Economics of Networks

Exercises

Solutions (in pdf format) should be submitted to m.d.konig@vu.nl.

Setup

1. Following the lecture, we want to estimate the SAR model:[1]

$$\mathbf{Y} = \lambda \mathbf{A}\mathbf{Y} + \rho \mathbf{Y}_{tot} + X\beta + \boldsymbol{\varepsilon}, \tag{1}$$

where $\mathbf{Y}$ is the output of the firms, $\mathbf{AY}$ is the collaboration partners' (neighbors') output, $\mathbf{Y}_{tot}$ is the total output of all firms in the same market, X is the productivity of the firms and $\boldsymbol{\varepsilon}$ is an error term.

2. Load the data into Matlab.

```
load('./Data/A.mat') % adjacency matrix
load('./Data/ID.mat') % firm ids
load('./Data/dm.mat') % missing data indicator
load('./Data/X1.mat') % firm covariates (productivity)
load('./Data/X2.mat') % total output
load('./Data/X3.mat') % neighbors' output
load('./Data/NAT.mat') % geographic locations
```

3. Adjust years.

```
%% drop the first t0 years (i.e. the starting year is (yr1+t0))
yr1 = 1966;
yr2 = 2006;
t0 = 1;
T = yr2-yr1+1;
dt = cat(1,dm{:});
dt = dt(n*t0+1:end);
T1 = T-t0;
```

4. Drop firms that appear less than twice.

```
di = (1:n*T1)';
```

[1] See also König et al. (2018).

```
dj = kron(ones(T1,1),(1:n)');
Dn = sparse(di,dj,dt); % number of rows = n*T, number of columns = n.
dd = sum(Dn,1); % how many times each firm appears in the data.
Dn = Dn(:,dd>1); % firms appearing more than once in the panel.
np = size(Dn,2); % number of remaining firms.
E  = speye(n);
E  = E(dd>1,:); % firms appearing more than once in the panel
ID = E*ID; % first column is the firm id, second column is the sector (SIC) code
for s = 1:T
    dm{s} = E*dm{s};
    X1{s} = E*X1{s};
    X2{s} = E*X2{s};
    X3{s} = E*X3{s};
end
```

5. Construct the data matrices.

```
di  = cell(T1,1);
dj2 = cell(T1,1);
dj3 = cell(T1,1);
n1 = 0;
n2 = 0;
for s = 1:T1
    D = Dn(n*(s-1)+1:n*s,:);
    D = D(sum(D,2)==1,:); % firms without missing observations
    [ni,nj] = size(D); % note: ni is the number of nonzero entries of D
    [ii,jj,¬] = find(D);
    di{s}  = n1+ii;
    dj2{s} = s*ones(ni,1);
    dj3{s} = n2+jj;

    n1 = n1+ni;
    n2 = n2+nj;
end
di  = cat(1,di{:});
dj2 = cat(1,dj2{:});
dj3 = cat(1,dj3{:});

D2 = sparse(di,dj2,ones(n1,1),n1,T1); % block diagonal matrix of D*1
D3 = sparse(di,dj3,ones(n1,1),n1,n2); % block diagonal matrix of D
L1 = D2(:,2:end); % time dummies
DD2 = (D2'*D2)\D2';
DD2 = speye(n1)-D2*DD2;
PD2 = DD2*D3;
Yn = cell(T1,1);
Zn = cell(T1,1);
Qn = cell(T1,1);
for s = (t0+1):T
    Yn{s-t0} = X1{s}(:,1);
    Zn{s-t0} = [X3{s}(:,1),X2{s}(:,1),X1{s}(:,2)];
    Qn{s-t0} = [X3{s}(:,2),X2{s}(:,2),X1{s}(:,2)];
end
Yn = cat(1,Yn{:});
Zn = cat(1,Zn{:});
Qn = cat(1,Qn{:});
```

```
nk = size(Zn,2);
```

Exercise 1: SAR model with time (fixed) effects.

1. We construct the estimator and standard errors as follows:

```
Y2 = PD2*Yn;
Z2 = PD2*Zn;
Q2 = PD2*Qn;
QQ = Q2'*Q2;
QZ = Q2'*Z2;
QY = Q2'*Y2;
PI = QQ\QZ;
ZZ = (QZ'*PI)\PI'; % See page 15 in the lecture notes: (QZ'*QQ\QZ)\(QQ\QZ)', ...
    Q = H, QQ = H'H, b = (QZ'*H'H\QZ)\(H'H\QZ)', written in compact form to ...
    save memory.
b = ZZ*QY; % See the optimal GMM estimator on page 15 in the lecture notes.
% robust s.e.
u2 = Y2-Z2*b;
V2 = Q2'*sparse(1:n1,1:n1,u2.^2)*Q2;
se = sqrt(spdiags(ZZ*V2*ZZ',0));
```

   Show how the above equations can be derived from the theory of SAR models discussed in the lecture.

2. Print the estimation results to a file.

```
fid = fopen(['./output1.txt'],'w');
coeff = {'lambda' 'rho    ' 'beta   ' };
for ip = 1:nk
    tstat = b(ip)/se(ip);
    tstat = abs(tstat);
    if tstat ≥ 2.326
        fprintf(fid,'%s: %7.4f*** (%6.4f)\n',coeff{ip},b(ip),se(ip));
    elseif tstat ≥ 1.96
        fprintf(fid,'%s: %7.4f** (%6.4f)\n',coeff{ip},b(ip),se(ip));
    elseif tstat ≥ 1.645
        fprintf(fid,'%s: %7.4f* (%6.4f)\n',coeff{ip},b(ip),se(ip));
    else
        fprintf(fid,'%s: %7.4f (%6.4f)\n',coeff{ip},b(ip),se(ip));
    end
end
fclose(fid);
```

   Discuss the estimation results.

Exercise 2: SAR model with firm fixed effects.

1. We construct the estimator and standard errors as follows:

```
Y1 = PD1*Yn;
Z1 = PD1*Zn;
Q1 = PD1*Qn;
QQ = Q1'*Q1;
QZ = Q1'*Z1;
QY = Q1'*Y1;
PI = QQ\QZ;
ZZ = (QZ'*PI)\PI';
b = ZZ*QY;
% robust s.e.
u1 = Y1-Z1*b;
V1 = Q1'*sparse(1:n1,1:n1,u1.^2)*Q1;
se = sqrt(spdiags(ZZ*V1*ZZ',0));
```

Show how the above equations can be derived from the theory of SAR models discussed in the lecture.

2. Print the estimation results to a file.

```
fid = fopen(['./output2.txt'],'w');
coeff = {'lambda' 'rho    ' 'beta    ' };
for ip = 1:nk
    tstat = b(ip)/se(ip);
    tstat = abs(tstat);
    if tstat >= 2.326
        fprintf(fid,'%s: %7.4f*** (%6.4f)\n',coeff{ip},b(ip),se(ip));
    elseif tstat >= 1.96
        fprintf(fid,'%s: %7.4f** (%6.4f)\n',coeff{ip},b(ip),se(ip));
    elseif tstat >= 1.645
        fprintf(fid,'%s: %7.4f* (%6.4f)\n',coeff{ip},b(ip),se(ip));
    else
        fprintf(fid,'%s: %7.4f (%6.4f)\n',coeff{ip},b(ip),se(ip));
    end
end
fclose(fid);
```

Discuss the estimation results.

Exercise 3: SAR model with both, firm and time fixed effects.

1. We construct the estimator and standard errors as follows:

```
Y3 = PD3*Yn;
Z3 = PD3*Zn;
Q3 = PD3*Qn;
QQ = Q3'*Q3;
QZ = Q3'*Z3;
QY = Q3'*Y3;
PI = QQ\QZ;
ZZ = (QZ'*PI)\PI';
b = ZZ*QY;
% robust s.e.
u3 = Y3-Z3*b;
V3 = Q3'*sparse(1:n1,1:n1,u3.^2)*Q3;
se = sqrt(spdiags(ZZ*V3*ZZ',0));
```

   Show how the above equations can be derived from the theory of SAR models discussed in the lecture.

2. Print the estimation results to a file.

```
fid = fopen(['./output3.txt'],'w');
coeff = {'lambda' 'rho    ' 'beta    ' };
for ip = 1:nk
    tstat = b(ip)/se(ip);
    tstat = abs(tstat);
    if tstat ≥ 2.326
        fprintf(fid,'%s: %7.4f*** (%6.4f)\n',coeff{ip},b(ip),se(ip));
    elseif tstat ≥ 1.96
        fprintf(fid,'%s: %7.4f** (%6.4f)\n',coeff{ip},b(ip),se(ip));
    elseif tstat ≥ 1.645
        fprintf(fid,'%s: %7.4f* (%6.4f)\n',coeff{ip},b(ip),se(ip));
    else
        fprintf(fid,'%s: %7.4f (%6.4f)\n',coeff{ip},b(ip),se(ip));
    end
end
fclose(fid);
```

   Discuss the estimation results.

Exercise 4: Logistic regression.

1. We repeat the estimation Exercises 1 to 3, but use predicted links (instead of the observed links) to construct IVs. For the logistic regression we use the following Matlab function `logit_obj.m`:

```
function [f,G,H] = logit_obj(b,Y,X)
% f: function value
% g: gradient
% H: hessian

n = length(Y);

u = exp(X*b);
P = u./(1+u);

f = Y.*log(P)+(1-Y).*log(1-P);
f = -sum(f);

g = X'*(Y-P);
g = -g;

H = -X'*sparse(1:n,1:n,P.*(1-P))*X;
H = -H;
```

2. Load adjacency matrices and patent proximity matrices.

```
ID3 = floor(ID(:,2)/100);
grp = bsxfun(@eq,ID3,ID3');
grp(triu(true(size(grp))))=0;
[gi,gj,~] = find(grp);
nl = length(gi);
T2 = yr2-yr0+1;
Wy = zeros(nl,T2);
Wx = zeros(nl,T2);
Wa = zeros(nl,T2);
Wb = zeros(nl,T2);
W1 = zeros(nl,T2);
W2 = zeros(nl,T2);
for s = 1:T2
    yr = yr0+s-1;
    load(['./Data/A_' int2str(yr) '.mat']) % Load adjacency matrix.
    B = double(A^2>0)-double(A>0); % Second order neighbors
    load(['./Data/P_' int2str(yr) '.mat']) % Load technology proximity matrix.
    Wy(:,s) = A(grp==1);
    Wx(:,s) = B(grp==1);
    Wa(:,s) = sum(Wy(:,1:s),2);
    Wb(:,s) = sum(Wx(:,1:s),2);
    W1(:,s) = P(grp==1);
end
```

3. Adjust location data and compute geographic distances.

```
tmp1 = NAT(:,1);
tmp2 = NAT(:,2);
tmp3 = NAT(:,3);
loc = zeros(np,2);
for i = 1:np
    if sum(tmp1==ID(i,1)) == 1
        loc(i,1) = tmp2(tmp1==ID(i,1));
        loc(i,2) = tmp3(tmp1==ID(i,1));
    end
end
dis = pdist2(loc,loc); % Pairwise distance between two sets of observations
prx = zeros(np);
prx(dis==0) = 1;
dc = prx(grp==1);
dc = kron(ones(T1,1),dc);
```

4. Construct data matrices for logistic regression.

```
n0 = nl*T1;
dy = reshape(Wy(:,T2-T1+1:T2),n0,1);
da = reshape(Wa(:,T2-T1-lnkyear+1:T2-lnkyear),n0,1);
db = reshape(Wb(:,T2-T1-lnkyear+1:T2-lnkyear),n0,1);
dp = reshape(W1(:,T2-T1-lnkyear+1:T2-lnkyear),n0,1);
dx = [da,db,dp,dp.^2,dc,ones(n0,1)];
kx = size(dx,2);
```

5. Estimate logistic regression parameters.

```
xx = dx'*dx;
b0 = xx\(dx'*dy);
option = optimoptions(@fminunc,'Algorithm','trust-region','GradObj', ...
    'on','Hessian','on','Display','notify','DerivativeCheck','off');
[b0,FVAL,EXITFLAG,OUTPUT,GRAD,HESSIAN] = fminunc('logit_obj',b0,option,dy,dx);
s0 = sqrt(diag(HESSIAN\speye(kx)))';

p0 = exp(dx*b0)./(1+exp(dx*b0));
d0 = 0;
for i = 1:n0
    d0 = d0+dy(i)*log(p0(i))+(1-dy(i))*log(1-p0(i));
end
y0 = mean(dy);
d1 = n0*(y0*log(y0)+(1-y0)*log(1-y0));
R2 = 1-d0/d1; % McFadden's R-squared (Cameron and Trivedi, p.474)
%http://www.ats.ucla.edu/stat/mult_pkg/faq/general/Psuedo_RSquareds.htm
```

6. Construct predicted adjacency matrix.

```
for s = (t0+1):T
    Ap = sparse(gi,gj,p0(nl*(s-t0-1)+1:nl*(s-t0)),np,np);
    Ap = Ap+Ap';
```

```
    X3{s}(:,2) = Ap*X1{s}(:,2);
end
```

7. Construct data matrices for SAR models.

```
Yn = cell(T1,1);
Zn = cell(T1,1);
Qn = cell(T1,1);
for s = (t0+1):T
    Yn{s-t0} = X1{s}(:,1);
    Zn{s-t0} = [X3{s}(:,1),X2{s}(:,1),X1{s}(:,2)];
    Qn{s-t0} = [X3{s}(:,2),X2{s}(:,2),X1{s}(:,2)];
end
Yn = cat(1,Yn{:});
Zn = cat(1,Zn{:});
Qn = cat(1,Qn{:});
nk = size(Zn,2);
```

8. 2SLS estimation of SAR model with firm fixed effects.

```
Y1 = PD1*Yn;
Z1 = PD1*Zn;
Q1 = PD1*Qn;
QQ = Q1'*Q1;
QZ = Q1'*Z1;
QY = Q1'*Y1;
PI = QQ\QZ;
ZZ = (QZ'*PI)\PI';
b1 = ZZ*QY;
% robust s.e.
u1 = Y1-Z1*b1;
V1 = Q1'*sparse(1:n1,1:n1,u1.^2)*Q1;
s1 = sqrt(spdiags(ZZ*V1*ZZ',0));
```

9. 2SLS estimation of SAR model with time (fixed) effects.

```
Y2 = PD2*Yn;
Z2 = PD2*Zn;
Q2 = PD2*Qn;
QQ = Q2'*Q2;
QZ = Q2'*Z2;
QY = Q2'*Y2;
PI = QQ\QZ;
ZZ = (QZ'*PI)\PI';
b2 = ZZ*QY;
% robust s.e.
u2 = Y2-Z2*b2;
V2 = Q2'*sparse(1:n1,1:n1,u2.^2)*Q2;
s2 = sqrt(spdiags(ZZ*V2*ZZ',0));
```

10. 2SLS estimation of SAR model with firm and time fixed effects.

```
Y3 = PD3*Yn;
Z3 = PD3*Zn;
Q3 = PD3*Qn;
QQ = Q3'*Q3;
QZ = Q3'*Z3;
QY = Q3'*Y3;
PI = QQ\QZ;
ZZ = (QZ'*PI)\PI';
b3 = ZZ*QY;
% robust s.e.
u3 = Y3-Z3*b3;
V3 = Q3'*sparse(1:n1,1:n1,u3.^2)*Q3;
s3 = sqrt(spdiags(ZZ*V3*ZZ',0));
```

11. Print results.

```
fid = fopen(['./output4.txt'],'w');
fprintf(fid,'Logistic regression \n');
coeff = {'da' 'db' 'pat' 'pat.^2' 'dc' 'const' };
for ip = 1:kx
    tstat = b0(ip)/s0(ip);
    tstat = abs(tstat);
    if tstat >= 2.326
        fprintf(fid,'%s: %7.4f*** (%6.4f)\n',coeff{ip},b0(ip),s0(ip));
    elseif tstat >= 1.96
        fprintf(fid,'%s: %7.4f** (%6.4f)\n',coeff{ip},b0(ip),s0(ip));
    elseif tstat >= 1.645
        fprintf(fid,'%s: %7.4f* (%6.4f)\n',coeff{ip},b0(ip),s0(ip));
    else
        fprintf(fid,'%s: %7.4f (%6.4f)\n',coeff{ip},b0(ip),s0(ip));
    end
end
fprintf(fid,'Link prediction R^2 is %7.4f \n',R2);
fprintf(fid,'\n');
fprintf(fid,'SAR with firm fixed effects \n');
coeff = {'lambda' 'rho   ' 'beta   ' };
for ip = 1:nk
    tstat = b1(ip)/s1(ip);
    tstat = abs(tstat);
    if tstat >= 2.326
        fprintf(fid,'%s: %7.4f*** (%6.4f)\n',coeff{ip},b1(ip),s1(ip));
    elseif tstat >= 1.96
        fprintf(fid,'%s: %7.4f** (%6.4f)\n',coeff{ip},b1(ip),s1(ip));
    elseif tstat >= 1.645
        fprintf(fid,'%s: %7.4f* (%6.4f)\n',coeff{ip},b1(ip),s1(ip));
    else
        fprintf(fid,'%s: %7.4f (%6.4f)\n',coeff{ip},b1(ip),s1(ip));
    end
end
fprintf(fid,'\n');
fprintf(fid,'SAR with time fixed effects \n');
for ip = 1:nk
```

```
    tstat = b2(ip)/s2(ip);
    tstat = abs(tstat);
    if tstat ≥ 2.326
        fprintf(fid,'%s:  %7.4f*** (%6.4f)\n',coeff{ip},b2(ip),s2(ip));
    elseif tstat ≥ 1.96
        fprintf(fid,'%s:  %7.4f** (%6.4f)\n',coeff{ip},b2(ip),s2(ip));
    elseif tstat ≥ 1.645
        fprintf(fid,'%s:  %7.4f* (%6.4f)\n',coeff{ip},b2(ip),s2(ip));
    else
        fprintf(fid,'%s:  %7.4f (%6.4f)\n',coeff{ip},b2(ip),s2(ip));
    end
end
fprintf(fid,'\n');
fprintf(fid,'SAR with firm and time fixed effects \n');
for ip = 1:nk
    tstat = b3(ip)/s3(ip);
    tstat = abs(tstat);
    if tstat ≥ 2.326
        fprintf(fid,'%s:  %7.4f*** (%6.4f)\n',coeff{ip},b3(ip),s3(ip));
    elseif tstat ≥ 1.96
        fprintf(fid,'%s:  %7.4f** (%6.4f)\n',coeff{ip},b3(ip),s3(ip));
    elseif tstat ≥ 1.645
        fprintf(fid,'%s:  %7.4f* (%6.4f)\n',coeff{ip},b3(ip),s3(ip));
    else
        fprintf(fid,'%s:  %7.4f (%6.4f)\n',coeff{ip},b3(ip),s3(ip));
    end
end
fclose(fid);
```

12. Discuss the estimation results.

Exercise 5: DMH algorithm.

1. As shown in the lecture, the joint network formation and effort adjustment process converges to a unique stationary distribution characterized by the Gibbs measure

$$\pi(G, Y|\theta) = c(\theta)^{-1} \exp[\sigma^{-2}\Phi(G, Y|\gamma)], \tag{2}$$

where $c(\theta) = \sum_{G \in \mathcal{G}(n)} \int_{\mathcal{Y}^n} \exp[\sigma^{-2}\Phi(G, Y|\gamma)]dY$. Given an observation (G, Y) from the stationary distribution defined in Equation (2), we can estimate the parameter vector θ using the DMH algorithm discussed in the lecture. Further details of the algorithm can be found in Appendix C.[2]

More specifically, in this exercise we want to estimate the spillover parameter λ, parameters in the marginal cost of production $\beta = (\beta_0, \beta_1^\top, \beta_2)^\top$ (with the dimension denoted by K), parameters in the collaboration cost $\delta = (\delta_0, \delta_1^\top, \delta_2, \delta_3, \delta_4)^\top$ (with the dimension denoted by S), and the noise parameter σ^2. These parameters are denoted by $\theta = (\lambda, \beta^\top, \delta^\top, \sigma^2)^\top$. We assign the prior distributions of model parameters and unknown variables as follows:

(i) Spillover effect parameter: $\lambda \sim U(-\|A\|_\infty^{-1}, \|A\|_\infty^{-1})$.

(ii) Parameters in the marginal cost of production: $\beta \sim N(\mu_\beta, \varsigma_\beta^2 I_K)$.

(iii) Parameters in the collaboration cost: $\delta \sim N(\mu_\delta, \varsigma_\delta^2 I_S)$.

(iv) Noise parameter: $\sigma^2 \sim N_{[0,\infty)}(\mu_\sigma, \varsigma_\sigma^2)$.

The above prior distributions are conjugate priors commonly used in the Bayesian literature. The spillover effect parameter λ shares similar properties as the spatial lag parameter in the spatial econometrics literature and we use a uniform prior for λ following Smith and LeSage (2004) and assume $\lambda \in (-\|A\|_\infty^{-1}, \|A\|_\infty^{-1})$ to guarantee that the best response function has a unique equilibrium. Finally, to guarantee that σ^2 is non-negative, we assume it follows a truncated normal distribution on $[0, \infty)$. We also assume independence across prior distributions of parameters and latent variables. We set $\mu_\beta = 0$, $\mu_\delta = 0$, $\mu_\sigma = 0$, $\varsigma_\beta^2 = \varsigma_\delta^2 = \varsigma_\sigma^2 = 100$, $\kappa = 1$ and $\alpha = 2$ to ensure our prior distributions cover a wide range of parameter spaces and thus be uninformative in our empirical analysis.

The Matlab code for this algorithm can be found below. The data for this exercise can be found in the file `./Data/data.mat`.

```
clear;
load ./Data/data;

L=2;      % Number of Monte Carlo repetitions.
N=100;    % Number of nodes in the network.
T=10000; % Number of iterations of the MCMC algorithm.
R=2;      % Number of iterations for simulating the network.

gamma_T=zeros(7,T);
lambda_T=zeros(T,1);
beta_T=zeros(T,1);
```

[2]See also Hsieh et al. (2018).

```

for  l=1:L % Monte Carlo repetitions.

    W=WW{l}; % W is the network matrix.
    C=CC{l}; % C is the matrix of exogenous dyadic variables.
    X=XX{l}; % X is the vector of exogenous individual variables for the ...
        outcome equation.
    Y=YY{l}; % Y is the vector of outcome variables.

    %% Jumping rate in the proposal distributions.
    c_1=1e-5;
    c_2=1e-4;
    c_3=1e-2;
    acc_1=0.0;
    acc_rate1=zeros(T,1);

    %% Initial values to start MCMC.
    gamma_T(:,1)=[-3.0, 1.0, -0.10, 0.3, -0.03, 0.5, 0.60 ];
    lambda_T(1)=0.0100;
    beta_T(1)= 0.8000;

    %% Hyper parameters.
    beta_0=0;
    gamma_0=zeros(1,7);
    lambda_0=0.0;
    G_0=eye(7)*100.0;
    B_0=100.0;

    for  t=2:T  % Start the MCMC algorithm.
        tic;

        %% Propose gamma by adaptive M-H following Haario, H., Saksman, E., ...
            Tamminen, J.: An adaptive Metropolis algorithm. Bernoulli 7(2), ...
            223-242 (2001).
        accept=0;
        while accept==0
            if t<500
                gamma_1=mvnrnd(gamma_T(:,t-1)',eye(7)*c_1);
            else
                gamma_1=mvnrnd(gamma_T(:,t-1)',cov(gamma_T(:,1:t-1)') ...
                *2.38^2/7)*0.6+mvnrnd(gamma_T(:,t-1)',eye(7)*c_1)*0.4;
            end
            if  gamma_1(7)>0
                accept=1;
            end
        end
        gamma_2=gamma_T(:,t-1);

        %% Propose lambda by adaptive M-H.
        accept=0;
        while accept==0
            if t≤500
                lambda_1=randn(1)*c_2+lambda_T(t-1);
            else
                lambda_1=mvnrnd(lambda_T(t-1)',cov(lambda_T(1:t-1)')*2.38^2)...
                *0.6 + mvnrnd(lambda_T(t-1)',eye(1)*c_2^2)*0.4;
```

```
            end
            if abs(lambda_1)≤1/20
                accept=1;
            end
        end

        %% Propose beta by adaptive M-H.
        if t≤500
            beta_1=randn(1)*c_3+beta_T(t-1);
        else
            beta_1=mvnrnd(beta_T(t-1)',cov(beta_T(1:t-1)')*2.38^2)*0.6...
                + mvnrnd(beta_T(t-1)',eye(1)*c_3^2)*0.4;
        end

        H=zeros(N,N);
        for i=1:N
            for j=1:N
                if j≠i
                    H(i,j)=gamma_1(1) ...
                    +gamma_1(2)*C(i,j)+gamma_1(3)*abs(X(i)-X(j));
                end
            end
        end

        S=eye(N)-lambda_1*W;
        S_INV=inv(S);

        S_new=S;
        S_old=S;
        W_old=W;
        W_new=W;

        FE=X*beta_1;

        S_INV_OLD=S_INV;
        Y2_star=S\FE;
        S_INV_OLD2=S_INV_OLD*gamma(7);
        Y2=mvnrnd(zeros(N,1),S_INV_OLD2,1)'+Y2_star;

        loglike_y_old=log(mvnpdf(Y2,Y2_star,S_INV_OLD2));

        for r=1:R  % Start to simulate auxiliary network and outcome.
            for i=1:N
                for j=1:N
                    if i≠j
                        W_new(i,j)=1-W_new(i,j);
                        W_new(j,i)=1-W_new(j,i);

                        S_INV_TEMP=S_INV;
                        if W_new(i,j)==1
                            S_INV_TEMP=-(-lambda_1)/(1+(-lambda_1)...
                            *S_INV(i,j))*S_INV(1:N,i)*S_INV(j,1:N)+S_INV_TEMP;
                            S_INV_NEW=S_INV_TEMP;
                            S_INV_NEW=-(-lambda_1)/(1+(-lambda_1)...
                            *S_INV_TEMP(i,j))...
                            *S_INV_TEMP(1:N,j)...
```

```
                                *S_INV_TEMP(i,1:N)+S_INV_NEW;
                                S_new(i,j)=S_new(i,j)-lambda_1;
                                S_new(j,i)=S_new(j,i)-lambda_1;
                            else
                                S_INV_TEMP=(-lambda_1)/(1-(-lambda_1)*S_INV(i,j))...
                                *S_INV(1:N,i)...
                                *S_INV(j,1:N)+S_INV_TEMP;
                                S_INV_NEW=S_INV_TEMP;
                                S_INV_NEW=(-lambda_1)/(1-(-lambda_1)...
                                *S_INV_TEMP(i,j))...
                                *S_INV_TEMP(1:N,j)...
                                *S_INV_TEMP(i,1:N)+S_INV_NEW;
                                S_new(i,j)=S_new(i,j)+lambda_1;
                                S_new(j,i)=S_new(j,i)+lambda_1;
                            end

                            loglike_y_new=loglike_y_old;

                            if rand(1)<=0.01   % Update outcome.
                                Y1_star=S_INV_NEW\FE;
                                S_INV_NEW2=S_INV_NEW*gamma_1(7);
                                Y1=mvnrnd(zeros(N,1),S_INV_NEW2,1)'+Y1_star;
                                loglike_y_new=log(mvnpdf(Y1,Y1_star,S_INV_NEW2));
                            else
                                Y1=Y2;
                            end

                            PHI1=FE'*Y1-0.5*Y1'*S_new*Y1;
                            PHI2=FE'*Y2-0.5*Y2'*S_old*Y2;

                            popularity=sum(W_new(i,:))-W_new(i,j)+sum(W_new(j,:))...
                            -W_new(i,j);
                            congestion=popularity^2;
                            cyclic=W_new(i,:)*W_new(j,:)'-W_new(i,j);

                            p_w=(H(i,j)+gamma_1(4)*popularity+gamma_1(5)*congestion...
                            +gamma_1(6)*cyclic)*(-1)^(1-W_new(i,j))+PHI1-PHI2;

                            p_w=p_w/gamma_1(7)+loglike_y_new-loglike_y_old;

                            if log(rand(1))<=p_w
                                W_old(i,j)=W_new(i,j);
                                W_old(j,i)=W_new(j,i);
                                S_old(i,j)=S_old(i,j);
                                S_old(j,i)=S_old(j,i);
                                S_INV=S_INV_NEW;
                                loglike_y_old=loglike_y_new;
                                Y2=Y1;
                            end
                            W_new(i,j)=W_old(i,j);
                            W_new(j,i)=W_old(j,i);
                            S_new(i,j)=S_old(i,j);
                            S_new(j,i)=S_old(j,i);
                        end
                    end
                end
```

```
        end

        if (abs(sum(sum(W_new,2))-sum(sum(W,2)))>50)  % Condition to reject ...
            auxiliary network.
            gamma_T(:,t)=gamma_T(:,t-1);
            lambda_T(t)=lambda_T(t-1);
            beta_T(t)=beta_T(t-1);
        else
            psi_1=zeros(N,N);
            psi_2=zeros(N,N);
            psi_3=zeros(N,N);
            psi_4=zeros(N,N);
            for i=1:N
                for j=1:N
                    if j≠i
                        popularity=sum(W(i,:))-W(i,j)+sum(W(j,:))-W(i,j);
                        congestion=popularity^2;
                        cyclic=W(i,:)*W(j,:)'-W(i,j);

                        popularity_new=sum(W_new(i,:))-W_new(i,j)...
                            +sum(W_new(j,:))-W_new(i,j);

                        congestion_new=popularity_new^2;
                        cyclic_new=W_new(i,:)*W_new(j,:)'-W_new(i,j);

                        psi_1(i,j)=gamma_1(1)+gamma_1(2)*C(i,j) ...
                            +gamma_1(3)*abs(X(i)-X(j)) ...
                            +gamma_1(4)*popularity+gamma_1(5)*congestion...
                            +(1.0/3.0)*gamma_1(6)*cyclic;

                        psi_2(i,j)=gamma_2(1)+gamma_2(2)*C(i,j)...
                            +gamma_2(3)*abs(X(i)-X(j)) ...
                            +gamma_2(4)*popularity+gamma_2(5)*congestion...
                            +(1.0/3.0)*gamma_2(6)*cyclic;

                        psi_3(i,j)=gamma_2(1)+gamma_2(2)*C(i,j)...
                            +gamma_2(3)*abs(X(i)-X(j)) ...
                            +gamma_2(4)*popularity_new ...
                            +gamma_2(5)*congestion_new...
                            +(1.0/3.0)*gamma_2(6)*cyclic_new;

                        psi_4(i,j)=gamma_1(1)+gamma_1(2)*C(i,j)...
                            +gamma_1(3)*abs(X(i)-X(j)) ...
                            +gamma_1(4)*popularity_new ...
                            +gamma_1(5)*congestion_new...
                            +(1.0/3.0)*gamma_1(6)*cyclic_new;

                    end
                end
            end

            psi_1=psi_1/gamma_1(7);
            psi_2=psi_2/gamma_T(7,t-1);
            psi_3=psi_3/gamma_T(7,t-1);
            psi_4=psi_4/gamma_1(7);
```

```
            p_w= trace(psi_1*W)-trace(psi_2*W)+trace(psi_3*W_new)...
            -trace(psi_4*W_new);

            S1=eye(N)-lambda_1*W;
            S2=eye(N)-lambda_T(t-1)*W;

            FE1=X*beta_1;
            FE2=X*beta_T(t-1);

            PHI1=FE1'*Y-0.5*Y'*S1*Y;
            PHI2=FE2'*Y-0.5*Y'*S2*Y;

            PHI1=PHI1/gamma_1(7);
            PHI2=PHI2/gamma_T(7,t-1);

            S3=eye(N)-lambda_T(t-1)*W_new;
            S4=eye(N)-lambda_1*W_new;

            PHI3=FE2'*Y2-0.5*Y2'*S3*Y2;
            PHI4=FE1'*Y2-0.5*Y2'*S4*Y2;

            PHI3=PHI3/gamma_T(7,t-1);
            PHI4=PHI4/gamma_1(7);

            pp=p_w/2+(PHI1-PHI2)+(PHI3-PHI4);

            pp = pp + log(mvnpdf(gamma_1,gamma_0,G_0)) ...
                - log(mvnpdf(gamma_T(:,t-1)',gamma_0,G_0))...
                    + log(mvnpdf(beta_1,beta_0,B_0))...
                    - log(mvnpdf(beta_T(t-1),beta_0,B_0));

            if log(rand(1))≤pp
                gamma_T(:,t)=gamma_1;
                lambda_T(t)=lambda_1;
                beta_T(t)=beta_1;
                acc_1=acc_1+1.0;
            else
                gamma_T(:,t)=gamma_T(:,t-1);
                lambda_T(t)=lambda_T(t-1);
                beta_T(t)=beta_T(t-1);
            end
        end

        acc_rate1(t)=acc_1/t;
        time=toc;

        if (t/1)-round(t/1)==0
            fprintf('t=%d\n',t);
            fprintf('time= %5.3f Secs\n',time);
            fprintf('lambda= %5.3f\n',lambda_T(t));
            fprintf('beta= %5.3f\n',beta_T(t));
            fprintf('gamma= %5.3f %5.3f %5.3f %5.3f %5.3f %5.3f ...
                %5.3f\n',gamma_T(:,t)');
            fprintf('acc_rate1= %5.3f\n',acc_rate1(t));
            fprintf('\n');
        end
```

```
    end

end
```

2. Report the parameter estimates and analyze the convergence of the algorithm. For the convergence analysis the Matlab script **chainstats.m** can be used (Geweke, 1992).

```
%% Plot the simulated parameter draws.
figure();
set(gca,'Layer','top')
set(gca,'FontSize',18);
subplot(2,1,1)
set(gca,'defaulttextinterpreter','latex')
plot(lambda_T,'-or');
hold on
plot(ones(length(lambda_T),1)*mean(lambda_T),'-k');
hold on
plot(ones(length(lambda_T),1)*mean(lambda_T)+std(lambda_T),'--k');
hold on
plot(ones(length(lambda_T),1)*mean(lambda_T)-std(lambda_T),'--k');
ylabel('$\lambda$')
xlabel('$t$')
subplot(2,1,2)
set(gca,'defaulttextinterpreter','latex')
plot(beta_T,'-ob');
hold on
plot(ones(length(beta_T),1)*mean(beta_T),'-k');
hold on
plot(ones(length(beta_T),1)*mean(beta_T)+std(beta_T),'--k');
hold on
plot(ones(length(beta_T),1)*mean(beta_T)-std(beta_T),'--k');
ylabel('$\beta$')
xlabel('$t$')

%% Analyze convergence following Geweke (1992).
chain = horzcat(lambda_T,beta_T);
chainstats(chain)

%% Compute p-values under the assumption of asymptotic normality.
z = mean(chain)./std(chain);
pvalue = 2*(1 - normcdf(z));
disp(['P-values: ' num2str(pvalue)])
```

Appendix

A Individual and Time Fixed Effects

Following Wansbeek and Kapteyn (1989) we consider the SAR model

$$\boldsymbol{y}_t = \lambda \boldsymbol{A}_t \boldsymbol{y}_t + \boldsymbol{X}_t \boldsymbol{\beta} + \boldsymbol{\eta} + \xi_t \mathbf{u}_n + \boldsymbol{\epsilon}_t.$$

In this model, $\boldsymbol{y}_t = (y_{1,t}, \cdots, y_{n,t})^\top$, where $y_{i,t}$ is the outcome of individual i at period t. $\boldsymbol{X}_t = (\boldsymbol{x}_{1,t}, \cdots, \boldsymbol{x}_{n,t})^\top$, where $\boldsymbol{x}_{i,t}$ is a $k \times 1$ vector of exogenous regressors. $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_n)^\top$. $\mathbf{u}_n$ is n-dimensional vector of ones.

For all T time periods, let $\boldsymbol{y} = (\boldsymbol{y}_1^\top, \cdots, \boldsymbol{y}_T^\top)^\top$, $\boldsymbol{X} = (\boldsymbol{X}_1^\top, \cdots, \boldsymbol{X}_T^\top)^\top$, $\boldsymbol{A} = \mathrm{diag}\{\boldsymbol{A}_t\}_{t=1}^n$, $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_T)^\top$, and $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_1^\top, \cdots, \boldsymbol{\epsilon}_T^\top)^\top$. The model can be rewritten as

$$\boldsymbol{y} = \lambda \boldsymbol{A} \boldsymbol{y} + \boldsymbol{X}\boldsymbol{\beta} + \mathbf{u}_T \otimes \boldsymbol{\eta} + \boldsymbol{\xi} \otimes \mathbf{u}_n + \boldsymbol{\epsilon}.$$

Let n_t be the number of observed firms in year t. Let $\boldsymbol{D}_t$ be the $n_t \times n$ matrix obtained from the $n \times n$ identity matrix from which rows corresponding to firms not observed in year t have been omitted. Let $\boldsymbol{D} = \mathrm{diag}\{\boldsymbol{D}_t\}_{t=1}^T$. Then, the model for the observed data is

$$\begin{aligned} \boldsymbol{D}\boldsymbol{y} &= \lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{D}(\mathbf{u}_T \otimes \boldsymbol{\eta}) + \boldsymbol{D}(\boldsymbol{\xi} \otimes \mathbf{u}_n) + \boldsymbol{D}\boldsymbol{\epsilon}, \\ \boldsymbol{D}\boldsymbol{y} &= \lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{X}\boldsymbol{\beta} + [\boldsymbol{D}_1^\top, \cdots, \boldsymbol{D}_T^\top]^\top \boldsymbol{\eta} + \mathrm{diag}\{\boldsymbol{D}_t \mathbf{u}_n\}_{t=1}^T \boldsymbol{\xi} + \boldsymbol{D}\boldsymbol{\epsilon}. \end{aligned}$$

Let $\boldsymbol{Z} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2]$, where $\boldsymbol{Z}_1 = [\boldsymbol{D}_1^\top, \cdots, \boldsymbol{D}_T^\top]^\top$ and $\boldsymbol{Z}_2 = \mathrm{diag}\{\boldsymbol{D}_t \mathbf{u}_n\}_{t=1}^T$. Let $\bar{\boldsymbol{Z}} = [\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top \boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top]\boldsymbol{Z}_2$ and $\boldsymbol{P} = [\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top \boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top] - \bar{\boldsymbol{Z}}(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}})^{-1}\bar{\boldsymbol{Z}}^\top$. As

$$\begin{aligned} \boldsymbol{P}\boldsymbol{Z}_1 &= [\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top \boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top]\boldsymbol{Z}_1 - \bar{\boldsymbol{Z}}(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}})^{-1}\boldsymbol{Z}_2^\top[\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top \boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top]\boldsymbol{Z}_1 = 0 \\ \boldsymbol{P}\boldsymbol{Z}_2 &= [\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top \boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top]\boldsymbol{Z}_2 - \bar{\boldsymbol{Z}}(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}})^{-1}\boldsymbol{Z}_2^\top[\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top \boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top]\boldsymbol{Z}_2 \\ &= \bar{\boldsymbol{Z}} - \bar{\boldsymbol{Z}}(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}})^{-1}\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}} = 0 \end{aligned}$$

we have $\boldsymbol{P}\boldsymbol{Z} = 0$, and thus we can use the projector $\boldsymbol{P}$ to eliminate the fixed effects

$$\boldsymbol{P}\boldsymbol{D}\boldsymbol{y} = \lambda \boldsymbol{P}\boldsymbol{D}\boldsymbol{A}\boldsymbol{y} + \boldsymbol{P}\boldsymbol{D}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{P}\boldsymbol{D}\boldsymbol{\epsilon}.$$

As an alternative, define $\bar{\boldsymbol{Z}} = [\boldsymbol{I} - \boldsymbol{Z}_2(\boldsymbol{Z}_2^\top \boldsymbol{Z}_2)^{-1}\boldsymbol{Z}_2^\top]\boldsymbol{Z}_1$ and $\boldsymbol{P} = [\boldsymbol{I} - \boldsymbol{Z}_2(\boldsymbol{Z}_2^\top \boldsymbol{Z}_2)^{-1}\boldsymbol{Z}_2^\top] - \bar{\boldsymbol{Z}}(\bar{\boldsymbol{Z}}^\top \bar{\boldsymbol{Z}})^{-1}\bar{\boldsymbol{Z}}^\top$. Use this projector if $\mathrm{rank}(\boldsymbol{Z}_1) < \mathrm{rank}(\boldsymbol{Z}_2)$.

To recover the fixed effects, one can regress the residuals on the time dummy and individual dummy by the OLS to estimate the coefficients.

B Individual and Time-varying Market Fixed Effects

We consider the following SAR model

$$\boldsymbol{y}_t = \lambda \boldsymbol{A}_t \boldsymbol{y}_t + \boldsymbol{X}_t \boldsymbol{\beta} + \boldsymbol{\eta} + \boldsymbol{L}_n \xi_t + \boldsymbol{\epsilon}_t.$$

In this model, $\boldsymbol{y}_t = (y_{1,t}, \cdots, y_{n,t})^\top$, where $y_{i,t}$ is the outcome of individual i at period t. $\boldsymbol{X}_t = (\boldsymbol{x}_{1,t}, \cdots, \boldsymbol{x}_{n,t})^\top$, where $\boldsymbol{x}_{i,t}$ is a $k \times 1$ vector of exogenous regressors. $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_n)^\top$. $\xi_t = (\xi_{1,t}, \cdots, \xi_{\bar{r},t})^\top$. $\boldsymbol{L}_n = \mathrm{diag}\{\mathbf{u}_{n_r}\}_{r=1}^{\bar{r}}$ where $\mathbf{u}_{n_r}$ is an n_r-dimensional vector of ones.

For all T time periods, let $\boldsymbol{y} = (\boldsymbol{y}_1^\top, \cdots, \boldsymbol{y}_T^\top)^\top$, $\boldsymbol{X} = (\boldsymbol{X}_1^\top, \cdots, \boldsymbol{X}_T^\top)^\top$, $\boldsymbol{A} = \mathrm{diag}\{\boldsymbol{A}_t\}_{t=1}^n$, $\boldsymbol{\xi} = (\xi_1^\top, \cdots, \xi_T^\top)^\top$, and $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_1^\top, \cdots, \boldsymbol{\epsilon}_T^\top)^\top$. The model can be rewritten as

$$\boldsymbol{y} = \lambda \boldsymbol{A}\boldsymbol{y} + \boldsymbol{X}\boldsymbol{\beta} + \mathbf{u}_T \otimes \boldsymbol{\eta} + (\boldsymbol{I}_T \otimes \boldsymbol{L}_n)\boldsymbol{\xi} + \boldsymbol{\epsilon}.$$

Let n_t be the number of observed firms in year t. Let $\boldsymbol{D}_t$ be the $n_t \times n$ matrix obtained from the $n \times n$ identity matrix from which rows corresponding to firms not observed in year t have been omitted. Let $\boldsymbol{D} = \mathrm{diag}\{\boldsymbol{D}_t\}_{t=1}^T$. Then, the model for the observed data is

$$\begin{aligned}\boldsymbol{D}\boldsymbol{y} &= \lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{D}(\mathbf{u}_T \otimes \boldsymbol{\eta}) + \boldsymbol{D}(\boldsymbol{I}_T \otimes \boldsymbol{L}_n)\boldsymbol{\xi} + \boldsymbol{D}\boldsymbol{\epsilon} \\ &= \lambda \boldsymbol{D}\boldsymbol{A}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{X}\boldsymbol{\beta} + [\boldsymbol{D}_1^\top, \cdots, \boldsymbol{D}_T^\top]^\top \boldsymbol{\eta} + \mathrm{diag}\{\boldsymbol{D}_t\boldsymbol{L}_n\}_{t=1}^T \boldsymbol{\xi} + \boldsymbol{D}\boldsymbol{\epsilon}.\end{aligned}$$

Let $\boldsymbol{Z} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2]$, where $\boldsymbol{Z}_1 = [\boldsymbol{D}_1^\top, \cdots, \boldsymbol{D}_T^\top]^\top$ and $\boldsymbol{Z}_2 = \mathrm{diag}\{\boldsymbol{D}_t\boldsymbol{L}_n\}_{t=1}^T$. Let $\bar{\boldsymbol{Z}} = [\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top]\boldsymbol{Z}_2$ and $\boldsymbol{P} = [\boldsymbol{I} - \boldsymbol{Z}_1(\boldsymbol{Z}_1^\top\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1^\top] - \bar{\boldsymbol{Z}}(\bar{\boldsymbol{Z}}^\top\bar{\boldsymbol{Z}})^{-1}\bar{\boldsymbol{Z}}^\top$. As $\boldsymbol{P}\boldsymbol{Z} = 0$, we can use the projector $\boldsymbol{P}$ to eliminate the fixed effects so that

$$\boldsymbol{P}\boldsymbol{D}\boldsymbol{y} = \lambda \boldsymbol{P}\boldsymbol{D}\boldsymbol{A}\boldsymbol{y} + \boldsymbol{P}\boldsymbol{D}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{P}\boldsymbol{D}\boldsymbol{\epsilon}.$$

C Double Metropolis Hastings (DMH) Algorithm

Given an observation (G, Y) from the stationary distribution defined in Equation (2), we can estimate the parameter vector θ based on the maximum likelihood principle. However, the frequentist maximum likelihood method is impractical due to the computational difficulty in evaluating the normalizing constant $c(\theta)$ in Equation (2), and a standard Bayesian method would encounter the same problem because, with the prior distribution $p(\theta)$, the posterior distribution $p(\theta|G, Y) \propto \pi(G, Y|\theta)p(\theta) = c(\theta)^{-1}\exp[\sigma^{-2}\Phi(G, Y|\gamma)]p(\theta)$ also contains the normalizing constant $c(\theta)$.

To sample from the posterior using Markov Chain Monte Carlo (MCMC) simulation, a standard MH algorithm (Chib and GreenberG, 1995) updates θ to $\widetilde{\theta}$, a random draw from the proposal distribution $q_\theta(\widetilde{\theta}|\theta)$, according to the acceptance probability

$$\alpha_{\theta,MH} = \min\left\{1, \frac{p(\widetilde{\theta}|G, Y)q_\theta(\theta|\widetilde{\theta})}{p(\theta|G, Y)q_\theta(\widetilde{\theta}|\theta)}\right\} = \min\left\{1, \frac{c(\theta)\exp[\widetilde{\sigma}^{-2}\Phi(G, Y|\widetilde{\gamma})]p(\widetilde{\theta})q_\theta(\theta|\widetilde{\theta})}{c(\widetilde{\theta})\exp[\sigma^{-2}\Phi(G, Y|\gamma)]p(\theta)q_\theta(\widetilde{\theta}|\theta)}\right\}.$$

The computational problem still exists as $c(\theta)$ and $c(\widetilde{\theta})$ in the acceptance probability do not cancel each other.

A way to bypass the evaluation of the intractable normalizing constant $c(\theta)$ is to use the exchange algorithm (Murray et al., 2006), which takes the following steps at each iteration:

Algorithm 1 (Exchange Algorithm)**.**

Step 1 *Draw $\widetilde{\theta}$ from the proposal distribution $q_\theta(\widetilde{\theta}|\theta)$.*

Step 2 *Generate $(\widetilde{G}, \widetilde{Y})$ from the distribution $\pi(G, Y|\widetilde{\theta})$ using a perfect sampler.*

Step 3 *Accept $\widetilde{\theta}$ according to the acceptance probability*

$$\begin{aligned} \alpha_{\theta,EX} &= \min\left\{1, \frac{p(\widetilde{\theta}|G,Y)q_\theta(\theta|\widetilde{\theta})\pi(\widetilde{G},\widetilde{Y}|\theta)}{p(\theta|G,Y)q_\theta(\widetilde{\theta}|\theta)\pi(\widetilde{G},\widetilde{Y}|\widetilde{\theta})}\right\} \\ &= \min\left\{1, \frac{\exp[\widetilde{\sigma}^{-2}\Phi(G,Y|\widetilde{\gamma})]p(\widetilde{\theta})q_\theta(\theta|\widetilde{\theta})\exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y}|\gamma)]}{\exp[\sigma^{-2}\Phi(G,Y|\gamma)]p(\theta)q_\theta(\widetilde{\theta}|\theta)\exp[\widetilde{\sigma}^{-2}\Phi(\widetilde{G},\widetilde{Y}|\widetilde{\gamma})]}\right\}. \end{aligned} \tag{3}$$

The main advantage of the exchange algorithm is that the acceptance probability does not contain the normalizing constant $c(\theta)$ and thus can be evaluated.[3]

In the second step of the exchange algorithm, we need to generate auxiliary date using a perfect sampler (Propp and Wilson, 1996), which is computationally costly for our model and, more generally, exponential random graph models (ERGMs) (Wasserman and Pattison, 1996). To overcome this issue, LianG (2010) and Mele (2017) propose a DMH algorithm, which uses a finite run of the MH algorithm initialized at the observed (G,Y) to generate auxiliary data $(\widetilde{G},\widetilde{Y})$. More specifically, at each iteration, the DMH algorithm follows the same steps as the exchange algorithm with the second step replaced by:

Step 2* Generate $(\widetilde{G},\widetilde{Y})$ from the distribution $\pi(G,Y|\widetilde{\theta})$ using a finite run of the MH algorithm initialized at the observed (G,Y).

We need to simulate both networks $\widetilde{G}$ and effort choices $\widetilde{Y}$ in Step 2* of the DMH algorithm. To generate $(\widetilde{G},\widetilde{Y})$ as follows:

Algorithm 2 (Auxiliary Data Generation)**.** *Given θ, at each iteration:*

Step 1 *Draw $\widetilde{G}$ from the proposal distribution $q_G(\widetilde{G}|g)$. Let $\widetilde{G}$ denote the adjacency matrix of $\widetilde{G}$.*

Step 2 *Generate $\widetilde{Y} \sim N(\widetilde{Y}^*, \Sigma_{\widetilde{Y}})$, where $\widetilde{Y}^* \equiv (I_n - \lambda\widetilde{A})^{-1}B(X)$, with $B(X) = [b(X_1),\cdots,b(X_n)]^\top$, is the equilibrium effort vector derived from the best response function, and $\Sigma_{\widetilde{Y}} = \sigma^2(I_n - \lambda\widetilde{A})^{-1}$.*

Step 3 *Accept $(\widetilde{G},\widetilde{Y})$ according to the acceptance probability*

$$\begin{aligned} \alpha_{(G,Y),MH} &= \min\left\{1, \frac{\pi(\widetilde{G},\widetilde{Y}|\theta)p_Y(Y|g)q_G(g|\widetilde{G})}{\pi(G,Y|\theta)p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|g)}\right\} \\ &= \min\left\{1, \frac{\exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y}|\gamma)]p_Y(Y|g)q_G(g|\widetilde{G})}{\exp[\sigma^{-2}\Phi(G,Y|\gamma)]p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|g)}\right\}, \end{aligned}$$

where $p_Y(\widetilde{Y}|\widetilde{G})$ denotes the density function of $N(\widetilde{Y}^*, \Sigma_{\widetilde{Y}})$.

In the following proposition, we show that the long run stationary distribution of the proposed MH algorithm is the Gibbs measure defined in Equation (2).

Proposition 1. *The unique stationary distribution of Algorithm 2 is $\pi(G,Y|\theta)$.*

[3] See LianG (2010) for discussion on the motivation and justification of the exchange algorithm.

Proof of Proposition 1. To show $\pi(G,Y|\theta)$ is the stationary distribution, we need to check the detailed balance condition, i.e., $\pi(G,Y|\theta)p(\widetilde{G},\widetilde{Y}|G,Y) = \pi(\widetilde{G},\widetilde{Y}|\theta)p(G,Y|\widetilde{G},\widetilde{Y})$ where

$$p(\widetilde{G},\widetilde{Y}|G,Y) = p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|G)\min\left\{1, \frac{\pi(\widetilde{G},\widetilde{Y}|\theta)p_Y(Y|g)q_G(G|\widetilde{G})}{\pi(G,Y|\theta)p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|G)}\right\}.$$

Indeed,

$$\begin{aligned}
& \pi(G,Y|\theta)p(\widetilde{G},\widetilde{Y}|G,Y) \\
= \;& c(\theta)^{-1}\exp[\sigma^{-2}\Phi(G,Y|\gamma)]p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|g)\min\left\{1, \frac{\exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y}|\gamma)]p_Y(Y|g)q_G(g|\widetilde{G})}{\exp[\sigma^{-2}\Phi(G,Y|\gamma)]p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|g)}\right\} \\
= \;& c(\theta)^{-1}\min\left\{\exp[\sigma^{-2}\Phi(G,Y|\gamma)]p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|g), \exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y}|\gamma)]p_Y(Y|g)q_G(g|\widetilde{G})\right\} \\
= \;& \min\left\{\frac{\exp[\sigma^{-2}\Phi(G,Y|\gamma)]p_Y(\widetilde{Y}|\widetilde{G})q_G(\widetilde{G}|g)}{\exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y}|\gamma)]p_Y(Y|g)q_G(g|\widetilde{G})}, 1\right\} c(\theta)^{-1}\exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y}|\gamma)]p_Y(Y|g)q_G(g|\widetilde{G}) \\
= \;& \pi(\widetilde{G},\widetilde{Y}|\theta)p(G,Y|\widetilde{G},\widetilde{Y}).
\end{aligned}$$

The desired result follows by the reversibility, irreducibility, and Harris recurrence of the Markov chain. □

In Step 2 of Algorithm 2, we generate $\widetilde{Y}$ from a multivariate normal distribution. We assume that link adjustment periods arrive much less frequent than effort adjustment periods (i.e., ρ_0 is very small) in the coevolution process. Given the network $\widetilde{G}$, it follows a standard Gibbs sampler argument that the transition density converges to

$$p_Y(\widetilde{Y}|\widetilde{G}) = \frac{\exp[\sigma^{-2}\Phi(\widetilde{G},\widetilde{Y})]}{\int_{\mathcal{Y}^n}\exp[\sigma^{-2}\Phi(\widetilde{G},Y)]dY}. \tag{4}$$

where

$$\begin{aligned}
\Phi(G,Y) &= \kappa(G) + \sum_{i\in\mathcal{N}} b(X_i)y_i + \frac{\lambda}{2}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} a_{ij}y_iy_j - \frac{1}{2}\sum_{i\in\mathcal{N}} y_i^2 \qquad (5)\\
&= \kappa(G) + B(X)^\top Y - \frac{1}{2}Y^\top(I_n - \lambda A)Y.
\end{aligned}$$

Inserting Equation (5) into Equation (4), it follows by the Gaussian integral formula that

$$\begin{aligned}
p_Y(\widetilde{Y}|\widetilde{G}) &= \frac{\exp[\sigma^{-2}B(X)^\top\widetilde{Y} - \frac{1}{2}\sigma^{-2}\widetilde{Y}^\top(I_n - \lambda\widetilde{A})\widetilde{Y}]}{\int_{\mathcal{Y}^n}\exp[\sigma^{-2}B(X)^\top Y - \frac{1}{2}\sigma^{-2}Y^\top(I_n - \lambda\widetilde{A})Y]dY} \\
&= (2\pi)^{-n/2}|\det\Sigma_{\widetilde{Y}}|^{-1/2}\exp[-\frac{1}{2}(\widetilde{Y} - \widetilde{Y}^*)^\top\Sigma_{\widetilde{Y}}^{-1}(\widetilde{Y} - \widetilde{Y}^*)]
\end{aligned}$$

which is the density function of $N(\widetilde{Y}^*, \Sigma_{\widetilde{Y}})$.

References

Ballester, C., Antoni Calvó-Armengol, A. and Zenou, Y.(2006). Who's Who in Networks. Wanted: The Key Player. *Econometrica*, 74(5):1403–1417.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335.

Geweke, J. (1992). Evaluating the accuracy of sampling-based approaches to the calculations of posterior moments. *Bayesian statistics*, 4:641–649.

Haario, H., Saksman, E. and Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242.

Hsieh C.S., König, M. D. and Liu, X. (2018). Network Formation with Local Complements and Global Substitutes: The Case of R&D Networks. CEPR Discussion Paper No. DP13161.

König, M. D., Liu, X., and Zenou, Y. (2018). R&D Networks: Theory, empirics and policy implications. *Review of Economics and Statistics.*

LianG, F. (2010). A double Metropolis-Hastings sampler for spatial models with intractable normalizing constants. *Journal of Statistical Computation and Simulation*, 80:1007 1022.

Mele, A. (2017). A structural model of dense network formation. *Econometrica*, 85: 825–850.

Murray, I., Ghahramani, Z. and MacKay, D. (2006). MCMC for doubly-intractable distributions. in R. Dechter and T. S. Richardson (eds), Proceedings of 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI), AUAI Press: Cambridge, MA, pp. 359 366.

Propp, J. and Wilson, D. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223 252.

Raftery, A.E. and Lewis, S.M. (1992). Practical Markov Chain Monte Carlo: comment: one long run with diagnostics: implementation strategies for Markov Chain Monte Carlo. *Statistical Science*, 7(4):493–497.

Smith, T. E. and LeSage, J. P. (2004). A bayesian probit model with spatial dependencies. *Advances in Econometrics*, 18:127 160.

Wansbeek, T. and Kapteyn, A. (1989). Estimation of the error-components model with incomplete panels. *Journal of Econometrics*, 41(3):341–361.

Wasserman, S. and Pattison, P. (1996). Logit models and logistic regressions for social networks: I. an introduction to Markov graphs and p*. *Psychometrika*, 61:401 425.