Homework 2 of STAT 5020

WANG Lijun

March 20, 2019

Q1

(a)

Solution. The matrix form is

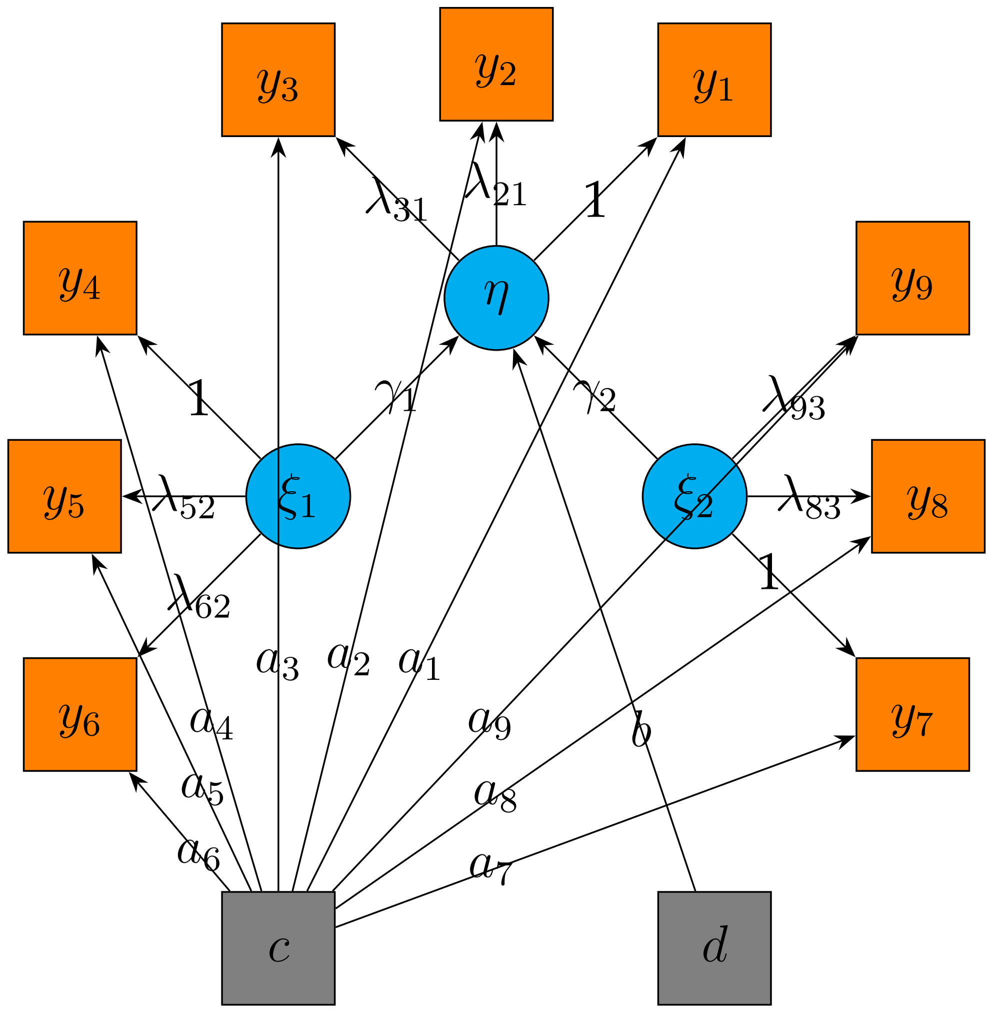
$$\boldsymbol{y}_i = \boldsymbol{\mu} + \mathbf{A}c_i + \boldsymbol{\Lambda}\boldsymbol{\omega}_i + \boldsymbol{\epsilon}_i \tag{1}$$

$$\eta_i = bd_i + \boldsymbol{\Gamma}\mathbf{F}(\boldsymbol{\xi}) + \delta_i\,, \tag{2}$$

where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_9]'$, $\boldsymbol{\epsilon}_i = [\epsilon_{1i}, \ldots, \epsilon_{9i}]'$, $\mathbf{A} = [a_1, \ldots, a_9]'$, $\omega_i = [\eta_i, \xi_{1i}, \xi_{2i}]'$, $\boldsymbol{\Gamma} = [\gamma_1, \ldots, \gamma_4]$, $\mathbf{F}(\boldsymbol{\xi}) = [\xi_{1i}, \xi_{2i}, \xi_{1i}^2, \xi_{2i}^2]'$ and

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1 & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \lambda_{83} \\ 0 & 0 & \lambda_{93} \end{bmatrix}.$$

The path diagram is



Assign the true values for the parameters as follows:

$$
\begin{aligned}
\lambda_{21} &= \lambda_{52} = \lambda_{83} = 0.9 \\
\lambda_{31} &= \lambda_{62} = \lambda_{93} = 0.7 \\
\mathbf{\Gamma} &= [0.4, 0.4, 0.3, 0.2] \\
\mu &= \mathbf{0} \\
\Psi_\epsilon &= \text{diag}\{0.3, 0.5, 0.4, 0.3, 0.5, 0.4, 0.3, 0.5, 0.4\} \\
\mathbf{A} &= [0.3, 0.7, 0.9, 0.3, 0.7, 0.9, 0.3, 0.7, 0.9]' \\
\psi_\delta &= 0.36 \\
b &= 0.5 \\
\Phi &= \begin{bmatrix} 1.0 & 0.3 \\ 0.3 & 1.0 \end{bmatrix}
\end{aligned}
$$

■

(b)

Solution. The model is defined in the file "q1.stan", and write function "genData(::Int)" to generate data. I use Julia interface of Stan to do simulation. The main source code file is "sem.jl" (Appendix A). ■

(c)

Solution. Traceplots can help us monitor the convergences, as showed in the Figure 3.

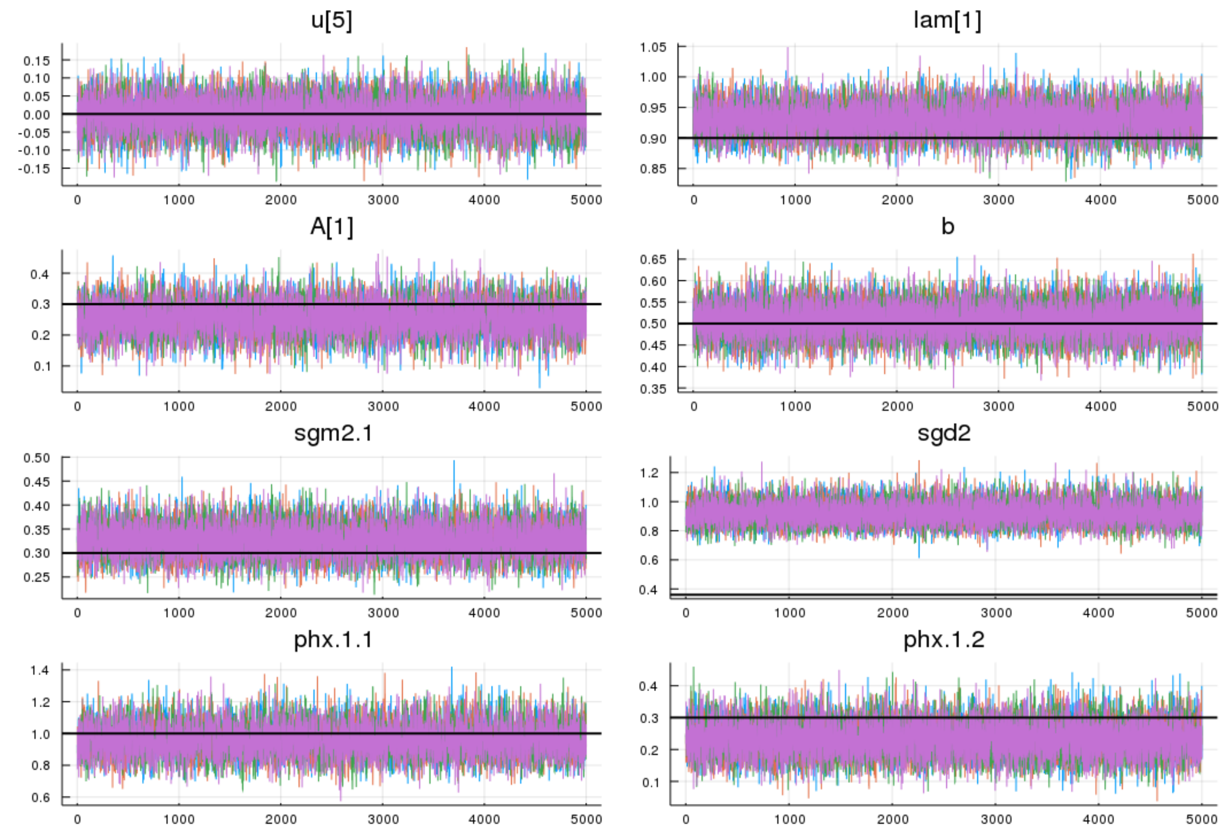


Figure 1: From top left to bottom right, plots represents four chains of observations corresponding to $\mu_5, a_1, b, \psi_{\epsilon 1}, \psi_\delta, \phi_{11}, \phi_{12}$ generated for different initial values, represented by different colors, and the black horizontal line is the true value for each parameter.

All parameters, except ψ_δ, would converges to its true value, I failed to figure out the reason why ψ_δ has a bad estimate, although converged.

■

(d)

Solution. The average absolute bias over 10 replication is

```
0.5608748105195
```

```
0.5004287968147001
0.4085746076235901
0.11994920809580101
0.083809744878196
0.11343451980021901
0.09338658459368002
0.09904335539176999
0.073445407798145
0.06120660459999999
0.045355297100000004
0.11440984739999992
0.08715910830000001
0.06571353839999998
0.07282894119999997
0.07830605475321001
0.07481963200000002
0.042005978500000004
0.09095389469137
0.07389264989999998
0.09300910469999996
0.07002369334684
0.08174603611999998
0.07883658880000005
0.07234076055
0.13401097347324004
0.16109005924780004
0.1418024249128
0.08712839048288
0.029333237500000015
0.06368324929999998
0.038501469300000014
0.06948034530000007
0.050057778099999985
0.04357015180000001
0.0557030602
0.037594690599999994
0.04499504720000005
0.37590144850000007
0.21113397350000007
0.13076420807688
0.13076420807688
0.18479104950000008
```

where the parameter order is

$$\mu_1, \ldots, \mu_9, \lambda_1, \ldots, \lambda_6, a_1, \ldots, a_9, b, \gamma_1, \ldots, \gamma_4, \psi_{\epsilon 1}, \ldots, \psi_{\epsilon 9}, \psi_\delta, \phi_{11}, \phi_{12}, \phi_{21}, \phi_{22}$$

the RMS is

```
0.36449025939002433
0.29799387184832743
0.19330057040630982
0.018580009880531635
0.011907937361162819
0.01746856125624562
0.013453470704039433
0.014691706237970526
0.006583119341598932
0.006026284376214681
0.0027087098026719447
0.022056288881352432
0.011510226396591989
0.0066381736709848985
0.008482623587508554
0.008064523554397036
0.006815466743207144
0.004096546961044399
0.011865104558665294
0.007718562040817101
0.009480850869618522
0.007888556221502835
0.01572230757428074
0.008062926769924586
0.008689046532359409
0.026379631232270252
0.03324528135748837
0.03588568521209713
0.012262550471119982
0.0013185427941095162
0.0062121170885075004
0.0025948778437549606
0.0061572108344055765
0.004625764078561339
0.0028927773905462824
0.004107521434185104
0.00300318721277933
0.004334048012140346
0.15252898453575475
0.055746124896860705
0.030818609167535434
0.030818609167535434
0.042208998225725616
```

We can see that most parameters have very smaller bias and RMS, except for some parameters related to the variance, such as ψ_δ. Although I want to improve the result, but there is no much time.

■

(e)

Solution. The prior inputs are

$$\begin{aligned} \boldsymbol{\mu}_0 &= \mathbf{0} \\ \alpha_{0\epsilon k} &= \alpha_{0\delta} = 9 \\ \beta_{0\epsilon k} &= \beta_{0\delta} = 4 \\ \rho_0 &= 4\,, \end{aligned}$$

and $A_{0k}, \mathbf{\Lambda}_{0k}, \mathbf{\Lambda}_{0\omega k}, \Phi_0$ are taken to be the true values.

To do sensitivity analysis, consider the following two perturbations.

$$\begin{aligned} &\text{Case 1: } \alpha_{0\epsilon k} = \alpha_{0\delta} = 3; \beta_{0\epsilon k} = \beta_{0\delta} = 10 \\ &\text{Case 2: } \alpha_{0\epsilon k} = \alpha_{0\delta} = 6; \beta_{0\epsilon k} = \beta_{0\delta} = 6 \end{aligned}$$

These two models are defined in the file “q1e1.stan” and “q1e2.stan” respectively.

Rerun the model, the plot the traceplots as follows.

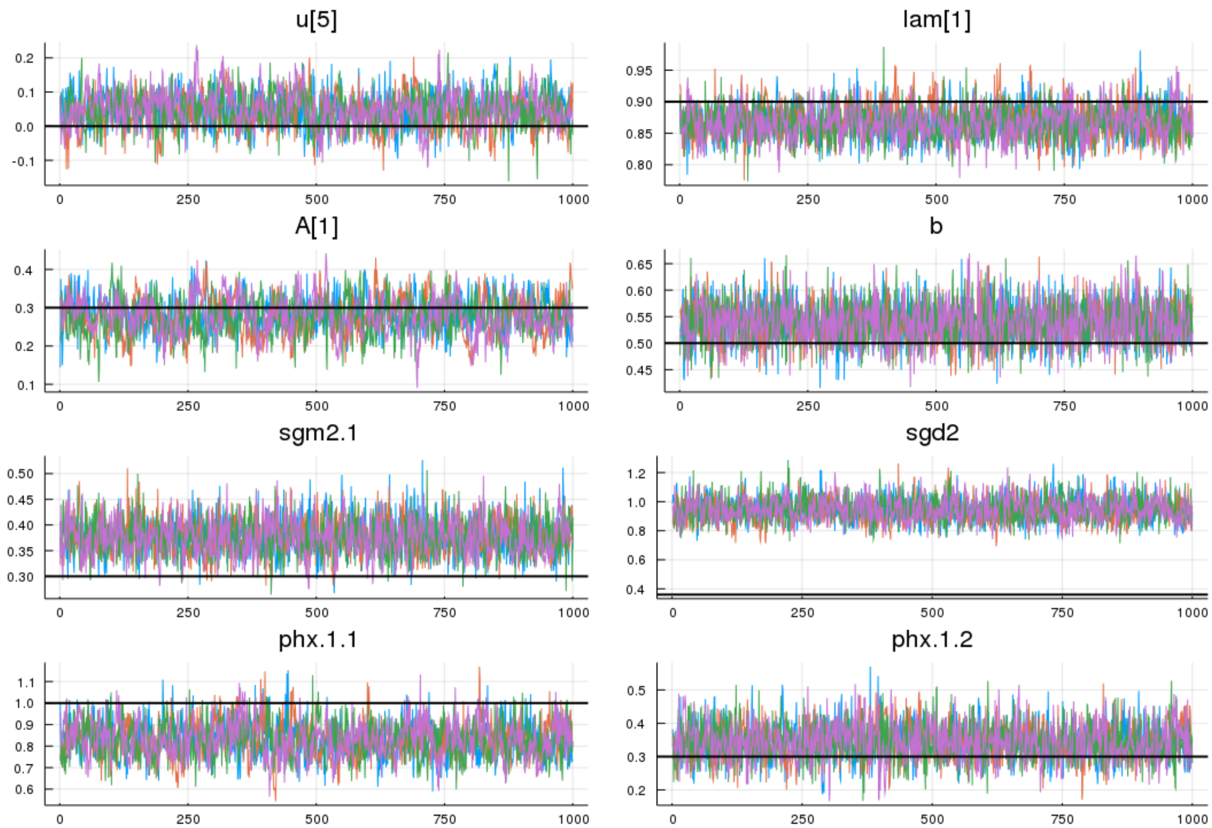


Figure 2: Perturbation case 1 of Figure 3.

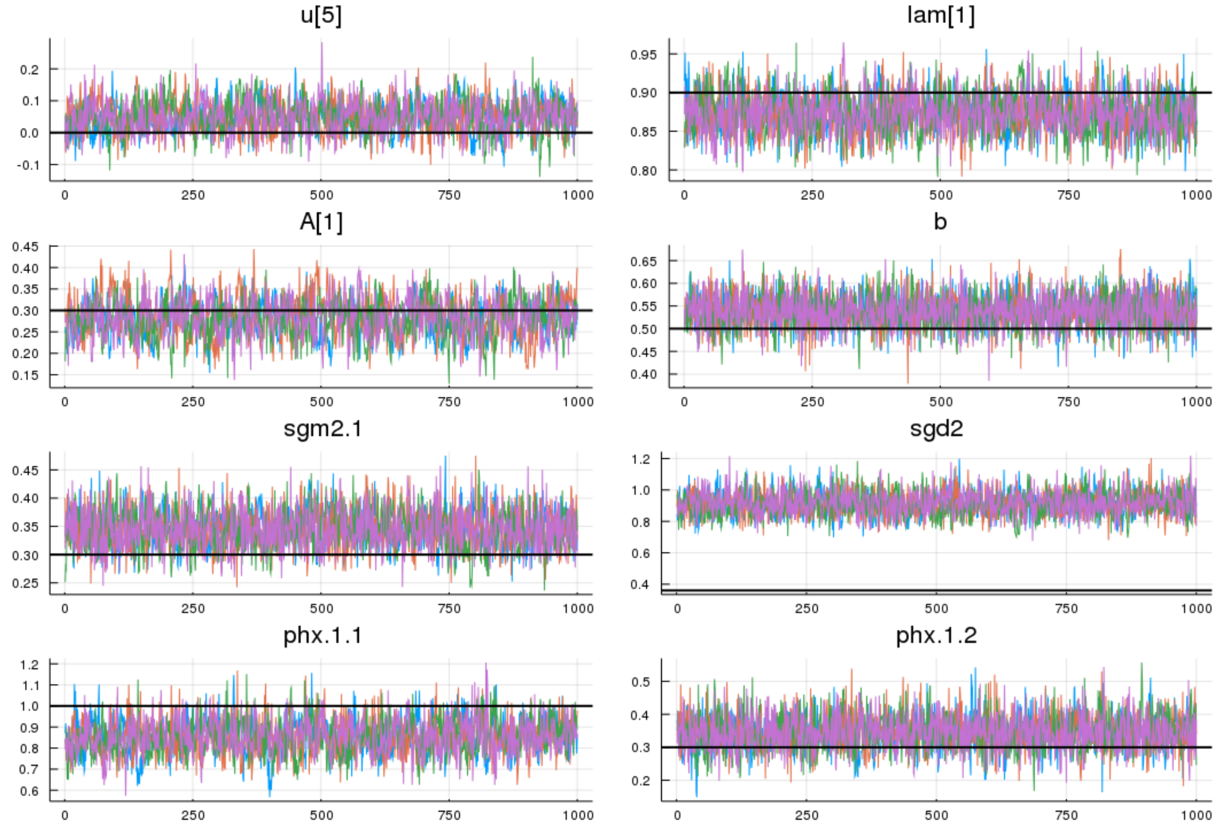


Figure 3: Perturbation case 2 of Figure 3.

These two traceplots with different perturbations are quite similar to the original Figure 3, so we can conclude that the Bayesian analysis is not sensitive to the inputs.

As you can see in the traceplots, ψ_δ is always underestimated and the error (absolute bias or RMS) is much higher. I have checked the code again and again, but I still failed to figure out the reason.

■

Q2

(a)

Solution. Let

$$M_t:\ \eta_i = bd_i + \gamma_1\xi_{1i} + \gamma_2\xi_{2i} + t(\gamma_3\xi_{1i}^2 + \gamma_4\xi_{2i}^2) + \delta_i\,,$$

then M_1 corresponds to the nonlinear SEM in Q1, while M_0 corresponds to its linear SEM counterpart.

The model is defined in the file "q2a.stan", which is almost same with "q1.stan", except for the calculation of "u[i]".

The $\widehat{\log B_{10}}$ over 10 replication are

```
-93.02818322896294
```

```
104.13306203024001
302.73710742961
1052.0785428888798
1222.865057605655
134.31117361605
811.4867376519248
254.41673631619997
139.01010735936998
-63.89951527270999
```

As you can see, only two replications prefer to M_0, the linear SEM, and thus we can conclude that the nonlinear SEM would perform better.

■

(b)

Solution. Let

$$M_t:\ \eta_i = bd_i + \gamma_1\xi_{1i} + \gamma_2\xi_{2i} + (1-t)\gamma_3\xi_{1i}\xi_{2i} + \gamma_4\xi_{1i}^2 + \gamma_5\xi_{2i}^2 + \delta_i\,,$$

then M_1 corresponds to the nonlinear SEM in Q1, while M_0 corresponds to the new SEM. The model is defined in the file "q2b.stan" (Appendix F), which is almost same with "q1.stan" (Appendix B and "q2a.stan" (Appendix E, except for a different calculation of "u[i]", and one more parameter γ.

The $\widehat{\log B_{10}}$ over 10 replication are

```
-225.983399744725
1392.6900717920098
899.072343060555
-790.0710668032499
-1203.73477001445
2531.6023784539
-317.10619694256997
-318.04820768020005
-262.44063651120007
1315.31390245448
```

Different from the previous question, now 6 replications prefer the more complex model, while only 4 replications prefer the "true" model. It seems incredible, but I think it is indeed natural because more complex model usually has a good fit, such as overfitting. To select a good model, it is necessary to investigate further by calculating other model comparison statistics.

■

A "sem.jl"

Note: Some greek symbols doesn't show properly.

```
using CmdStan
using Distributions

## ############################################
## parameter setup
## ############################################
b = 0.5
 = [0.4, 0.4, 0.3, 0.2]
_ = 0.36
_ = repeat([0.3, 0.5, 0.4], inner = 3)
 = [1.0 0.3; 0.3 1.0]
A = repeat([0.3, 0.7, 0.9], outer = 3)
 = [1 0 0;
    0.9 0 0;
    0.7 0 0;
    0 1 0;
    0 0.9 0;
    0 0.7 0;
    0 0 1;
    0 0 0.9;
    0 0 0.7]
df_d = 5
df_c = 9
u = zeros(9)

## ############################################
## simulate data
## ############################################
function genData(N::Int)
    c = ones(N)
    d = ones(N)
    Y = ones(N, 9)
    for i = 1:N
        c[i] = rand(TDist(df_c))
        d[i] = rand(TDist(df_d))
         = rand(MvNormal())
         = b*d[i] + [1]*[1] + [2]*[2] + [3]*[1]^2 + [4]*[2]^2 +
           rand(Normal(sqrt(_)))
        Y[i,:] = u + A * c[i] +  * vcat(, ) + rand(MvNormal(sqrt
           .(_)))
    end
    return c, d, Y
end
```

```
monitor = vcat("u.".*string.(1:9),
    "lam.".*string.(1:6),
    "A.".*string.(1:9),
    "gam.".*string.(1:4),
    "sgm2.".*string.(1:9),
    "phx.1.1", "phx.1.2", "phx.2.1", "phx.2.2",
    "sgd2",
    "b"
)


model = Stanmodel(name = "sem", model = read(open("q1.stan"),
   String), monitors = monitor)

## ##############################
## sensitivity analysis
## ##############################
modele1 = Stanmodel(name = "seme1", model = read(open("q1e1.stan
   "), String), monitors = monitor)
modele2 = Stanmodel(name = "seme2", model = read(open("q1e2.stan
   "), String), monitors = monitor)


N = 500
c, d, Y = genData(N)
data = Dict("N" => N,
            "c" => c,
            "d" => d,
            "Phi0" => ,
            "Y" => Y)

rc, sim, cnames = stan(model, data, summary = true)
rce1, sime1, cnamese1 = stan(modele1, data, summary = true)
rce2, sime2, cnamese2 = stan(modele2, data, summary = true)

## ############################################
## Trace plots
## ############################################

using Plots
function traceplot(res, idx::Int, lbl::String, truth)
    p = plot(res[:,:,1][:,idx], title = lbl, label="chain 1")
    for j = 2:4
        plot!(p, res[:,:,j][:,idx], label="chain "*string(j))
    end
```

```
    hline!(p, [truth], color=:black, lw=2, label="truth")
    return p
end

#p1 = traceplot(sim, 1, "u[1]", 0);
p1 = traceplot(sim, 5, "u[5]", 0);
p2 = traceplot(sim, 10, "lam[1]", 0.9);
p3 = traceplot(sim, 16, "A[1]", 0.3);
p4 = traceplot(sim, 25, "b", 0.5);
p5 = traceplot(sim, 30, "sgm2.1", 0.3);
p6 = traceplot(sim, 39, "sgd2", 0.36);
p7 = traceplot(sim, 40, "phx.1.1", 1.0);
p8 = traceplot(sim, 41, "phx.1.2", 0.3);
p = plot(p1, p2, p3, p4, p5, p6, p7, p8, layout = (4,2), legend=
   false, size = (1080, 720));
savefig(p, "traceplots.png")


## sensitivity analysis

p1 = traceplot(sime1, 5, "u[5]", 0);
p2 = traceplot(sime1, 10, "lam[1]", 0.9);
p3 = traceplot(sime1, 16, "A[1]", 0.3);
p4 = traceplot(sime1, 25, "b", 0.5);
p5 = traceplot(sime1, 30, "sgm2.1", 0.3);
p6 = traceplot(sime1, 39, "sgd2", 0.36);
p7 = traceplot(sime1, 40, "phx.1.1", 1.0);
p8 = traceplot(sime1, 41, "phx.1.2", 0.3);
p = plot(p1, p2, p3, p4, p5, p6, p7, p8, layout = (4,2), legend=
   false, size = (1080, 720));
savefig(p, "traceplots_e1.png")
p1 = traceplot(sime2, 5, "u[5]", 0);
p2 = traceplot(sime2, 10, "lam[1]", 0.9);
p3 = traceplot(sime2, 16, "A[1]", 0.3);
p4 = traceplot(sime2, 25, "b", 0.5);
p5 = traceplot(sime2, 30, "sgm2.1", 0.3);
p6 = traceplot(sime2, 39, "sgd2", 0.36);
p7 = traceplot(sime2, 40, "phx.1.1", 1.0);
p8 = traceplot(sime2, 41, "phx.1.2", 0.3);
p = plot(p1, p2, p3, p4, p5, p6, p7, p8, layout = (4,2), legend=
   false, size = (1080, 720));
savefig(p, "traceplots_e2.png")

=#
```

```
## ############################################
## calculate logBF
## ############################################
function logBF(model, data; nt::Int=10)
    U = ones(nt+1)
    for s = 1:(nt+1)
        data["ts"] = 1/nt*(s-1)
        rc, sim, cnames = stan(model, data, summary = false)
        U[s] = mean(sim[:,:,1])
    end
    res = 0
    for s = 1:nt
        res += ( U[s] + U[s+1] ) * (1 / nt) / 2
    end
    return res
end

q1_model = Stanmodel(name = "sem1", model = read(open("q1.stan")
   ,String), monitors = monitor, nchains=1)
q2a_model = Stanmodel(name = "sem2a", model = read(open("q2a.
   stan"),String), monitors = ["U"], nchains=1)
q2b_model = Stanmodel(name = "sem2b", model = read(open("q2b.
   stan"),String), monitors = ["U"], nchains=1)

# 10 replications
truth = vcat(u, repeat([0.9,0.7], outer=3), A, b, , _, _, [:])
N = 500

bf_a = ones(10)
bf_b = ones(10)
bias = ones(10,43)
rms = ones(10,43)
for i = 1:10
    println("i = ", i)
    c, d, Y = genData(N)
    data = Dict("N" => N,
            "c" => c,
            "d" => d,
            "Phi0" => ,
            "Y" => Y)
    rc, sim, cnames = stan(q1_model, data, summary = false)
    bias[i,:] = abs.(mean(sim[:,:,1], dims=1)' .- truth)
    rms[i,:] = (mean(sim[:,:,1], dims=1)' .- truth).^2
    bf_a[i] = logBF(q2a_model, data)
    bf_b[i] = logBF(q2b_model, data)
```

```
end
avg_bias = mean(bias, dims = 1)
avg_rms = mean(rms, dims = 1)
```

B “q1.stan”

```
data {
    int<lower=1> N; // number of observations
    matrix[N, 9] Y; // data matrix
    vector[N] c; // fixed covariate
    vector[N] d; // fixed covariate
    cov_matrix[2] Phi0; // covariance matrix of Inverse Wishart
       Dist.
}
transformed data {
    vector[2] zero = rep_vector(0, 2);
}
parameters {
    vector[9] u;
    vector[6] lam;
    vector[9] A;
    real b;
    vector[4] gam;
    vector<lower=0.0>[9] sgm2;
    real<lower=0.0> sgd2;
    cov_matrix[2] phx;
    vector[N] eta;
    matrix[N, 2] xi;
}
transformed parameters {
    matrix[N, 9] mu;
    vector[N] nu;
    vector[9] sgm = sqrt(sgm2);
    real sgd = sqrt(sgd2);
    for (i in 1:N){
        nu[i] = b * d[i] + gam[1] * xi[i, 1] + gam[2] * xi[i, 2]
            + gam[3] * xi[i, 1] * xi[i, 1] + gam[4] * xi[i, 2] *
            xi[i, 2];
        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];
        for (j in 1:9)
            mu[i, j] = mu[i, j] + A[j] * c[i];
    }
}
model {
    // hyper prior
    sgm2 ~ inv_gamma(9, 4);
    sgd2 ~ inv_gamma(9, 4);

    // prior
    A ~ multi_normal([0.3, 0.5, 0.4, 0.3, 0.5, 0.4, 0.3, 0.5,
       0.4], diag_matrix(sgm2));
    u ~ normal(0, 1);
    lam[1] ~ normal(0.9, sgm[2]);
    lam[2] ~ normal(0.7, sgm[3]);
    lam[3] ~ normal(0.9, sgm[5]);
    lam[4] ~ normal(0.7, sgm[6]);
    lam[5] ~ normal(0.9, sgm[8]);
    lam[6] ~ normal(0.7, sgm[9]);
    b ~ normal(0.5, sgd);
    gam[1] ~ normal(0.4, sgd);
    gam[2] ~ normal(0.4, sgd);
    gam[3] ~ normal(0.3, sgd);
    gam[4] ~ normal(0.2, sgd);
    phx ~ inv_wishart(4, Phi0);

    // structural equation
    for (i in 1:N){
        xi[i] ~ multi_normal(zero, phx);
        eta[i] ~ normal(nu[i], sgd);
    }

    // the likelihood
    for (i in 1:N){
        for (j in 1:9){
            Y[i, j] ~ normal(mu[i, j], sgm[j]);
        }
    }
}
```

C “q1e1.stan”

```
data {
    int<lower=1> N; // number of observations
    matrix[N, 9] Y; // data matrix
    vector[N] c; // fixed covariate
    vector[N] d; // fixed covariate
    cov_matrix[2] Phi0; // covariance matrix of Inverse Wishart
       Dist.
}
transformed data {
    vector[2] zero = rep_vector(0, 2);
}
parameters {
    vector[9] u;
    vector[6] lam;
    vector[9] A;
    real b;
    vector[4] gam;
    vector<lower=0.0>[9] sgm2;
    real<lower=0.0> sgd2;
    cov_matrix[2] phx;
    vector[N] eta;
    matrix[N, 2] xi;
}
transformed parameters {
    matrix[N, 9] mu;
    vector[N] nu;
    vector[9] sgm = sqrt(sgm2);
    real sgd = sqrt(sgd2);
    for (i in 1:N){
        nu[i] = b * d[i] + gam[1] * xi[i, 1] + gam[2] * xi[i, 2]
            + gam[3] * xi[i, 1] * xi[i, 1] + gam[4] * xi[i, 2] *
            xi[i, 2];
        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];
        for (j in 1:9)
            mu[i, j] = mu[i, j] + A[j] * c[i];
```

```
    }
}
model {
    // hyper prior
    sgm2 ~ inv_gamma(3, 10);
    sgd2 ~ inv_gamma(3, 10);

    // prior
    A ~ multi_normal([0.3, 0.5, 0.4, 0.3, 0.5, 0.4, 0.3, 0.5,
       0.4], diag_matrix(sgm2));
    u ~ normal(0, 1);
    lam[1] ~ normal(0.9, sgm[2]);
    lam[2] ~ normal(0.7, sgm[3]);
    lam[3] ~ normal(0.9, sgm[5]);
    lam[4] ~ normal(0.7, sgm[6]);
    lam[5] ~ normal(0.9, sgm[8]);
    lam[6] ~ normal(0.7, sgm[9]);
    b ~ normal(0.5, sgd);
    gam[1] ~ normal(0.4, sgd);
    gam[2] ~ normal(0.4, sgd);
    gam[3] ~ normal(0.3, sgd);
    gam[4] ~ normal(0.2, sgd);
    phx ~ inv_wishart(4, Phi0);

    // structural equation
    for (i in 1:N){
        xi[i] ~ multi_normal(zero, phx);
        eta[i] ~ normal(nu[i], sgd);
    }

    // the likelihood
    for (i in 1:N){
        for (j in 1:9){
            Y[i, j] ~ normal(mu[i, j], sgm[j]);
        }
    }
}
```

D "q1e2.stan"

```
data {
    int<lower=1> N; // number of observations
    matrix[N, 9] Y; // data matrix
```

```
    vector[N] c; // fixed covariate
    vector[N] d; // fixed covariate
    cov_matrix[2] Phi0; // covariance matrix of Inverse Wishart
       Dist.
}
transformed data {
    vector[2] zero = rep_vector(0, 2);
}
parameters {
    vector[9] u;
    vector[6] lam;
    vector[9] A;
    real b;
    vector[4] gam;
    vector<lower=0.0>[9] sgm2;
    real<lower=0.0> sgd2;
    cov_matrix[2] phx;
    vector[N] eta;
    matrix[N, 2] xi;
}
transformed parameters {
    matrix[N, 9] mu;
    vector[N] nu;
    vector[9] sgm = sqrt(sgm2);
    real sgd = sqrt(sgd2);
    for (i in 1:N){
        nu[i] = b * d[i] + gam[1] * xi[i, 1] + gam[2] * xi[i, 2]
            + gam[3] * xi[i, 1] * xi[i, 1] + gam[4] * xi[i, 2] *
            xi[i, 2];
        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];
        for (j in 1:9)
            mu[i, j] = mu[i, j] + A[j] * c[i];
    }
}
model {
    // hyper prior
    sgm2 ~ inv_gamma(6, 6);
```

```
    sgd2 ~ inv_gamma(6, 6);

    // prior
    A ~ multi_normal([0.3, 0.5, 0.4, 0.3, 0.5, 0.4, 0.3, 0.5,
       0.4], diag_matrix(sgm2));
    u ~ normal(0, 1);
    lam[1] ~ normal(0.9, sgm[2]);
    lam[2] ~ normal(0.7, sgm[3]);
    lam[3] ~ normal(0.9, sgm[5]);
    lam[4] ~ normal(0.7, sgm[6]);
    lam[5] ~ normal(0.9, sgm[8]);
    lam[6] ~ normal(0.7, sgm[9]);
    b ~ normal(0.5, sgd);
    gam[1] ~ normal(0.4, sgd);
    gam[2] ~ normal(0.4, sgd);
    gam[3] ~ normal(0.3, sgd);
    gam[4] ~ normal(0.2, sgd);
    phx ~ inv_wishart(4, Phi0);

    // structural equation
    for (i in 1:N){
        xi[i] ~ multi_normal(zero, phx);
        eta[i] ~ normal(nu[i], sgd);
    }

    // the likelihood
    for (i in 1:N){
        for (j in 1:9){
            Y[i, j] ~ normal(mu[i, j], sgm[j]);
        }
    }
}
```

E “q2a.stan”

```
data {
    int<lower=1> N; // number of observations
    matrix[N, 9] Y; // data matrix
    vector[N] c; // fixed covariate
    vector[N] d; // fixed covariate
    cov_matrix[2] Phi0; // covariance matrix of Inverse Wishart
       Dist.
    real ts;
```

```
}
parameters {
    vector[9] u;
    vector[6] lam;
    vector[9] A;
    real b;
    vector[4] gam;
    vector<lower=0.0>[9] sgm2;
    real<lower=0.0> sgd2;
    cov_matrix[2] phx;
    vector[N] eta;
    matrix[N, 2] xi;
}
transformed parameters {
    matrix[N, 9] mu;
    vector[N] nu;
    vector[2] zero = rep_vector(0, 2);
    vector[9] sgm = sqrt(sgm2);
    real sgd = sqrt(sgd2);
    for (i in 1:N){
        nu[i] = b * d[i] + gam[1] * xi[i, 1] + gam[2] * xi[i, 2]
            + ts * (gam[3] * xi[i, 1] * xi[i, 1] + gam[4] * xi[i
            , 2] * xi[i, 2]);
        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];
        for (j in 1:9)
            mu[i, j] = mu[i, j] + A[j] * c[i];
    }
}
model {
    // hyper prior
    sgm2 ~ inv_gamma(9, 4);
    sgd2 ~ inv_gamma(9, 4);

    // prior
    A ~ multi_normal([0.3, 0.5, 0.4, 0.3, 0.5, 0.4, 0.3, 0.5,
       0.4], diag_matrix(sgm2));
    u ~ normal(0, 1);
```

```
    lam[1] ~ normal(0.9, sgm[2]);
    lam[2] ~ normal(0.7, sgm[3]);
    lam[3] ~ normal(0.9, sgm[5]);
    lam[4] ~ normal(0.7, sgm[6]);
    lam[5] ~ normal(0.9, sgm[8]);
    lam[6] ~ normal(0.7, sgm[9]);
    b ~ normal(0.5, sgd);
    gam[1] ~ normal(0.4, sgd);
    gam[2] ~ normal(0.4, sgd);
    gam[3] ~ normal(0.3, sgd);
    gam[4] ~ normal(0.2, sgd);
    phx ~ inv_wishart(4, Phi0);

    // structural equation
    for (i in 1:N){
        xi[i] ~ multi_normal(zero, phx);
        eta[i] ~ normal(nu[i], sgd);
    }

    // the likelihood
    for (i in 1:N){
        for (j in 1:9){
            Y[i, j] ~ normal(mu[i, j], sgm[j]);
        }
    }
}
generated quantities {
    real U = 0;
    for (i in 1:N){
        U -= ( eta[i] - b*d[i] - gam[1]*xi[i,1] - gam[2]*xi[i,2]
                - ts*(gam[3]*xi[i,1]*xi[i,1] + gam[4]*xi[i,2]*xi
                   [i,2]) ) *
             (-1)*(gam[3]*xi[i,1]*xi[i,1]+gam[4]*xi[i,2]*xi[i
                ,2]) / sgd;
    }
}
```

F “q2b.stan”

```
data {
    int<lower=1> N; // number of observations
    matrix[N, 9] Y; // data matrix
    vector[N] c; // fixed covariate
```

```
    vector[N] d; // fixed covariate
    cov_matrix[2] Phi0; // covariance matrix of Inverse Wishart
       Dist.
    real ts;
}
parameters {
    vector[9] u;
    vector[6] lam;
    vector[9] A;
    real b;
    vector[5] gam;
    vector<lower=0.0>[9] sgm2;
    real<lower=0.0> sgd2;
    cov_matrix[2] phx;
    vector[N] eta;
    matrix[N, 2] xi;
}
transformed parameters {
    matrix[N, 9] mu;
    vector[N] nu;
    vector[2] zero = rep_vector(0, 2);
    vector[9] sgm = sqrt(sgm2);
    real sgd = sqrt(sgd2);
    for (i in 1:N){
        nu[i] = b * d[i] + gam[1] * xi[i, 1] + gam[2] * xi[i, 2]
            + (1 - ts) * gam[5]*xi[i, 1]*xi[i,2] + gam[3] * xi[i
           , 1] * xi[i, 1] + gam[4] * xi[i, 2] * xi[i, 2];
        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];
        for (j in 1:9)
            mu[i, j] = mu[i, j] + A[j] * c[i];
    }
}
model {
    // hyper prior
    sgm2 ~ inv_gamma(9, 4);
    sgd2 ~ inv_gamma(9, 4);
```

```
    // prior
    A ~ multi_normal([0.3, 0.5, 0.4, 0.3, 0.5, 0.4, 0.3, 0.5,
       0.4], diag_matrix(sgm2));
    u ~ normal(0, 1);
    lam[1] ~ normal(0.9, sgm[2]);
    lam[2] ~ normal(0.7, sgm[3]);
    lam[3] ~ normal(0.9, sgm[5]);
    lam[4] ~ normal(0.7, sgm[6]);
    lam[5] ~ normal(0.9, sgm[8]);
    lam[6] ~ normal(0.7, sgm[9]);
    b ~ normal(0.5, sgd);
    gam[1] ~ normal(0.4, sgd);
    gam[2] ~ normal(0.4, sgd);
    gam[3] ~ normal(0.3, sgd);
    gam[4] ~ normal(0.2, sgd);
    gam[5] ~ normal(0.5, sgd);
    phx ~ inv_wishart(4, Phi0);

    // structural equation
    for (i in 1:N){
        xi[i] ~ multi_normal(zero, phx);
        eta[i] ~ normal(nu[i], sgd);
    }

    // the likelihood
    for (i in 1:N){
        for (j in 1:9){
            Y[i, j] ~ normal(mu[i, j], sgm[j]);
        }
    }
}
generated quantities {
    real U = 0;
    for (i in 1:N){
        U -= ( eta[i] - b*d[i] - gam[1]*xi[i,1] - gam[2]*xi[i,2]
               - (1-ts)*gam[5]*xi[i,1]*xi[i,2] ) *
             (gam[5]*xi[i,1]*xi[i,2]) / sgd;
    }
}
```