorm(3, 0, sqrt(0.4))
    eps[7:9] = rnorm(3, 0, sqrt(0.5))

    Y[i, 1] = 0.2 * BC[i] + Eta[i] + eps[1]
    Y[i, 2] = -0.2 * BC[i] + 0.9 * Eta[i] + eps[2]
    Y[i, 3] = 0.4 * BC[i] + 0.6 * Eta[i] + eps[3]
    Y[i, 4] = 0.3 * BC[i] + XI[i, 1] + eps[4]
    Y[i, 5] = -0.2 * BC[i] + 0.7 * XI[i, 1] + eps[5]
    Y[i, 6] = 0.4 * BC[i] + 0.9 * XI[i, 1] + eps[6]
    Y[i, 7] = 0.5 * BC[i] + XI[i, 2] + eps[7]
    Y[i, 8] = -0.4 * BC[i] + 0.8 * XI[i, 2] + eps[8]
    Y[i, 9] = 0.3 * BC[i] + 0.6 * XI[i, 2] + eps[9]


  }

  # Run WINBUGS
  data = list(N = 500, zero = c(0, 0), d = BD, c = BC, R = R, y = Y)

  write.table(Y, paste(datapath, "Y-", t, ".txt", sep = ""))
  write.table(BD, paste(datapath, "BD-", t, ".txt", sep = ""))
  write.table(BC, paste(datapath, "BC-", t, ".txt", sep = ""))

  model = bugs(data, inits, parameters, model.file = paste0(getwd(),"/../model.txt"),
               n.chains = 3, n.iter = 3000, n.burnin = 1000,
               n.thin = 1, bugs.directory = winBUGS.path,
               working.directory = iterpath, debug = FALSE)

  # 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
  Ea[t, ] = model$mean$a
  SEa[t, ] = model$sd$a
  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]

  print(model$summary)
}



# True values for evaluating the estimates
```

```r
Tu = matrix(rep(0, 9), nrow = 1)
Ta = matrix(c(0.2, -0.2, 0.4, 0.3, -0.2, 0.4, 0.5, -0.4, 0.3), nrow = 1)
Tlam = matrix(c(0.9, 0.6, 0.7, 0.9, 0.8, 0.6), nrow = 1)
Tb = 0.5
Tgam = matrix(c(0.4, 0.3, -0.5, 0.1), nrow = 1)
Tphx = matrix(c(1, 0.3, 1), nrow = 1)
Tsgm = matrix(rep(c(0.3, 0.4, 0.5), each = 3), nrow = 1)
Tsgd = 0.36

resultlst = list(
  Tu = Tu,
  Ta = Ta,
  Tlam = Tlam,
  Tb = Tb,
  Tgam = Tgam,
  Tphx = Tphx,
  Tsgm = Tsgm,
  Tsgd = Tsgd
)

save(Tu, Ta, Tlam, Tb, Tgam, Tphx, Tsgm, Tsgd,
     file = paste0(getwd(),"/trueparam.RData"))


# report result

reportq13 <- function(est, tru) {
  if (length(tru)==1){
    return(list(
      mean = mean(est - tru),
      rmse = sqrt(mean((est - tru)^2))
    ))

  }else{
    return(list(
      mean = apply(sweep(est, 2, tru), 2, mean),
      rmse = sqrt(apply(sweep(est, 2, tru)^2, 2, mean))
    ))
  }
}

reportq13(Eu, Tu)
reportq13(Ea, Ta)
reportq13(Elam, Tlam)
reportq13(Eb, Tb)
reportq13(Egam, Tgam)
reportq13(Ephx, Tphx)
reportq13(Esgm, Tsgm)
reportq13(Esgd, Tsgd)

resultlst = list(
  Eu = Eu,
  SEu = SEu,
  Elam = Elam,
  SElam = SElam,
  Eb = Eb,
  SEb = SEb,
  Ea = Ea,
  SEa = SEa,
  Egam = Egam,
  SEgam = SEgam,
  Esgm = Esgm,
  SEsgm = SEsgm,
  Esgd = Esgd,
  SEsgd = SEsgd,
  Ephx = Ephx,
  SEphx = SEphx

)

save(resultlst, file = paste0(getwd(),'/model-', timestamp, ".RData"))
```

Model comparison by DIC

```
library(mvtnorm)
library(R2WinBUGS)

timestamp = strftime(Sys.time(), "%Y%m%d-%H")
winBUGS.path = "D:/pkgs/WinBUGS14/"
datapath = paste0(getwd(),'/data')


iter = 10
NY = 9  # dimension of Y
Neta = 1  # dimension of eta
Nxi = 2  # dimension of xi
Ngam = 4  # dimension of gamma
R = matrix(c(1, 0.3, 0.3, 1), nrow = 2)


dic = numeric(iter)


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

init1 = list(u = rep(0, NY), lam = rep(0, NY - Neta - Nxi), b = 0,
             a = rep(0, NY), gam = rep(0, Ngam), psi = rep(1, NY),
             psd = 1, phi = matrix(c(1, 0, 0, 1), nrow = 2))

inits = list(init1)

for (r in 1:iter) {
  iterpath = paste0(getwd(),"/dictrue",r)
  dir.create(iterpath, showWarnings = FALSE, recursive = TRUE)

  # load previous dataset
  Y = as.matrix(read.table(paste0(datapath, "/Y-", r, ".txt")))
  BD = read.table(paste0(datapath, "/BD-", r, ".txt"))$x
  BC = read.table(paste0(datapath, "/BC-", r, ".txt"))$x

  # Run WINBUGS
  data = list(N = 500, zero = c(0, 0), d = BD, c = BC, R = R, y = Y)

  model = bugs(data, inits, parameters, model.file = paste0(getwd(),"/../model.txt"),
               n.chains = 1, n.iter = 3000, n.burnin = 1000,
               n.thin = 1, bugs.directory = winBUGS.path,
               working.directory = iterpath, debug = FALSE)

  dic[r] = model$DIC
}

resultlst = list(
  dic = dic, dic.mean = mean(dic), dic.sd = sd(dic)
)

print(resultlst)

save(resultlst, file = paste0(getwd(),'/dictrue-', timestamp, ".RData"))
```

True model (Model (1))

```
model{
  for (i in 1:N) {
      for (j in 1:9) {
          y[i, j] ~ dnorm(mu[i, j], psi[j])
      }
      mu[i, 1] <- u[1] + a[1] * c[i] + eta[i]
      mu[i, 2] <- u[2] + a[2] * c[i] + lam[1] * eta[i]
      mu[i, 3] <- u[3] + a[3] * c[i] + lam[2] * eta[i]
      mu[i, 4] <- u[4] + a[4] * c[i] + xi[i, 1]
```

```
        mu[i, 5] <- u[5] + a[5] * c[i] + lam[3] * xi[i, 1]
        mu[i, 6] <- u[6] + a[6] * c[i] + lam[4] * xi[i, 1]
        mu[i, 7] <- u[7] + a[7] * c[i] + xi[i, 2]
        mu[i, 8] <- u[8] + a[8] * c[i] + lam[5] * xi[i, 2]
        mu[i, 9] <- u[9] + a[9] * c[i] + lam[6] * xi[i, 2]

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

        nu[i] <- b * d[i] + gam[1] * xi[i, 1] + gam[2] * xi[i, 2] + gam[3] * xi[i, 1] * xi[i, 2]
            + gam[4] * 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(0.5, psi[2])
    lam[2] ~ dnorm(0.5, psi[3])
    lam[3] ~ dnorm(0.5, psi[5])
    lam[4] ~ dnorm(0.5, psi[6])
    lam[5] ~ dnorm(0.5, psi[8])
    lam[6] ~ dnorm(0.5, psi[9])

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

    for (j in 1:9) {
        psi[j] ~ dgamma(9, 4)
        sgm[j] <- 1/psi[j]
        u[j] ~ dnorm(0, 1)
        a[j] ~ dnorm(0.3, psi[j])
    }  # end of j

    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
```

Linear model (Model (2))

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

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

        nu[i] <- b * d[i] + gam[1] * xi[i, 1] + gam[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(0.5, psi[2])
  lam[2] ~ dnorm(0.5, psi[3])
  lam[3] ~ dnorm(0.5, psi[5])
  lam[4] ~ dnorm(0.5, psi[6])
  lam[5] ~ dnorm(0.5, psi[8])
  lam[6] ~ dnorm(0.5, psi[9])

  b ~ dnorm(0.5, psd)
  gam[1] ~ dnorm(0.4, psd)
  gam[2] ~ dnorm(0.3, psd)

  for (j in 1:9) {
      psi[j] ~ dgamma(9, 4)
      sgm[j] <- 1/psi[j]
      u[j] ~ dnorm(0, 1)
      a[j] ~ dnorm(0.3, psi[j])
  }  # end of j

  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
```

Alternative model (Model (3))

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

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

      nu[i] <- b * d[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(0.5, psi[2])
  lam[2] ~ dnorm(0.5, psi[3])
  lam[3] ~ dnorm(0.5, psi[5])
  lam[4] ~ dnorm(0.5, psi[6])
  lam[5] ~ dnorm(0.5, psi[8])
  lam[6] ~ dnorm(0.5, psi[9])

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

```

```
  for (j in 1:9) {
      psi[j] ~ dgamma(9, 4)
      sgm[j] <- 1/psi[j]
      u[j] ~ dnorm(0, 1)
      a[j] ~ dnorm(0.3, psi[j])
  }  # end of j

  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
```

Model comparison (BF)

```
library(R2WinBUGS)  #Load R2WinBUGS package

timestamp = strftime(Sys.time(), "%Y%m%d-%H")
winBUGS.path = "D:/pkgs/WinBUGS14/"
datapath = paste0(getwd(),'/data')

iter = 10
cut = 20

NY = 9  # dimension of Y
Neta = 1  # dimension of eta
Nxi = 2  # dimension of xi
Ngam = 4  # dimension of gamma
R = matrix(c(1, 0.3, 0.3, 1), nrow = 2)


lbf = numeric(iter)


parameters = c("ubar")

init1 = list(u = rep(0, NY), lam = rep(0, NY - Neta - Nxi), b = 0,
             a = rep(0, NY), gam = rep(0, Ngam), psi = rep(1, NY),
             psd = 1, phi = matrix(c(1, 0, 0, 1), nrow = 2))

inits = list(init1)

# Path sampling
for (r in 1:10) {
  iterpath = paste0(getwd(), "/bflinear", r)
  dir.create(iterpath, showWarnings = FALSE, recursive = TRUE)

  # load previous dataset
  Y = as.matrix(read.table(paste0(datapath, "/Y-", r, ".txt")))
  BD = read.table(paste0(datapath, "/BD-", r, ".txt"))$x
  BC = read.table(paste0(datapath, "/BC-", r, ".txt"))$x

  data = list(N = 500, zero = c(0, 0), d = BD, c = BC, R = R, y = Y, t = NA)

  u = numeric(cut)
  for (i in 1:cut) {
    data$t <- (i - 1)/(cut - 1)

    model = bugs(data, inits, parameters,
                 model.file = paste0(iterpath, "/../model_BF_linear.txt"),
                 n.chains = 1, n.iter = 3000,
                 n.burnin = 1000, n.thin = 1, bugs.directory = winBUGS.path,
                 working.directory = iterpath)

    u[i] <- model$mean$ubar

  }

```

```
  # Caluate log Bayes factor
  logBF = 0
  for (i in 1:(cut - 1)) {
    logBF = logBF + (u[i + 1] + u[i])/(2 * (cut - 1))
  }

  lbf[r] = logBF
}

resultlst = list(
  lbf = lbf, lbf.mean = mean(lbf), lbf.sd = sd(lbf)
)

save(resultlst, file = paste0(getwd(), '/bflinear-', timestamp, ".RData"))
```

Linear model ($t = 0$) vs true model ($t = 1$) for BF

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

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

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

        uu[i] <- (eta[i] - nu[i]) * psd * (gam[3] * xi[i, 1] * xi[i, 2] + gam[4] * xi[i, 2] *
            xi[i, 2])

        xi[i, 1:2] ~ dmnorm(zero[1:2], phi[1:2, 1:2])

    }  # end of i

    ubar <- sum(uu[])

    # prior distribution
    lam[1] ~ dnorm(0.5, psi[2])
    lam[2] ~ dnorm(0.5, psi[3])
    lam[3] ~ dnorm(0.5, psi[5])
    lam[4] ~ dnorm(0.5, psi[6])
    lam[5] ~ dnorm(0.5, psi[8])
    lam[6] ~ dnorm(0.5, psi[9])

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

    for (j in 1:9) {
        psi[j] ~ dgamma(9, 4)
        sgm[j] <- 1/psi[j]
        u[j] ~ dnorm(0, 1)
        a[j] ~ dnorm(0.3, psi[j])
    }  # end of j

    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
```

True model ($t = 0$) vs alternative non-linear model ($t = 1$)

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

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

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

        uu[i] <- (eta[i] - nu[i]) * psd * (gam[4] * xi[i, 1] * xi[i, 1])

        xi[i, 1:2] ~ dmnorm(zero[1:2], phi[1:2, 1:2])

    }  # end of i

    ubar <- sum(uu[])

    # prior distribution
    lam[1] ~ dnorm(0.5, psi[2])
    lam[2] ~ dnorm(0.5, psi[3])
    lam[3] ~ dnorm(0.5, psi[5])
    lam[4] ~ dnorm(0.5, psi[6])
    lam[5] ~ dnorm(0.5, psi[8])
    lam[6] ~ dnorm(0.5, psi[9])

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

    for (j in 1:9) {
        psi[j] ~ dgamma(9, 4)
        sgm[j] <- 1/psi[j]
        u[j] ~ dnorm(0, 1)
        a[j] ~ dnorm(0.3, psi[j])
    }  # end of j

    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
```