PartialNetwork: An R package for estimating peer effects using partial network information

Vincent Boucher and Elysée Aristide Houndetoungan

2023-10-07

Abstract

This vignette makes it easier to get started with the package **PartialNetwork**. We illustrate the estimators presented in Boucher and Houndetoungan (2022) with examples of simulated data. Section 1 illustrates the instrumental variable (IV) estimator, which is biased when the network is sampled. We calculate the bias and propose a general consistent estimator using a simulated general method of moments (SGMM) in Section 2. Section 3 presents the Bayesian estimator that jointly estimates the spatial autoregressive (SAR) model and the network formation model. Finally, we discuss in Section 4 how to address the problem of selection bias that can arise when missing links are not completely random.

Contents

1 Instrumental variable (IV) procedure

We provide the function `sim.IV(dnetwork, X, y, replication, power)` where `dnetwork` is the network linking probabilities, `X` is a matrix of covariates, `y` (optional) is the vector of outcome, `replication` (optional, default = 1) is the number of replications, and `power` (optional, default = 1) is the number of powers of the interaction matrix used to generate the instruments. The function outputs a proxy for `Gy` and simulated instruments.

1.1 Model without contextual effects

The following code provides an example using a sample of 30 networks of size 50 each. For the sake of the example, we assume that linking probabilities are *known* and drawn from an uniform distribution. We first simulate data. Then, we estimate the linear-in-means model using our IV procedure, using the known linking probabilities to generate approximations of the true network.

```
library(PartialNetwork)
set.seed(123)
# Number of groups
M             <- 30
# size of each group
N             <- rep(50,M)
# individual effects
beta          <- c(2,1,1.5)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# network distribution
distr         <- runif(sum(N*(N-1)))
distr         <- vec.to.mat(distr, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(distr)
# normalise
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = FALSE, Glist = G0norm,
                                  theta = c(alpha, beta, se))
y             <- y$y
# generate instruments
instr         <- sim.IV(distr, X, y, replication = 1, power = 1)
GY1c1         <- instr[[1]]$G1y       # proxy for Gy (draw 1)
GXc1          <- instr[[1]]$G1X[,,1]  # proxy for GX (draw 1)
GXc2          <- instr[[1]]$G2X[,,1]  # proxy for GX (draw 2)
# build dataset
# keep only instrument constructed using a different draw than the one used to proxy Gy
dataset           <- as.data.frame(cbind(y, X, GY1c1, GXc1, GXc2))
colnames(dataset) <- c("y","X1","X2","G1y", "G1X1", "G1X2", "G2X1", "G2X2")
```

Once the instruments are generated, the estimation can be performed using standard tools, e.g. the function `ivreg` from the **AER** package. For example, if we use the same draw for the proxy and the instruments, the estimation is "bad".

```r
library(AER)
# Same draws
out.iv1            <- ivreg(y ~ X1 + X2 + G1y | X1 + X2 + G1X1 + G1X2, data = dataset)
summary(out.iv1)
```

```
##
## Call:
## ivreg(formula = y ~ X1 + X2 + G1y | X1 + X2 + G1X1 + G1X2, data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.32409 -0.73973  0.02989  0.73541  3.86358
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.420695   0.433865   12.49   <2e-16 ***
## X1          1.003496   0.005585  179.67   <2e-16 ***
## X2          1.494316   0.010412  143.52   <2e-16 ***
## G1y         0.238036   0.020422   11.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 1496 degrees of freedom
## Multiple R-Squared: 0.9728,  Adjusted R-squared: 0.9728
## Wald test: 1.783e+04 on 3 and 1496 DF,  p-value: < 2.2e-16
```

If we use different draws for the proxy and the instruments, the estimation is “good”.

```r
# Different draws
out.iv2            <- ivreg(y ~ X1 + X2 + G1y | X1 + X2 + G2X1 + G2X2, data = dataset)
summary(out.iv2)
```

```
##
## Call:
## ivreg(formula = y ~ X1 + X2 + G1y | X1 + X2 + G2X1 + G2X2, data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.46502 -0.74230 -0.03304  0.76565  3.87127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.618919   0.690513   2.345   0.0192 *
## X1          1.005368   0.005677 177.085   <2e-16 ***
## X2          1.492886   0.010574 141.181   <2e-16 ***
## G1y         0.420011   0.032829  12.794   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.088 on 1496 degrees of freedom
## Multiple R-Squared: 0.972,   Adjusted R-squared: 0.9719
## Wald test: 1.73e+04 on 3 and 1496 DF,  p-value: < 2.2e-16
```

However, as stated by our Proposition 2, this estimator is biased. We can approximate the bias as follows.

```r
ddS     <- as.matrix(cbind(1, dataset[,c("X1", "X2", "G1y")]))         #\ddot{S}
dZ      <- as.matrix(cbind(1, dataset[,c("X1", "X2", "G2X1", "G2X2")]))#\dot{Z}
dZddS   <- crossprod(dZ, ddS)/sum(N)
W       <- solve(crossprod(dZ)/sum(N))
matM    <- solve(crossprod(dZddS, W%*%dZddS), crossprod(dZddS, W))
maxbias <- apply(sapply(1:1000, function(...){
  dddGy <- peer.avg(sim.network(distr, normalise = TRUE) , y)
  abs(matM%*%crossprod(dZ, dddGy - dataset$G1y)/sum(N))
}), 1, max); names(maxbias) <- c("(Intercept)", "X1", "X2", "G1y")
{cat("Maximal absolute bias\n"); print(maxbias)}
```

```
## Maximal absolute bias
## (Intercept)          X1          X2         G1y
##  2.26468568  0.01612011  0.02774771  0.10832980
```

1.2 Model with contextual effects

We now assume that the model includes contextual effects. We first generate data.

```r
library(PartialNetwork)
set.seed(123)
# Number of groups
M             <- 30
# size of each group
N             <- rep(50,M)
# individual effects
beta          <- c(2,1,1.5)
# contextual effects
gamma         <- c(5, -3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# network distribution
distr         <- runif(sum(N*(N-1)))
distr         <- vec.to.mat(distr, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(distr)
# normalise
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                  theta = c(alpha, beta, gamma, se))
y             <- y$y
# GX
GX            <- peer.avg(G0norm, X)
# generate instruments
# we need power = 2 for models with contextual effetcs
instr         <- sim.IV(distr, X, y, replication = 1, power = 2)
GY1c1         <- instr[[1]]$G1y       # proxy for Gy (draw 1)
GXc1          <- instr[[1]]$G1X[,,1]  # proxy for GX (draw 1)
GXc2          <- instr[[1]]$G2X[,,1]  # proxy for GX (draw 2)
```

```r
GXc2sq         <- instr[[1]]$G2X[,,2]  # proxy for G^2X (draw 2)
# build dataset
# keep only instrument constructed using a different draw than the one used to proxy Gy
dataset           <- as.data.frame(cbind(y, X, GX, GY1c1, GXc1, GXc2, GXc2sq))
colnames(dataset) <- c("y","X1","X2", "GX1", "GX2", "G1y", "G1X1", "G1X2", "G2X1", "G2X2",
                       "G2X1sq", "G2X2sq")
```

As pointed out in the paper, the IV procedure requires **GX** being observed. In additions, when contextual effects are included, we consider the extended model.

```r
# Different draws
out.iv2           <- ivreg(y ~ X1 + X2 + GX1 + GX2 + G1X1 + G1X2 + G1y | X1 + X2 + GX1 +
                             GX2 + G1X1 + G1X2 + G2X1 + G2X2 + G2X1sq + G2X2sq,
                           data = dataset)
summary(out.iv2)
```

```
##
## Call:
## ivreg(formula = y ~ X1 + X2 + GX1 + GX2 + G1X1 + G1X2 + G1y |
##     X1 + X2 + GX1 + GX2 + G1X1 + G1X2 + G2X1 + G2X2 + G2X1sq +
##         G2X2sq, data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.12287 -0.70836 -0.01074  0.69947  3.42180
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.080986   0.423878   4.909 1.01e-06 ***
## X1           1.002130   0.005526 181.357  < 2e-16 ***
## X2           1.482665   0.010109 146.665  < 2e-16 ***
## GX1          5.282043   0.039440 133.927  < 2e-16 ***
## GX2         -2.325314   0.065593 -35.451  < 2e-16 ***
## G1X1        -0.383316   0.042162  -9.091  < 2e-16 ***
## G1X2        -0.660039   0.065543 -10.070  < 2e-16 ***
## G1y          0.405913   0.007856  51.671  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.034 on 1492 degrees of freedom
## Multiple R-Squared: 0.9868,  Adjusted R-squared: 0.9867
## Wald test: 1.59e+04 on 7 and 1492 DF,  p-value: < 2.2e-16
```

We also compute the maximal absolute bias.

```r
ddS     <- as.matrix(cbind(1, dataset[,c("X1", "X2", "GX1", "GX2", "G1X1", "G1X2",
                                         "G1y")]))
dZ      <- as.matrix(cbind(1, dataset[,c("X1", "X2", "GX1", "GX2", "G1X1",
                                         "G1X2", "G2X1", "G2X2", "G2X1sq", "G2X2sq")]))
dZddS   <- crossprod(dZ, ddS)/sum(N)
W       <- solve(crossprod(dZ)/sum(N))
matM    <- solve(crossprod(dZddS, W%*%dZddS), crossprod(dZddS, W))
maxbias <- apply(sapply(1:1000, function(...){
  dddGy <- peer.avg(sim.network(distr, normalise = TRUE) , y)
  abs(matM%*%crossprod(dZ, dddGy - dataset$G1y)/sum(N))
```

```r
}), 1, max); names(maxbias) <- c("(Intercept)", "X1", "X2", "GX1", "GX2", "G1X1",
                                 "G1X2", "G1y")
{cat("Maximal absolute bias\n"); print(maxbias)}
```

```
## Maximal absolute bias
## (Intercept)          X1          X2         GX1         GX2        G1X1
##  4.46212210  0.02409743  0.03819981  0.35564658  0.84623411  0.79857190
##        G1X2         G1y
##  1.23678405  0.05471398
```

2 Simulated Method of Moments

As shown in the paper (see Boucher and Houndetoungan (2022)), our IV estimator is inconsistent. Although the bias is expected to be small, in general, the IV estimator performs less well when the model includes contextual effects. Therefore, we propose a Simulated Method of Moments (SMM) estimator by correcting the bias of the moment function use by the IV estimator. Our SMM estimator is then consistent and also deals with group heterogeneity.

2.1 Models without group heterogeneity

We first simulate data.

```r
library(PartialNetwork)
set.seed(123)
# Number of groups
M             <- 100
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(2, 1, 1.5, 5, -3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# network distribution
distr         <- runif(sum(N*(N-1)))
distr         <- vec.to.mat(distr, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(distr)
# normalise
G0norm        <- norm.network(G0)
# Matrix GX
GX            <- peer.avg(G0norm, X)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                  theta = c(alpha, beta, se))
Gy            <- y$Gy
y             <- y$y
# build dataset
dataset           <- as.data.frame(cbind(y, X, Gy, GX))
colnames(dataset) <- c("y","X1","X2", "Gy", "GX1", "GX2")
```

The estimation can be performed using the function `smmSAR` (do `?smmSAR` to read the help file of the function). The function allows to specify if `GX` and `Gy` are observed. We provide an example for each case.

If `GX` and `Gy` are observed (instruments will be constructed using the network distribution).

```
out.smm1       <- smmSAR(y ~ X1 + X2 | Gy | GX1 + GX2, dnetwork = distr, contextual = T,
                         smm.ctr  = list(R = 1, print = F), data = dataset)
summary(out.smm1)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2 | Gy | GX1 + GX2
##
## Contextual effects: Yes
## Fixed effects: No
##
## Network details
## GX Observed
## Gy Observed
## Number of groups: 100
## Sample size     : 3000
##
## Simulation settings
## R = 1
## Smoother : FALSE
##
## Coefficients:
##              Estimate Robust SE t value Pr(>|t|)
## Gy            0.402802  0.003384  119.05   <2e-16 ***
## (Intercept)   1.937466  0.304190    6.37  1.9e-10 ***
## X1            0.999482  0.003688  271.00   <2e-16 ***
## X2            1.498394  0.006721  222.93   <2e-16 ***
## GX1           4.991688  0.022285  224.00   <2e-16 ***
## GX2          -2.984391  0.038628  -77.26   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If `GX` is observed and not `Gy`.

```
out.smm2       <- smmSAR(y ~ X1 + X2 || GX1 + GX2, dnetwork = distr, contextual = T,
                         smm.ctr  = list(R = 1, print = F), data = dataset)
summary(out.smm2)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2 || GX1 + GX2
##
## Contextual effects: Yes
## Fixed effects: No
##
## Network details
## GX Observed
## Gy Not Observed
## Number of groups: 100
## Sample size     : 3000
##
```

```
## Simulation settings
## R = 1
## Smoother : FALSE
##
## Coefficients:
##              Estimate Robust SE t value Pr(>|t|)
## Gy           0.405399  0.004939   82.08   <2e-16 ***
## (Intercept)  2.163780  0.448694    4.82 1.42e-06 ***
## X1           0.993788  0.005105  194.68   <2e-16 ***
## X2           1.503612  0.009571  157.10   <2e-16 ***
## GX1          4.968672  0.033626  147.76   <2e-16 ***
## GX2         -3.016854  0.056747  -53.16   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If Gy is observed and not GX.

```
out.smm3       <- smmSAR(y ~ X1 + X2 | Gy, dnetwork = distr, contextual = T,
                         smm.ctr  = list(R = 100, print = F), data = dataset)
summary(out.smm3)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2 | Gy
##
## Contextual effects: Yes
## Fixed effects: No
##
## Network details
## GX Not Observed
## Gy Observed
## Number of groups: 100
## Sample size     : 3000
##
## Simulation settings
## R = 100
## Smoother : FALSE
##
## Coefficients:
##              Estimate Robust SE t value Pr(>|t|)
## Gy           0.434991  0.022240   19.56   <2e-16 ***
## (Intercept)  3.622088  1.700431    2.13   0.0332   *
## X1           0.972009  0.017013   57.13   <2e-16 ***
## X2           1.522467  0.029772   51.14   <2e-16 ***
## G: X1        4.748437  0.182937   25.96   <2e-16 ***
## G: X2       -3.187962  0.215019  -14.83   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If neither Gy nor GX are observed.

```
out.smm4       <- smmSAR(y ~ X1 + X2, dnetwork = distr, contextual = T,
                         smm.ctr  = list(R = 100, print = F), data = dataset)
summary(out.smm4)
```

```
## Simulated Method of Moments estimation of SAR model
```

```
##
## Formula = y ~ X1 + X2
##
## Contextual effects: Yes
## Fixed effects: No
##
## Network details
## GX Not Observed
## Gy Not Observed
## Number of groups: 100
## Sample size     : 3000
##
## Simulation settings
## R = 100
## Smoother : FALSE
##
## Coefficients:
##              Estimate Robust SE t value Pr(>|t|)
## Gy           0.435568  0.019749   22.05   <2e-16 ***
## (Intercept)  3.794497  1.614307    2.35   0.0187   *
## X1           0.970033  0.016865   57.52   <2e-16 ***
## X2           1.516125  0.030030   50.49   <2e-16 ***
## G: X1        4.725284  0.162581   29.06   <2e-16 ***
## G: X2       -3.205673  0.215419  -14.88   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.2 Models with group heterogeneity

We assume here that every group has an unobserved characteristic which affects the outcome. Let us first simulate the data.

```r
library(PartialNetwork)
set.seed(123)
# Number of groups
M             <- 200
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(1, 1, 1.5, 5, -3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# network distribution
distr         <- runif(sum(N*(N-1)))
distr         <- vec.to.mat(distr, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5), rpois(sum(N),7))
# Groups' fixed effects
# In order to have groups' heterogeneity correlated to X (fixed effects),
# We consider the quantile of X2 at 25% in the group
eff           <- unlist(lapply(1:M, function(x)
  rep(quantile(X[(c(0, cumsum(N))[x]+1):(cumsum(N)[x]),2], probs = 0.25), each = N[x])))
```

```r
print(c("cor(eff, X1)" = cor(eff, X[,1]), "cor(eff, X2)" = cor(eff, X[,2])))
```

```
## cor(eff, X1) cor(eff, X2)
##  0.005889583  0.116427543
```

```r
# We can see that eff is correlated to X2. We can confirm that the correlation is
# strongly significant.
print(c("p.value.cor(eff, X1)" = cor.test(eff, X[,1])$p.value,
        "p.value.cor(eff, X2)" = cor.test(eff, X[,2])$p.value))
```

```
## p.value.cor(eff, X1) p.value.cor(eff, X2)
##         6.483080e-01         1.464903e-19
```

```r
# true network
G0              <- sim.network(distr)
# normalise
G0norm          <- norm.network(G0)
# Matrix GX
GX              <- peer.avg(G0norm, X)
# simulate dependent variable use an external package
y               <- CDatanet::simsar(~ -1 + eff + X | X, Glist = G0norm,
                                    theta = c(alpha, beta, se))
Gy              <- y$Gy
y               <- y$y
# build dataset
dataset           <- as.data.frame(cbind(y, X, Gy, GX))
colnames(dataset) <- c("y","X1","X2", "Gy", "GX1", "GX2")
```

The group heterogeneity is correlated to $\mathbf{X}$ and induces bias if we do not control for it. In practice, we do not observe `eff` and we cannot add 200 dummies variables as explanatory variables to the model. We can control for group heterogeneity by taking the difference of each variable with respect to the group average (see Bramoullé et al. (2009)). To do this, we only need to set `fixed.effects = TRUE` in `smmSAR`. We provide examples.

If `GX` is observed and not `Gy`.

```r
out.smmeff1 <- smmSAR(y ~ X1 + X2 || GX1 + GX2, dnetwork = distr, contextual = T,
                      fixed.effects = T, smm.ctr  = list(R = 1, print = F),
                      data = dataset)
summary(out.smmeff1)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2 || GX1 + GX2
##
## Contextual effects: Yes
## Fixed effects: Yes
##
## Network details
## GX Observed
## Gy Not Observed
## Number of groups: 200
## Sample size     : 6000
##
## Simulation settings
## R = 1
```

```
## Smoother : FALSE
##
## Coefficients:
##      Estimate Robust SE t value Pr(>|t|)
## Gy   0.393514  0.054650    7.20    6e-13 ***
## X1   0.996275  0.003787  263.09   <2e-16 ***
## X2   1.499437  0.006454  232.33   <2e-16 ***
## GX1  5.021975  0.037127  135.27   <2e-16 ***
## GX2 -2.969625  0.094368  -31.47   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If `Gy` is observed and not `GX`.

```r
out.smmeff2 <- smmSAR(y ~ X1 + X2 | Gy, dnetwork = distr, contextual = T,
                      fixed.effects = T, smm.ctr  = list(R = 100, print = F),
                      data = dataset)
summary(out.smmeff2)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2 | Gy
##
## Contextual effects: Yes
## Fixed effects: Yes
##
## Network details
## GX Not Observed
## Gy Observed
## Number of groups: 200
## Sample size     : 6000
##
## Simulation settings
## R = 100
## Smoother : FALSE
##
## Coefficients:
##        Estimate Robust SE t value Pr(>|t|)
## Gy      0.244797  0.063231    3.87 0.000108 ***
## X1      1.002317  0.012512   80.11   <2e-16 ***
## X2      1.471207  0.024425   60.23   <2e-16 ***
## G: X1   5.261058  0.176945   29.73   <2e-16 ***
## G: X2  -2.889746  0.367652   -7.86 3.77e-15 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If neither `Gy` nor `GX` are observed.

```r
out.smmeff3 <- smmSAR(y ~ X1 + X2, dnetwork = distr, contextual = T, fixed.effects = T,
                      smm.ctr  = list(R = 100, print = F), data = dataset)
summary(out.smmeff3)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2
##
```

```
## Contextual effects: Yes
## Fixed effects: Yes
##
## Network details
## GX Not Observed
## Gy Not Observed
## Number of groups: 200
## Sample size     : 6000
##
## Simulation settings
## R = 100
## Smoother : FALSE
##
## Coefficients:
##         Estimate Robust SE t value Pr(>|t|)
## Gy      0.095920  0.199551    0.48    0.631
## X1      1.001540  0.012432   80.56   <2e-16 ***
## X2      1.474940  0.024380   60.50   <2e-16 ***
## G: X1   5.420860  0.156186   34.71   <2e-16 ***
## G: X2 -2.590426  0.382618   -6.77 1.29e-11 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see, the estimator with fixed effects has larger variance. Thus, we cannot illustrate its consistency using only one simulation. Therefore, we conduct Monte Carlo simulations.

We construct the function `fMC` which simulates data and computes the SMM estimator following the same process as described above.

```r
fMC <- function(...){
  # Number of groups
  M              <- 200
  # size of each group
  N              <- rep(30,M)
  # individual effects
  beta           <- c(1, 1, 1.5, 5, -3)
  # endogenous effects
  alpha          <- 0.4
  # std-dev errors
  se             <- 1
  # network distribution
  distr          <- runif(sum(N*(N-1)))
  distr          <- vec.to.mat(distr, N, normalise = FALSE)
  # covariates
  X              <- cbind(rnorm(sum(N),0,5), rpois(sum(N),7))
  # Groups' fixed effects
  # We defined the groups' fixed effect as the quantile at 25% of X2 in the group
  # This implies that the effects are correlated with X
  eff            <- unlist(lapply(1:M, function(x)
    rep(quantile(X[(c(0, cumsum(N))[x]+1):(cumsum(N)[x]),2], probs = 0.25), each = N[x])))
  # true network
  G0             <- sim.network(distr)
  # normalise
  G0norm         <- norm.network(G0)
  # Matrix GX
```

```r
  GX            <- peer.avg(G0norm, X)
  # simulate dependent variable use an external package
  y             <- CDatanet::simsar(~ -1 + eff + X | X, Glist = G0norm,
                                    theta = c(alpha, beta, se))
  Gy            <- y$Gy
  y             <- y$y
  # build dataset
  dataset           <- as.data.frame(cbind(y, X, Gy, GX))
  colnames(dataset) <- c("y","X1","X2", "Gy", "GX1", "GX2")
  out.smmeff1 <- smmSAR(y ~ X1 + X2 || GX1 + GX2, dnetwork = distr, contextual = T,
                        fixed.effects = T, smm.ctr  = list(R = 1, print = F),
                        data = dataset)
  out.smmeff2 <- smmSAR(y ~ X1 + X2 | Gy, dnetwork = distr, contextual = T,
                        fixed.effects = T, smm.ctr  = list(R = 100, print = F),
                        data = dataset)
  out.smmeff3 <- smmSAR(y ~ X1 + X2, dnetwork = distr, contextual = T, fixed.effects = T,
                       smm.ctr  = list(R = 100, print = F), data = dataset)
  out         <- data.frame("GX.observed"   = out.smmeff1$estimates,
                            "Gy.observed"   = out.smmeff2$estimates,
                            "None.observed" = out.smmeff3$estimates)
  out
}
```

We perform 250 simulations.

```r
smm.Monte.C   <- lapply(1:250, fMC)
```

We compute the average of the estimates

```r
Reduce('+', smm.Monte.C)/250
```

```
##     GX.observed Gy.observed None.observed
## Gy    0.3986183    0.404233      0.394228
## X1    0.9999592    1.001174      1.000887
## X2    1.4998792    1.499467      1.499780
## GX1   5.0014374    5.002064      5.003466
## GX2  -2.9996416   -3.013024     -2.989256
```

The SMM estimator performs well even when we only have the distribution of the network, $\mathbf{GX}$ and $\mathbf{Gy}$ are not observed, and the model includes group heterogeneity.

2.3 How to compute the variance when the network distribution is estimated?

In practice, the econometrician does not observed the true distribution of the entire network. They can only have an estimator based on partial network data (see Boucher and Houndetoungan (2022)). Because this estimator is used instead of the true distribution, this increases the variance of the SMM estimator. We develop a method to estimate the variance by taking into account the uncertainty related to the estimator of the network distribution.

Assume that the network distribution is logistic, ie,

$$\frac{p_{ij}}{1-p_{ij}} = \exp(\rho_0 + \rho_1|X_{i1} + X_{j1}| + \rho_2|X_{i2} - X_{j2}|) \tag{1}$$

and $\mathbb{P}(a_{ij} = 1|\mathbf{P}) = p_{ij}$, where $\mathbf{A}$ is the adjacency matrix, a_{ij} is the $(i,\, j)$-th entry of $\mathbf{A}$ and $\mathbf{P}$ is the matrix of p_{ij}. We simulated data following this assumption.

```r
library(PartialNetwork)
set.seed(123)
# Number of groups
M        <- 100
# size of each group
N        <- rep(30,M)
# covariates
X        <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# network formation model parameter
rho      <- c(-0.8, 0.2, -0.1)
# individual effects
beta     <- c(2, 1, 1.5, 5, -3)
# endogenous effects
alpha    <- 0.4
# std-dev errors
se       <- 1
# network
tmp      <- c(0, cumsum(N))
X1l      <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],1])
X2l      <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],2])
dist.net <- function(x, y) abs(x - y)
X1.mat   <- lapply(1:M, function(m) {
  matrix(kronecker(X1l[[m]], X1l[[m]], FUN = dist.net), N[m])})
X2.mat   <- lapply(1:M, function(m) {
  matrix(kronecker(X2l[[m]], X2l[[m]], FUN = dist.net), N[m])})
Xnet     <- as.matrix(cbind("Const" = 1,
                            "dX1"   = mat.to.vec(X1.mat),
                            "dX2"   = mat.to.vec(X2.mat)))
ynet     <- Xnet %*% rho
ynet     <- c(1*((ynet + rlogis(length(ynet))) > 0))
G0       <- vec.to.mat(ynet, N, normalise = FALSE)
# normalise
G0norm   <- norm.network(G0)
# Matrix GX
GX       <- peer.avg(G0norm, X)
# simulate dependent variable use an external package
y        <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                             theta = c(alpha, beta, se))
Gy       <- y$Gy
y        <- y$y
# build dataset
dataset           <- as.data.frame(cbind(y, X, Gy, GX))
colnames(dataset) <- c("y","X1","X2", "Gy", "GX1", "GX2")
```

We do not observed the true distribution of the network. We observe a subset of $\{a_{ij}\}$. Let $\mathcal{A}^{obs} = \{(i,\ j),\ a_{ij} \text{ is observed}\}$ and $\mathcal{A}^{nobs} = \{(i,\ j),\ a_{ij} \text{ is not observed}\}$. We assume that $|\mathcal{A}^{obs}| \to \infty$ as the sample size grows to ∞. Therefore, ρ_0, ρ_1, and ρ_2 can be consistently estimated.

```r
nNet      <- nrow(Xnet) # network formation model sample size
Aobs      <- sample(1:nNet, round(0.3*nNet)) # We observed 30%
# We can estimate rho using the gml function from the stats package
logestim  <- glm(ynet[Aobs] ~ -1 + Xnet[Aobs,], family = binomial(link = "logit"))
slogestim <- summary(logestim)
rho.est   <- logestim$coefficients
```

```r
rho.var   <- slogestim$cov.unscaled # we also need the covariance of the estimator
```

We assume that the explanatory variables X_{i1} and X_{i2} are observed for any i in the sample. Using the estimator of ρ_0, ρ_1, and ρ_2, we can also compute $\hat{\mathbf{P}}$, a consistent estimator of $\mathbf{P}$, from Equation (1).

```r
d.logit     <- lapply(1:M, function(x) {
  out       <- 1/(1 + exp(-rho.est[1] - rho.est[2]*X1.mat[[x]] -
                            rho.est[3]*X2.mat[[x]]))
  diag(out) <- 0
  out})
```

We can use $\hat{\mathbf{P}}$ for our SMM estimator. We focus on the case where neither $\mathbf{GX}$ nor $\mathbf{Gy}$ are observed. The same strategy can be used for other cases.

```r
smm.logit   <- smmSAR(y ~ X1 + X2, dnetwork = d.logit, contextual = T,
                      smm.ctr  = list(R = 100, print = F), data = dataset)
summary(smm.logit)
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2
##
## Contextual effects: Yes
## Fixed effects: No
##
## Network details
## GX Not Observed
## Gy Not Observed
## Number of groups: 100
## Sample size     : 3000
##
## Simulation settings
## R = 100
## Smoother : FALSE
##
## Coefficients:
##              Estimate Robust SE t value Pr(>|t|)
## Gy           0.398606  0.020494   19.45   <2e-16 ***
## (Intercept) -0.844648  1.816353   -0.47    0.642
## X1           0.991423  0.036141   27.43   <2e-16 ***
## X2           1.469934  0.047443   30.98   <2e-16 ***
## G: X1        4.936460  0.144670   34.12   <2e-16 ***
## G: X2       -2.520642  0.271198   -9.29   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The variance of the estimator computed above is conditionally on `d.logit`. It does not take into account the uncertainty related to the estimation of `d.logit`. To compute the right variance, we need a simulator from the distribution of `d.logit`. This simulator is a function which samples one network distribution from the distribution of `d.logit`. For instance, for the logit model, we can sample $\boldsymbol{\rho}$ from $N\left(\hat{\boldsymbol{\rho}}, \mathbb{V}(\hat{\boldsymbol{\rho}})\right)$ and then compute $\hat{\mathbf{P}}$. Depending on the network formation model, users can compute the adequate simulator.

```r
fdist       <- function(rho.est, rho.var, M, X1.mat, X2.mat){
  rho.est1   <- MASS::mvrnorm(mu = rho.est, Sigma = rho.var)
  lapply(1:M, function(x) {
```

```
  out        <- 1/(1 + exp(-rho.est1[1] - rho.est1[2]*X1.mat[[x]] -
                           rho.est1[3]*X2.mat[[x]]))
  diag(out)  <- 0
  out})
}
```

The function can be passed into the argument `.fun` of the `summary` method. We also need to pass the arguments of the simulator as a list into `.args`.

```
fdist_args  <- list(rho.est = rho.est, rho.var = rho.var, M = M, X1.mat = X1.mat,
                    X2.mat = X2.mat)
summary(smm.logit, dnetwork = d.logit, data = dataset, .fun = fdist, .args = fdist_args,
        sim = 500, ncores = 8) # ncores performs simulations in parallel
```

```
## Simulated Method of Moments estimation of SAR model
##
## Formula = y ~ X1 + X2
##
## Contextual effects: Yes
## Fixed effects: No
##
## Network details
## GX Not Observed
## Gy Not Observed
## Number of groups: 100
## Sample size     : 3000
##
## Simulation settings
## R = 100
## Smoother : FALSE
##
## Coefficients:
##              Estimate Robust SE t value Pr(>|t|)
## Gy           0.398606  0.023085   17.27   <2e-16 ***
## (Intercept) -0.844648  1.939298   -0.44    0.663
## X1           0.991423  0.038220   25.94   <2e-16 ***
## X2           1.469934  0.048514   30.30   <2e-16 ***
## G: X1        4.936460  0.176471   27.97   <2e-16 ***
## G: X2       -2.520642  0.289902   -8.69   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can notice that the variance is larger when we take into account the uncertainty of the estimation of the logit model.

3 Bayesian estimator

The Bayesian estimator is neatly packed in the function `mcmcSAR` (see the help page of the function in the package, using `?mcmcSAR`, for more details on the function).

3.1 without network formation model

For the sake of the example, we assume that linking probabilities are *known* and drawn from an uniform distribution. We first simulate data. Then, we estimate the linear-in-means model using our Bayesian estimator.

In the following example (example I-1, output `out.none1`), we assume that the network is entirely observed.

We first simulate data.

```r
library(PartialNetwork)
set.seed(123)
# EXAMPLE I: WITHOUT NETWORK FORMATION MODEL
# Number of groups
M             <- 50
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(2,1,1.5)
# contextual effects
gamma         <- c(5,-3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# network distribution
distr         <- runif(sum(N*(N-1)))
distr         <- vec.to.mat(distr, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(distr)
# normalize
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                  theta = c(alpha, beta, gamma, se))
y             <- y$y
# dataset
dataset       <- as.data.frame(cbind(y, X1 = X[,1], X2 = X[,2]))
```

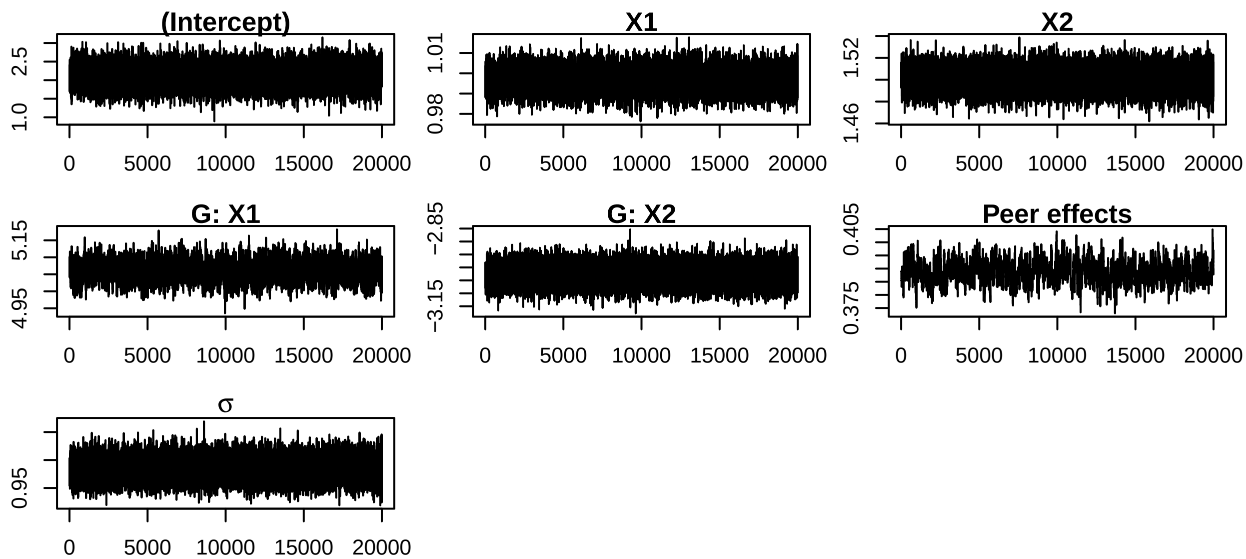
Once the data are simulated, the estimation can be performed using the function `mcmcSAR`.

```r
# Example I-1: When the network is fully observed
out.none1     <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                         G0 = G0, data = dataset, iteration = 2e4)
summary(out.none1)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 100%
## Network formation model: none
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
```

```
##
## Outcome model
##                     Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept)    2.1084419 0.270707848  1.5826644  2.6400052    +
## X1             0.9969651 0.005095668  0.9868044  1.0069689    +
## X2             1.5007138 0.009647679  1.4817806  1.5194103    +
## G: X1          5.0618683 0.027087144  5.0083412  5.1150957    +
## G: X2         -3.0280018 0.036370779 -3.0992659 -2.9572279    -
## Peer effects   0.3887471 0.004172086  0.3806533  0.3968838    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 0.9842002
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.44065
```

```
plot(out.none1, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```



For Example I-2, we assume that only 60% of the links are observed.

```
# Example I-2: When a part of the network is observed
# 60% of the network data is observed
G0.obs       <- lapply(N, function(x) matrix(rbinom(x^2, 1, 0.6), x))
```

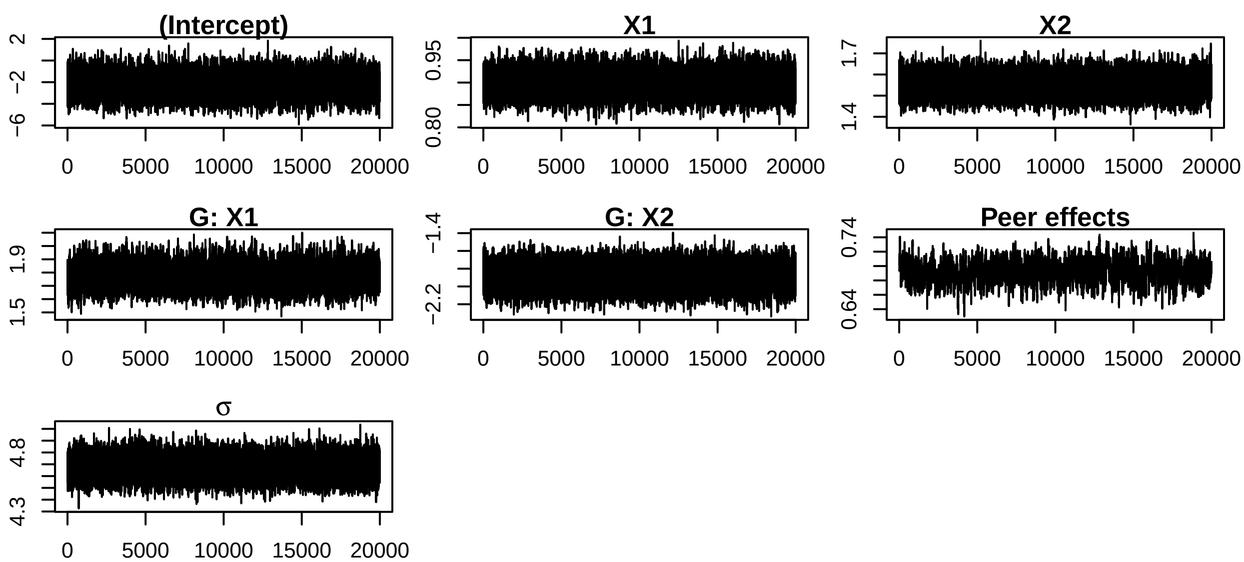
Estimation `out.none2.1` assumes that the sampled network is the true one (inconsistent, peer effects are overestimated).

```
# replace the non-observed part of the network by 0 (missing links)
G0.start     <- lapply(1:M, function(x) G0[[x]]*G0.obs[[x]])
# Use network with missing data as the true network
out.none2.1  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                        G0 = G0.start,   data = dataset, iteration = 2e4)
summary(out.none2.1) # the peer effets seem overestimated
```

```
## Bayesian estimation of SAR model
##
```

```
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 100%
## Network formation model: none
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
##                    Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)  -2.1618686 0.98109671 -4.0888701 -0.2357028    -
## X1            0.8981056 0.02441735  0.8502003  0.9464940    +
## X2            1.5538011 0.04593390  1.4629160  1.6431383    +
## G: X1         1.7806811 0.08282075  1.6189351  1.9423661    +
## G: X2        -1.8902743 0.12318796 -2.1341919 -1.6507850    -
## Peer effects  0.6918847 0.01614648  0.6601531  0.7221453    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 4.67309
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4317
```

```
plot(out.none2.1, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```

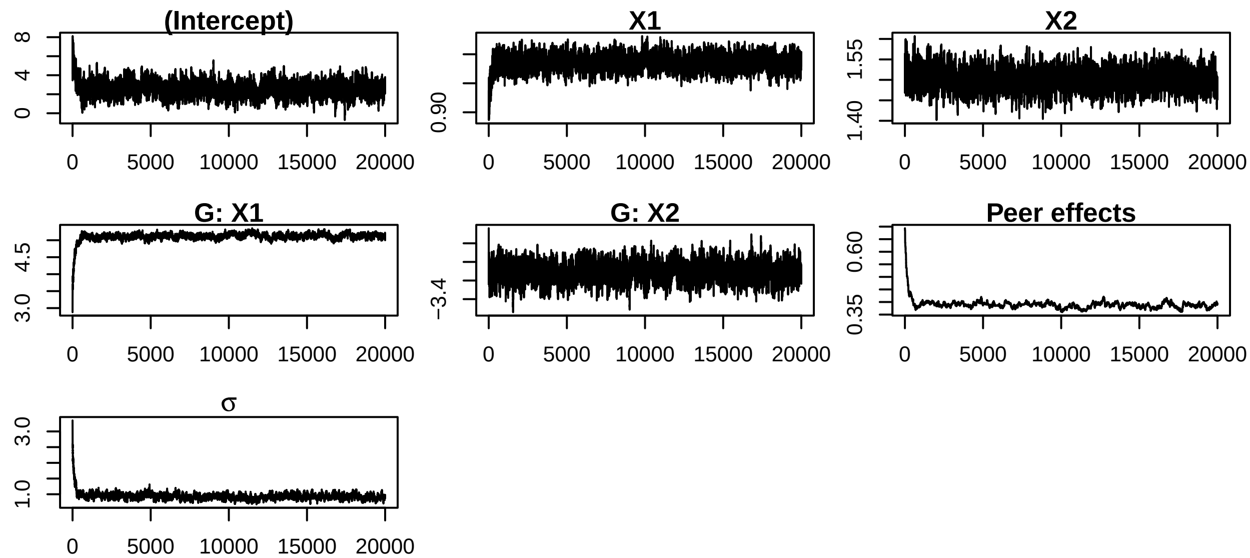


Estimation `out.none2.2` specifies which links are observed and which ones are not. The true probabilities are used to sample un-observed links (consistent).

```
out.none2.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = G0.obs,
                        G0 = G0.start, data = dataset,
                        mlinks = list(dnetwork = distr), iteration = 2e4)
summary(out.none2.2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 60.03448%
## Network formation model: none
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
##                   Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)   2.4958479 0.69651708  1.1226662  3.8387679    +
## X1            0.9856616 0.01260015  0.9613221  1.0106794    +
## X2            1.5010658 0.02297712  1.4568985  1.5459223    +
## G: X1         5.1350345 0.06840032  5.0008082  5.2673527    +
## G: X2        -3.1077379 0.09488307 -3.2872052 -2.9191241    -
## Peer effects  0.3854260 0.01063395  0.3659470  0.4072424    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 0.9138584
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4353
```

```
plot(out.none2.2, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```



For Example I-3, we assume that only linking probabilities are known.

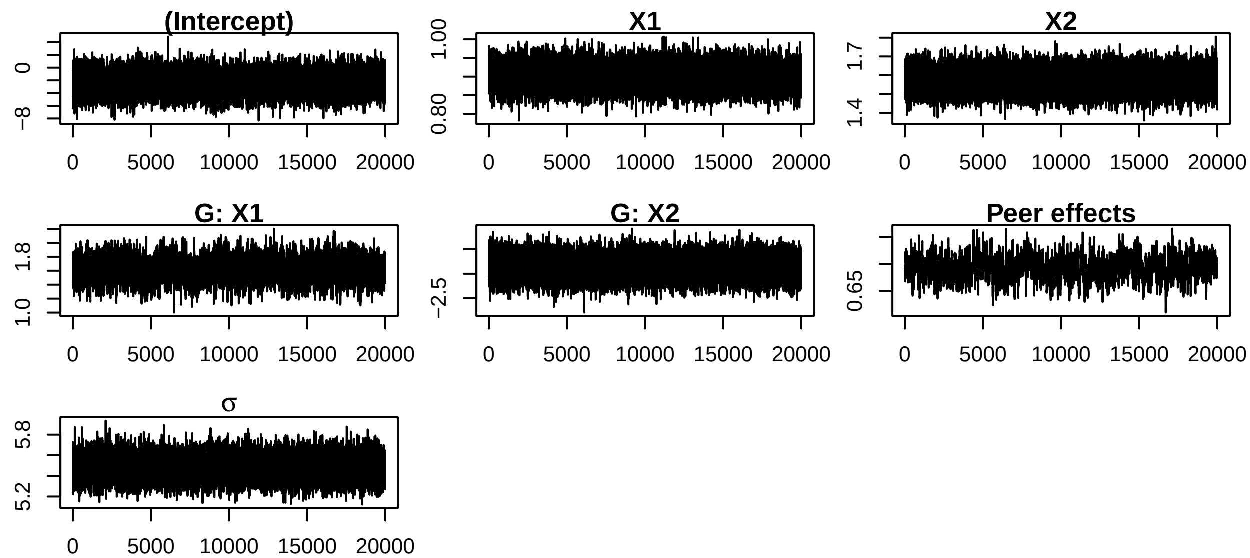
Estimation `out.none3.1` assumes the researcher uses a draw from that distribution as the true network (inconsistent, peer effects are overestimated).

```
# Example I-3: When only the network distribution is available
# Simulate a fictitious network and use as true network
G0.tmp       <- sim.network(distr)
out.none3.1  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                        G0 = G0.tmp, data = dataset, iteration = 2e4)
summary(out.none3.1)  # the peer effets seem overestimated
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 100%
## Network formation model: none
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
##                    Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)  -2.3651528 1.52951840 -5.3173803  0.6195651
## X1            0.9006007 0.02901363  0.8439843  0.9579893    +
## X2            1.5728392 0.05401973  1.4654145  1.6781822    +
## G: X1         1.6090619 0.14837791  1.3230986  1.9003541    +
## G: X2        -1.8691793 0.20357252 -2.2682623 -1.4670086    -
## Peer effects  0.6933608 0.02093661  0.6519094  0.7332266    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
```

```
##
## Error standard-deviation: 5.477948
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.44445
```

```r
plot(out.none3.1, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```



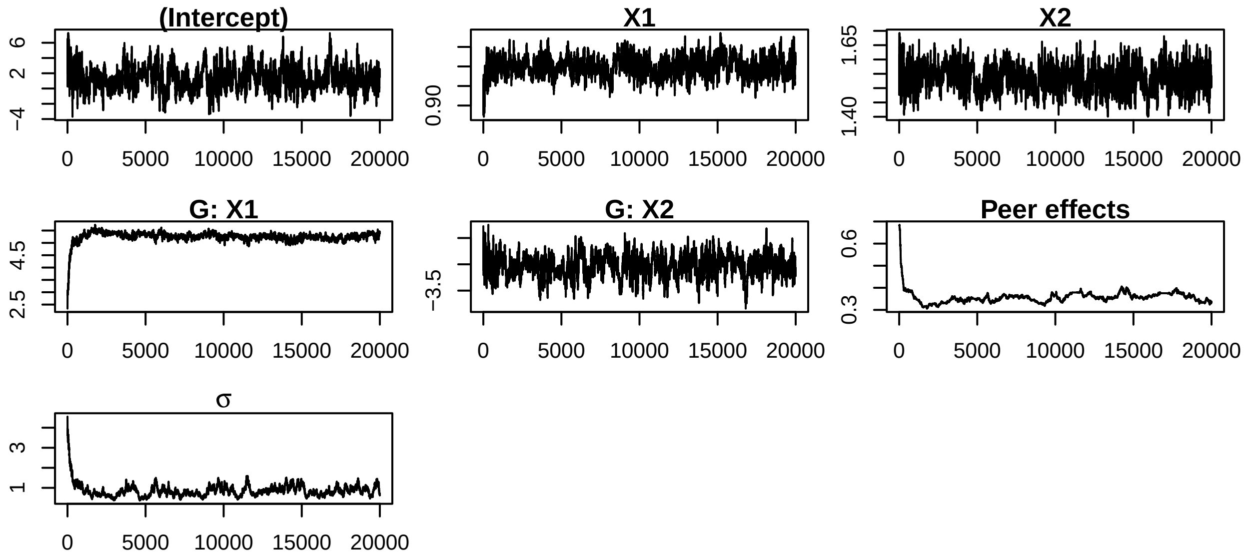
Estimation out.none3.2 specifies that no link is observed, but that the distribution is known (consistent).

```r
out.none3.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        data = dataset, mlinks = list(dnetwork = distr), iteration = 2e4)
summary(out.none3.2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 0%
## Network formation model: none
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
##                     Mean  Std.Error     Inf CI    Sup CI Sign
## (Intercept)    1.4277749 1.42900704 -1.1026970  4.643800
## X1             0.9989731 0.02375023  0.9552666  1.045906    +
## X2             1.5315158 0.04308283  1.4466932  1.614467    +
## G: X1          5.2253950 0.09951337  5.0163979  5.403797    +
## G: X2         -3.0262493 0.19945934 -3.4802317 -2.669676    -
## Peer effects   0.3633954 0.01522105  0.3333313  0.394371    +
```

```
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 0.8788236
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.42995
```

```
plot(out.none3.2, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```



3.2 With logit model as network formation model

For this example, we assume that links are generated using a simple logit model. We do not observe the true distribution.

We first simulate data.

```
# EXAMPLE II: NETWORK FORMATION MODEL: LOGIT
library(PartialNetwork)
set.seed(123)
# Number of groups
M             <- 50
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(2,1,1.5)
# contextual effects
gamma         <- c(5,-3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 2
# parameters of the network formation model
rho           <- c(-2, -.5, .2)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
```

```r
# compute distance between individuals
tmp           <- c(0, cumsum(N))
X1l           <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],1])
X2l           <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],2])
dist.net      <- function(x, y) abs(x - y)
X1.mat        <- lapply(1:M, function(m) {
  matrix(kronecker(X1l[[m]], X1l[[m]], FUN = dist.net), N[m])})
X2.mat        <- lapply(1:M, function(m) {
  matrix(kronecker(X2l[[m]], X2l[[m]], FUN = dist.net), N[m])})
# true network
Xnet          <- as.matrix(cbind("Const" = 1,
                                 "dX1"   = mat.to.vec(X1.mat),
                                 "dX2"   = mat.to.vec(X2.mat)))
ynet          <- Xnet %*% rho
ynet          <- 1*((ynet + rlogis(length(ynet))) > 0)
G0            <- vec.to.mat(ynet, N, normalise = FALSE)
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                   theta = c(alpha, beta, gamma, se))
y             <- y$y
# dataset
dataset       <- as.data.frame(cbind(y, X1 = X[,1], X2 = X[,2]))
```

For example II-1, we assume that the researcher only **observes 60% of the links**, but **know that the network formation model is logistic**.

```r
# Example II-1: When a part of the network is observed
# 60% of the network data is observed
G0.obs       <- lapply(N, function(x) matrix(rbinom(x^2, 1, 0.6), x))
# replace the non-observed part of the network by 0
G0.start     <- lapply(1:M, function(x) G0[[x]]*G0.obs[[x]])
# Infer the missing links in the network data
mlinks       <- list(model = "logit", mlinks.formula = ~ dX1 + dX2,
                     mlinks.data = as.data.frame(Xnet))
```

Once the data are simulated, the estimation can be performed.

```r
out.logi2.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = G0.obs,
                        G0 = G0.start, data = dataset, mlinks = mlinks,
                        iteration = 2e4)
summary(out.logi2.2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 59.86437%
## Network formation model: logit
## Formula = ~dX1 + dX2
##
## Network sampling
```

```
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept) -1.9931965 0.040442463 -2.0703130 -1.9105883    -
## dX1         -0.4979367 0.010932594 -0.5187390 -0.4767044    -
## dX2          0.1880581 0.007507114  0.1729477  0.2027337    +
##
## Outcome model
##                    Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept)   1.8957994 0.214243106  1.4777867  2.3253420    +
## X1            1.0002262 0.014071404  0.9724657  1.0278364    +
## X2            1.4872727 0.027027509  1.4335510  1.5392856    +
## G: X1         5.0374247 0.032011917  4.9748985  5.0990331    +
## G: X2        -2.9619437 0.016299332 -2.9940291 -2.9301698    -
## Peer effects  0.3944225 0.004362122  0.3860250  0.4033238    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 2.063969
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.438
## rho acceptance rate         : 0.27185
```

Example II-2 disregards the information about observed links (which we used to estimate the logit model) and only uses the asymptotic distribution of the network formation parameters.

```r
# Example II-2: When only the network distribution is available
# Infer the network data
# We only provide estimate of rho and its variance
Gvec         <- mat.to.vec(G0, ceiled = TRUE)
logestim     <- glm(Gvec ~ -1 + Xnet, family = binomial(link = "logit"))
slogestim    <- summary(logestim)
estimates    <- list("rho"     = logestim$coefficients,
                     "var.rho" = slogestim$cov.unscaled,
                     "N"       = N)
mlinks       <- list(model = "logit", mlinks.formula = ~ dX1 + dX2,
                     mlinks.data = as.data.frame(Xnet), estimates = estimates)
out.logi3.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        data = dataset, mlinks = mlinks, iteration = 2e4)
summary(out.logi3.2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 0%
## Network formation model: logit
```

```
## Formula = ~dX1 + dX2
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept) -1.9858238 0.042329799 -2.0689256 -1.9037267    -
## dX1         -0.5010106 0.010957459 -0.5225418 -0.4797864    -
## dX2          0.1911830 0.008126942  0.1756805  0.2071377    +
##
## Outcome model
##                    Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept)   1.9729428 0.298396335  1.3931863  2.5588524    +
## X1            0.9995212 0.015258206  0.9701962  1.0298869    +
## X2            1.4744075 0.040937768  1.3930613  1.5527800    +
## G: X1         5.1232632 0.050531818  5.0198489  5.2214341    +
## G: X2        -2.9802447 0.023855733 -3.0270739 -2.9344943    -
## Peer effects  0.3867595 0.006831039  0.3735510  0.4005997    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 1.999493
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4446
## rho acceptance rate         : 0.27905
```

Remarks

In the previous example, partial network information is available to estimate `rho` and `var.rho`. These estimates are included in the object `estimates`. Then in the MCMC, we set `G0.obs = "none"`, which means that the entire network will be inferred. In this situation, the posterior distribution of `rho` is strongly linked to the prior given by the practitioner, i.e. a normal distribution of parameters, the initial estimates of `rho` and `var.rho`. Indeed, an additional source of identification of the posterior distribution of `rho` may come from the spatial autoregressive (SAR) model. However, the available information in the observed part of the network is not used to update `rho` since it is already used when computing `estimates`.

From a certain point of view, inferring the observed part of the network can be considered inefficient. It is possible to keep fixed the observed entries of the adjacency matrix. As in example II-1, it is sufficient to set `G0.obs = G0.obs` instead of `G0.obs = "none"` in the function `mcmcSAR`. However, the observed part of the network is assumed to be non-stochastic in this case and will not be used to update `rho`. As soon as the practitioner gives an initial estimate of `rho` and `var.rho` in `estimates`, no information from the observed part of the network is used to update `rho`. The initial estimate of `rho` and `var.rho` summarizes the information.

We present an example below.

```
estimates    <- list("rho"     = logestim$coefficients,
                     "var.rho" = slogestim$cov.unscaled)
mlinks       <- list(model = "logit", mlinks.formula = ~ dX1 + dX2,
                     mlinks.data = as.data.frame(Xnet), estimates = estimates)
out.logi4.1  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = G0.obs,
```

```r
                        G0 = G0.start, data = dataset, mlinks = mlinks,
                        iteration = 2e4)
summary(out.logi4.1)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 59.86437%
## Network formation model: logit
## Formula = ~dX1 + dX2
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept) -2.0092988 0.042661052 -2.0925217 -1.9238912    -
## dX1         -0.5034023 0.011988619 -0.5268789 -0.4808867    -
## dX2          0.1941234 0.007959299  0.1783828  0.2101409    +
##
## Outcome model
##                   Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)   1.9158118 0.21163386  1.5032696  2.3329715    +
## X1            1.0002418 0.01402774  0.9727381  1.0272120    +
## X2            1.4848439 0.02683932  1.4310721  1.5373504    +
## G: X1         5.0406959 0.03218391  4.9784026  5.1045492    +
## G: X2        -2.9636788 0.01626540 -2.9953130 -2.9317992    -
## Peer effects  0.3939556 0.00447481  0.3851044  0.4025636    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 2.062611
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4381
## rho acceptance rate         : 0.261
```

One can notice that the network formation model estimate is not too different from the initial estimate of rho.

```r
print(slogestim)
```

```
##
## Call:
## glm(formula = Gvec ~ -1 + Xnet, family = binomial(link = "logit"))
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
```

```
## XnetConst -2.015140   0.047964  -42.01   <2e-16 ***
## XnetdX1   -0.504533   0.013239  -38.11   <2e-16 ***
## XnetdX2    0.196443   0.008922   22.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 60304  on 43500  degrees of freedom
## Residual deviance: 13652  on 43497  degrees of freedom
## AIC: 13658
##
## Number of Fisher Scoring iterations: 8
```

It is also possible to update `rho` in the MCMC using the observed part of the network. A particular case is Example II-1, where the distribution of `rho` is not given by the practitioner. In this example, the observed part of the network is used to estimate `rho` and `var.rho` (as in Example II-2). These estimates are then used as the prior distribution in the MCMC and `rho` is updated using the available information in the network. If the practitioner wants to define another prior, they can use `prior` instead of `estimate`.

This is an example where we assume that the prior distribution of `rho` is the standard normal distribution.

```
prior        <- list("rho"     = c(0, 0, 0),
                     "var.rho" = diag(3))
mlinks       <- list(model = "logit", mlinks.formula = ~ dX1 + dX2,
                     mlinks.data = as.data.frame(Xnet), prior = prior)
out.logi4.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = G0.obs,
                        G0 = G0.start, data = dataset, mlinks = mlinks,
                        iteration = 2e4)
summary(out.logi4.2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 59.86437%
## Network formation model: logit
## Formula = ~dX1 + dX2
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##                   Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept) -1.9927978 0.06411828 -2.1167138 -1.8681492    -
## dX1         -0.4983036 0.01458258 -0.5295421 -0.4707847    -
## dX2          0.1880291 0.01139670  0.1641840  0.2096634    +
##
## Outcome model
##                    Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept)   1.9090108 0.211424517  1.4985943  2.3336822    +
## X1            1.0003289 0.014143207  0.9724134  1.0282466    +
```

```
## X2            1.4845801 0.026705375  1.4317157  1.5375678    +
## G: X1         5.0369480 0.033329710  4.9718236  5.1031692    +
## G: X2        -2.9619636 0.016066491 -2.9939222 -2.9304163    -
## Peer effects  0.3943393 0.004561521  0.3853677  0.4034199    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 2.062558
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.43555
## rho acceptance rate         : 0.2694
```

The MCMC performs well although the distribution of `rho` defined in `prior` is not true. This is because the observed part of the network is used to update `rho`. In contrast, the MCMC could be inconsistent if one defines `estimates` as `prior` because `rho` will not be updated using the observed entries in the adjacency matrix, but only using information from the outcome `y`.

3.3 With latent space model as network formation model

3.3.1 ARD, Breza et al. (2020)

We also offer a function of the estimator in Breza et al. (2020). We first simulate data. We then estimate the model's parameters assuming that the researcher only knows ARD. We present two examples, one for which we observe ARD for the entire population (Example 1) and one for which we observe ARD for only 70% of the population (Example 2).

The data is simulated following a procedure similar to the one in Breza et al. (2020).

```r
library(PartialNetwork)
set.seed(123)
# LATENT SPACE MODEL
N           <- 500
genzeta     <- 1
mu          <- -1.35
sigma       <- 0.37
K           <- 12     # number of traits
P           <- 3      # Sphere dimension
# ARD parameters
# Generate z (spherical coordinates)
genz        <- rvMF(N, rep(0,P))
# Generate nu  from a Normal(mu, sigma^2) (The gregariousness)
gennu       <- rnorm(N, mu, sigma)
# compute degrees
gend        <- N*exp(gennu)*exp(mu+0.5*sigma^2)*exp(logCpvMF(P,0) - logCpvMF(P,genzeta))
# Link probabilities
distr       <- sim.dnetwork(gennu, gend, genzeta, genz)
# Adjacency matrix
G           <- sim.network(distr)
# Generate vk, the trait location
genv        <- rvMF(K, rep(0, P))
# set fixed some vk  distant
genv[1,]    <- c(1, 0, 0)
genv[2,]    <- c(0, 1, 0)
```

```r
genv[3,]     <- c(0, 0, 1)
# eta, the intensity parameter
geneta       <- abs(rnorm(K, 2, 1))
# Build traits matrix
densityatz   <- matrix(0, N, K)
for(k in 1:K){
  densityatz[,k] <- dvMF(genz, genv[k,]*geneta[k])
}
trait        <- matrix(0, N, K)
NK           <- floor(runif(K, 0.8, 0.95)*colSums(densityatz)/apply(densityatz, 2, max))
for (k in 1:K) {
  trait[,k] <- rbinom(N, 1, NK[k]*densityatz[,k]/sum(densityatz[,k]))
}
# Build ADR
ARD          <- G %*% trait
# generate b
genb         <- numeric(K)
for(k in 1:K){
  genb[k]    <- sum(G[,trait[,k]==1])/sum(G)
}
```

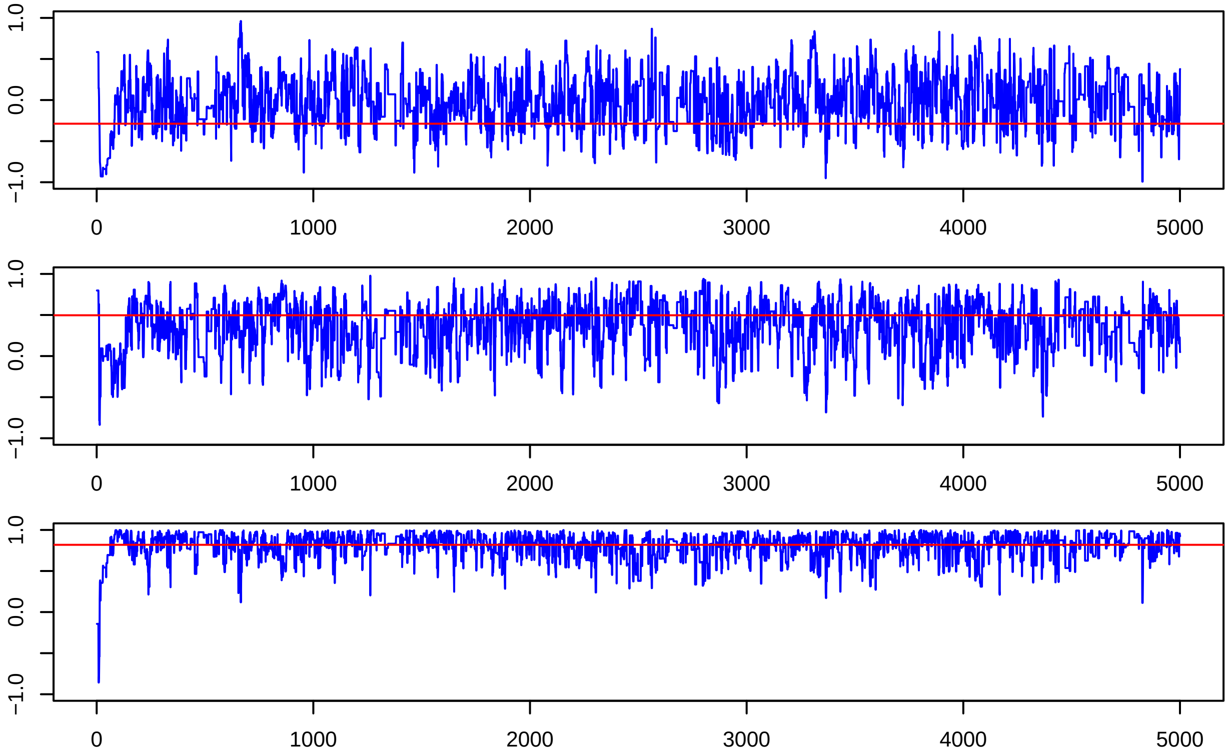
Example 1: we observe ARD for the entire population

```r
# Example1: ARD is observed for the whole population
# initialization
d0     <- exp(rnorm(N)); b0 <- exp(rnorm(K)); eta0 <- rep(1,K)
zeta0  <- 2; z0 <- matrix(rvMF(N, rep(0,P)), N); v0 <- matrix(rvMF(K,rep(0, P)), K)
# We should fix some vk and bk
vfixcolumn      <- 1:5
bfixcolumn      <- c(3, 7, 9)
b0[bfixcolumn]  <- genb[bfixcolumn]
v0[vfixcolumn,] <- genv[vfixcolumn,]
start           <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0,
                        "zeta" = zeta0)
```

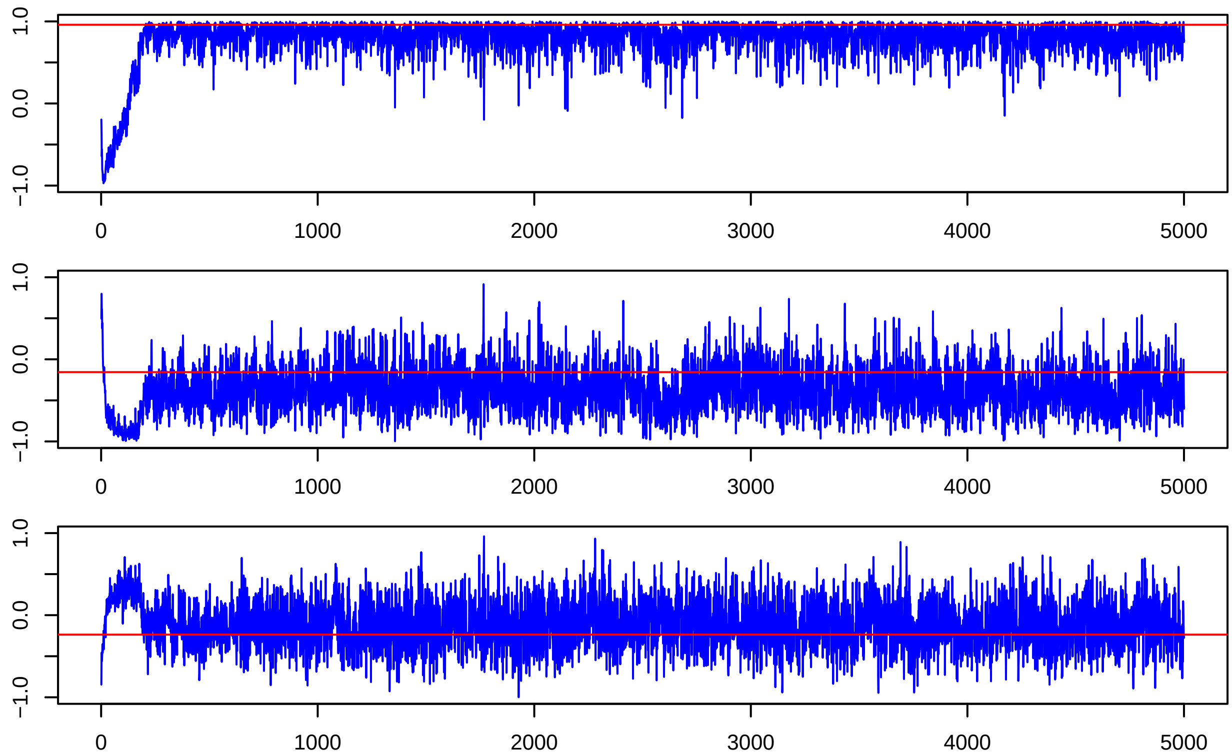
The estimation can be performed using the function mcmcARD

```r
# MCMC
estim.ard1      <- mcmcARD(Y = ARD, traitARD = trait, start = start, fixv = vfixcolumn,
                          consb = bfixcolumn, iteration = 5000)
```

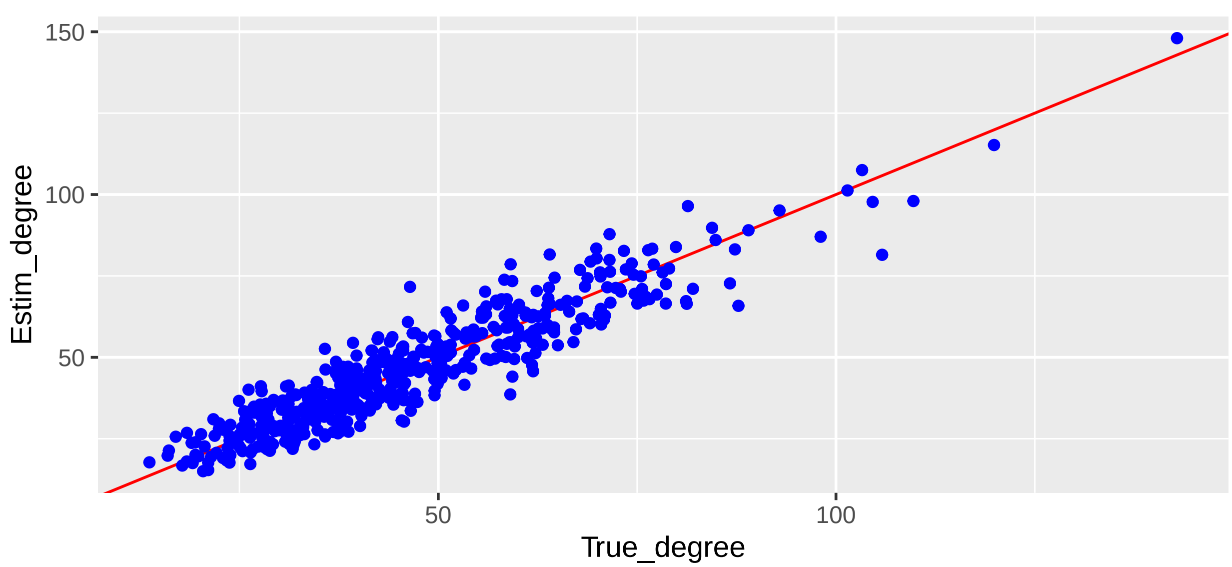
```r
# plot coordinates of individual 123
i      <- 123
zi     <- estim.ard1$simulations$z[i,,]
invisible(lapply(1:3, function(x) {
  plot(zi[x,], type = "l", ylab = "", col = "blue", ylim = c(-1, 1))
  abline(h = genz[i, x], col = "red")
}))
```




```
# plot coordinates of the trait 8
k     <- 8
vk    <- estim.ard1$simulations$v[k,,]
invisible(lapply(1:3, function(x) {
  plot(vk[x,], type = "l", ylab = "", col = "blue", ylim = c(-1, 1))
  abline(h = genv[k, x], col = "red")
}))
```




```r
# plot degree
library(ggplot2)
data.plot1 <- data.frame(True_degree  = gend,
                         Estim_degree = colMeans(tail(estim.ard1$simulations$d, 2500)))
ggplot(data = data.plot1, aes(x = True_degree, y = Estim_degree)) +
  geom_abline(col = "red") + geom_point(col = "blue")
```



Example 2: we observe ARD for only 70% of the population

```r
# Example2: ARD is observed for 70% population
# sample with ARD
n          <- round(0.7*N)
# individual with ARD
```

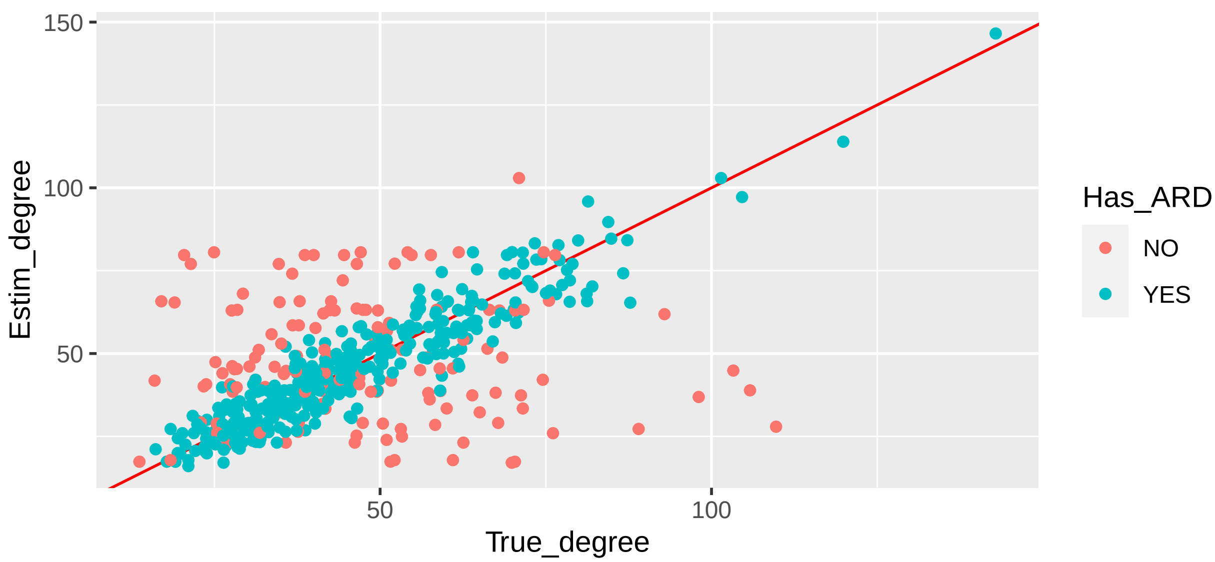
```r
iselect     <- sort(sample(1:N, n, replace = FALSE))
ARDs        <- ARD[iselect,]
traits      <- trait[iselect,]
# initialization
d0          <- d0[iselect]; z0 <- z0[iselect,]
start       <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0, "zeta" = zeta0)
```

The estimation can be performed using the function `mcmcARD`

```r
# MCMC
estim.ard2 <- mcmcARD(Y = ARDs, traitARD = traits, start = start, fixv = vfixcolumn,
                      consb = bfixcolumn, iteration = 5000)
```

```r
# estimation for non ARD
# we need a logical vector indicating if the i-th element has ARD
hasARD      <- (1:N) %in% iselect
# we use the matrix of traits to estimate distance between individuals
estim.nard2 <- fit.dnetwork(estim.ard2, X = trait, obsARD = hasARD, m = 1)
```

```r
# estimated degree
estd           <- estim.nard2$degree
data.plot2     <- data.frame(True_degree  = gend,
                             Estim_degree = estd,
                             Has_ARD      = ifelse(hasARD, "YES", "NO"))
ggplot(data = data.plot2, aes(x = True_degree, y = Estim_degree, colour = Has_ARD)) +
  geom_abline(col = "red") + geom_point()
```



3.3.2 Estimating peer effects model with ARD

Given the predicted probabilities, estimated using the estimator proposed by Breza et al. (2020) assuming that ARD are observed for the entire population, we implement our Bayesian estimator assuming that the posterior distribution of the linking probabilities are jointly normally distributed.

We first simulate data.

```r
library(PartialNetwork)
set.seed(123)
M              <- 30
```

```
N             <- rep(60, M)
genzeta       <- 3
mu            <- -1.35
sigma         <- 0.37
K             <- 12    # number of traits
P             <- 3     # Sphere dimension

# IN THIS LOOP, WE GENERATE DATA FOLLOWING BREZA ET AL. (2020) AND
# ESTIMATE THEIR LATENT SPACE MODEL FOR EACH SUB-NETWORK.
estimates     <- list()
list.trait    <- list()
G0            <- list()
for (m in 1:M) {
  #######################################################################################
  #######                            SIMULATION STAGE                           ########
  #######################################################################################
  # ARD parameters
  # Generate z (spherical coordinates)
  genz    <- rvMF(N[m], rep(0,P))
  # Generate nu  from a Normal(mu, sigma^2) (The gregariousness)
  gennu   <- rnorm(N[m],mu,sigma)
  # compute degrees
  gend    <- N[m]*exp(gennu)*exp(mu+0.5*sigma^2)*exp(logCpvMF(P,0) - logCpvMF(P,genzeta))
  # Link probabilities
  distr   <- sim.dnetwork(gennu, gend, genzeta, genz)
  # Adjacency matrix
  G        <- sim.network(distr)
  G0[[m]]  <- G
  # Generate vk, the trait location
  genv     <- rvMF(K, rep(0, P))
  # set fixed some vk  distant
  genv[1,] <- c(1, 0, 0)
  genv[2,] <- c(0, 1, 0)
  genv[3,] <- c(0, 0, 1)
  # eta, the intensity parameter
  geneta   <-abs(rnorm(K, 2, 1))
  # Build traits matrix
  densityatz       <- matrix(0, N[m], K)
  for(k in 1:K){
    densityatz[,k] <- dvMF(genz, genv[k,]*geneta[k])
  }
  trait       <- matrix(0, N[m], K)
  NK          <- floor(runif(K, .8, .95)*colSums(densityatz)/apply(densityatz, 2, max))
  for (k in 1:K) {
    trait[,k] <- rbinom(N[m], 1, NK[k]*densityatz[,k]/sum(densityatz[,k]))
  }
  list.trait[[m]]  <- trait
  # Build ADR
  ARD         <- G %*% trait
  # generate b
  genb        <- numeric(K)
  for(k in 1:K){
    genb[k]  <- sum(G[,trait[,k]==1])/sum(G) + 1e-8
```

```r
  }

  ########################################################################################
  #######                            ESTIMATION STAGE                          ########
  ########################################################################################
  # initialization
  d0     <- gend; b0 <- exp(rnorm(K)); eta0 <- rep(1,K); zeta0 <- genzeta
  z0     <- matrix(rvMF(N[m], rep(0,P)), N[m]); v0 <- matrix(rvMF(K,rep(0, P)), K)
  # We should fix some vk and bk
  vfixcolumn      <- 1:5
  bfixcolumn      <- c(1, 3, 5, 7, 9, 11)
  b0[bfixcolumn]  <- genb[bfixcolumn]
  v0[vfixcolumn,] <- genv[vfixcolumn,]
  start           <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0,
                          "zeta" = zeta0)
  estimates[[m]]  <- mcmcARD(Y = ARD, traitARD = trait, start = start, fixv = vfixcolumn,
                             consb = bfixcolumn, sim.d = FALSE, sim.zeta = FALSE,
                             iteration = 5000, ctrl.mcmc = list(print = FALSE))
}

# SIMULATE DATA FOR THE OUTCOME MODEL
# individual effects
beta          <- c(2,1,1.5)
# contextual effects
gamma         <- c(5,-3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# Normalise G0
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                  theta = c(alpha, beta, gamma, se))
y             <- y$y
# dataset
dataset       <- as.data.frame(cbind(y, X1 = X[,1], X2 = X[,2]))
```

Once the data are simulated, the estimation can be performed using the function `mcmcSAR`.

```r
mlinks       <- list(model = "latent space", estimates = estimates)
out.lspa1    <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        data = dataset, mlinks = mlinks, iteration = 2e4)
summary(out.lspa1)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
```

```
## Percentage of observed network data: 0%
## Network formation model: latent space
## Percentage of observed ARD: 100%
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
##                    Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)   0.9547959 1.41795429 -2.0688825  3.6448944
## X1            0.9842598 0.02422465  0.9345022  1.0296432    +
## X2            1.5323150 0.03951923  1.4559555  1.6079640    +
## G: X1         4.7051757 0.10486196  4.4619005  4.8740800    +
## G: X2        -2.8675601 0.20420531 -3.2623000 -2.4214902    -
## Peer effects  0.4204254 0.02301326  0.3844534  0.4607408    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 0.7831947
## Number of groups: 30
## Total sample size: 1800
##
## Peer effects acceptance rate: 0.43145
## rho acceptance rate         : 0.2697867
```

3.3.3 Estimating peer effects with partial ARD

Given the predicted probabilities, estimated using the estimator proposed by Breza et al. (2020) assuming that ARD are observed for 70% ∼ 100% of the population, we implement our Bayesian estimator assuming that the posterior distribution of the linking probabilities are jointly normally distributed.

We first simulate data.

```
library(PartialNetwork)
set.seed(123)
M             <- 30
N             <- rep(60, M)
genzeta       <- 3
mu            <- -1.35
sigma         <- 0.37
K             <- 12
P             <- 3

# IN THIS LOOP, WE GENERATE DATA FOLLOWING BREZA ET AL. (2020) AND
# ESTIMATE THEIR LATENT SPACE MODEL FOR EACH SUB-NETWORK.
estimates     <- list()
list.trait    <- list()
obARD         <- list()
G0            <- list()
for (m in 1:M) {
  #######################################################################################
  #######                           SIMULATION STAGE                           ########
  #######################################################################################
```

```r
# ARD parameters
# Generate z (spherical coordinates)
genz    <- rvMF(N[m], rep(0,P))
# Generate nu  from a Normal(mu, sigma^2) (The gregariousness)
gennu   <- rnorm(N[m],mu,sigma)
# compute degrees
gend    <- N[m]*exp(gennu)*exp(mu+0.5*sigma^2)*exp(logCpvMF(P,0) - logCpvMF(P,genzeta))
# Link probabilities
distr   <- sim.dnetwork(gennu, gend, genzeta, genz)
# Adjacency matrix
G        <- sim.network(distr)
G0[[m]]  <- G
# Generate vk, the trait location
genv     <- rvMF(K, rep(0, P))
# set fixed some vk  distant
genv[1,] <- c(1, 0, 0)
genv[2,] <- c(0, 1, 0)
genv[3,] <- c(0, 0, 1)
# eta, the intensity parameter
geneta   <-abs(rnorm(K, 2, 1))
# Build traits matrix
densityatz  <- matrix(0, N[m], K)
for(k in 1:K){
  densityatz[,k] <- dvMF(genz, genv[k,]*geneta[k])
}
trait       <- matrix(0, N[m], K)
NK          <- floor(runif(K, .8, .95)*colSums(densityatz)/apply(densityatz, 2, max))
for (k in 1:K) {
  trait[,k] <- rbinom(N[m], 1, NK[k]*densityatz[,k]/sum(densityatz[,k]))
}
list.trait[[m]]  <- trait
# Build ADR
ARD         <- G %*% trait
# generate b
genb        <- numeric(K)
for(k in 1:K){
  genb[k]   <- sum(G[,trait[,k]==1])/sum(G) + 1e-8
}
# sample with ARD
n          <- round(runif(1, .7, 1)*N[m])
# individual with ARD
iselect    <- sort(sample(1:N[m], n, replace = FALSE))
hasARD     <- (1:N[m]) %in% iselect
obARD[[m]] <- hasARD
ARDs       <- ARD[iselect,]
traits     <- trait[iselect,]
#######################################################################################
#######                          ESTIMATION STAGE                            ########
#######################################################################################
# initialization
d0         <- gend[iselect]; b0 <- exp(rnorm(K)); eta0 <- rep(1,K); zeta0 <- genzeta
z0         <- matrix(rvMF(n, rep(0,P)), n); v0 <- matrix(rvMF(K, rep(0, P)), K)
# We should fix some vk and bk
```

```r
  vfixcolumn     <- 1:5
  bfixcolumn     <- c(1, 3, 5, 7, 9, 11)
  b0[bfixcolumn] <- genb[bfixcolumn]; v0[vfixcolumn,] <- genv[vfixcolumn,]
  start          <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0,
                         "zeta" = zeta0)
  estimates[[m]] <- mcmcARD(Y = ARDs, traitARD = traits, start = start, fixv = vfixcolumn,
                           consb = bfixcolumn,  sim.d = FALSE, sim.zeta = FALSE,
                           iteration = 5000, ctrl.mcmc = list(print = FALSE))
}

# SIMULATE DATA FOR THE OUTCOME MODEL
# individual effects
beta           <- c(2,1,1.5)
# contextual effects
gamma          <- c(5,-3)
# endogenous effects
alpha          <- 0.4
# std-dev errors
se             <- 1
# covariates
X              <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# Normalise G0
G0norm         <- norm.network(G0)
# simulate dependent variable use an external package
y              <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                   theta = c(alpha, beta, gamma, se))
y              <- y$y
# dataset
dataset        <- as.data.frame(cbind(y, X1 = X[,1], X2 = X[,2]))
```

Once the data are simulated, the estimation can be performed using the function `mcmcSAR`.

```r
mlinks       <- list(model = "latent space", estimates = estimates,
                     mlinks.data = list.trait, obsARD = obARD)
out.lspa2    <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        data = dataset, mlinks = mlinks, iteration = 2e4)
summary(out.lspa2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 0%
## Network formation model: latent space
## Percentage of observed ARD: 84.27778%
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
```

```
##                    Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)   0.5927081 2.67866330 -4.6527320  5.7171349
## X1            1.0745350 0.03283973  1.0119450  1.1395640    +
## X2            1.5285286 0.05537676  1.4197317  1.6344581    +
## G: X1         4.1053862 0.21814690  3.6706579  4.5504417    +
## G: X2        -2.7046589 0.36336002 -3.4192060 -2.0005901    -
## Peer effects  0.4610160 0.02918803  0.3994566  0.5149815    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 3.779413
## Number of groups: 30
## Total sample size: 1800
##
## Peer effects acceptance rate: 0.44075
## rho acceptance rate         : 0.2705733
```

4 The selection bias issue

In many applications, missing links are not completely random. For example, an individual who has no friends is less likely to have missing value on their row in the adjacency matrix than someone who has a lot of links. Even regressing a logit/probit network formation model using the entries of the adjacency matrix that are correctly measured will not yield a consistent estimator due to the selection bias issue.

Consider the case of missing links due to unmatched declared friends. This is for example the case of the network data collected by the National Longitudinal Study of Adolescent to Adult Health (Add Health). We know the *true* number of links for each individual. This information is crucial to address the selection bias issue. When an individual has missing links, we could doubt all the "zeros" on their row in the adjacency matrix. To consistently estimate the SAR model, we can simply assume that information from this row is not true, i.e, the row has "zeros" everywhere in the object `G0.obs`. Information from individuals without missing links can be used to estimate `rho` and doubtful rows in the adjacency matrix will be inferred. However, there is a selection bias issue because we do not use rows with missing links to estimate `rho`. These rows are likely to have a lot of `ones`. As a consequence, the network formation model estimating using rows with no unmatched friends will simulate fewer links than the true data-generating process (DGP).

A straightforward approach to address the issue is to weight the selected rows that are used to estimate `rho` (see Manski and Lerman (1977)). Every selected row must be weighted by the inverse of the probability of having no unmatched links. To do so, the practitioner can include an input `weights` in the list `mlinks` which is an argument of the function `mcmcsar`. The input `weights` must be a vector of dimension the number of "ones" in `G0.obs`.

We consider the following example where the number of missing links is randomly chosen between zero and the true number of links.

```r
library(PartialNetwork)
library(dplyr)
set.seed(123)
# Number of groups
M             <- 50
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(2, 1, 1.5)
# contextual effects
```

```r
gamma          <- c(5,-3)
# endogenous effects
alpha          <- 0.4
# std-dev errors
se             <- 2
# parameters of the network formation model
rho            <- c(-0.5, -.5, .4)
# covariates
X              <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# compute distance between individuals
tmp            <- c(0, cumsum(N))
X1l            <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],1])
X2l            <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],2])
dist.net       <- function(x, y) abs(x - y)
X1.mat         <- lapply(1:M, function(m) {
  matrix(kronecker(X1l[[m]], X1l[[m]], FUN = dist.net), N[m])})
X2.mat         <- lapply(1:M, function(m) {
  matrix(kronecker(X2l[[m]], X2l[[m]], FUN = dist.net), N[m])})
# true network
Xnet           <- as.matrix(cbind("Const" = 1,
                                 "dX1"   = mat.to.vec(X1.mat),
                                 "dX2"   = mat.to.vec(X2.mat)))
ynet           <- Xnet %*% rho
ynet           <- c(1*((ynet + rlogis(length(ynet))) > 0))
G0             <- vec.to.mat(ynet, N, normalise = FALSE)
# number of friends
nfriends       <- unlist(lapply(G0, function(x) rowSums(x)))
# number of missing links
nmislink       <- sapply(nfriends, function(x) sample(0:x, 1))
```

We now simulate the observed network by removing links from `G0` depending on the number of unmatched links. We also simulate the outcome `y` using the true network.

```r
Gobs           <- list(M)  # The observed network
G0.obs         <- list(M)  # Which information is true and doubtful
for(x in 1:M){
  Gx           <- G0[[x]]
  G0.obsx      <- matrix(1, N[x], N[x]); diag(G0.obsx) <- 0
  csum         <- cumsum(c(0, N))
  nmis         <- nmislink[(csum[x] + 1):csum[x + 1]]
  for (i in 1:N[x]) {
    if(nmis[i] > 0){
      tmp      <- which(c(Gx[i,]) == 1)
      if(length(which(c(Gx[i,]) == 1)) > 1) {
        tmp    <- sample(which(c(Gx[i,]) == 1), nmis[i])
      }
      Gx[i,tmp]    <- 0
      G0.obsx[i,] <- 0
    }
  }
  Gobs[[x]]    <- Gx
  G0.obs[[x]] <- G0.obsx
}
G0norm         <- norm.network(G0)
```

```r
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                  theta = c(alpha, beta, gamma, se))
y             <- y$y
# data set
dataset       <- as.data.frame(cbind(y, X1 = X[,1], X2 = X[,2]))
```

Assume that we estimate the network formation by selecting the individual with no unmatched links.

```r
mlinks        <- list(model = "logit", mlinks.formula = ~ dX1 + dX2,
                      mlinks.data = as.data.frame(Xnet))
out.selb1     <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0 = Gobs,
                         G0.obs = G0.obs, data = dataset, mlinks = mlinks,
                         iteration = 2e4)
summary(out.selb1)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 55.2%
## Network formation model: logit
## Formula = ~dX1 + dX2
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept) -2.2554473 0.053338525 -2.3625815 -2.1511715    -
## dX1         -0.5062865 0.014675852 -0.5350419 -0.4768971    -
## dX2          0.1885610 0.009838786  0.1696495  0.2076671    +
##
## Outcome model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept)   2.0197155 0.214213468  1.6082923  2.4357397    +
## X1            1.0100124 0.013975372  0.9821508  1.0372596    +
## X2            1.4983784 0.027151491  1.4443090  1.5511542    +
## G: X1         4.9516461 0.033324639  4.8874755  5.0172057    +
## G: X2        -2.9817629 0.016237918 -3.0137233 -2.9503623    -
## Peer effects  0.4059135 0.004523017  0.3969188  0.4147982    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 2.062936
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.44535
```

```
## rho acceptance rate         : 0.27225
```

Although the peer effect estimate appears to be correct, the network formation estimator is likely biased. Indeed the estimator of the intercept is -2.26 and is significantly lower than its actual value of -2. As a result, the network formation model is likely to simulate fewer links than the true DGP. This is an issue if the practitioner uses the model to simulate policy impact on the outcome y or network features, such as centrality.

To control for the selection issue, we can weight selected individuals. The weight depends on the framework, especially how missing links occur. In our example, the number of missing links is chosen randomly between zero and the number of declared links. If the individual has n_i declared friends, the probability of having no unmatched links can be estimated by the proportion of the number of individuals who has no unmatched links in the subset of individuals who declare n_i.

```r
G0.obsvec      <- as.logical(mat.to.vec(G0.obs))
Gvec           <- mat.to.vec(Gobs, ceiled = TRUE)[G0.obsvec]
W              <- unlist(data.frame(nfriends = nfriends, nmislink = nmislink) %>%
                        group_by(nfriends) %>%
                        summarise(w = length(nmislink)/sum(nmislink == 0)) %>%
                        select(w))
W              <- lapply(1:M, function(x){
  matrix(rep(W[rowSums(G0[[x]]) + 1], each = N[x]), N[x], byrow = TRUE)})
weights        <- mat.to.vec(W)[G0.obsvec]
mlinks         <- list(model = "logit", mlinks.formula = ~ dX1 + dX2,
                       mlinks.data = as.data.frame(Xnet), weights = weights)
out.selb2      <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0 = Gobs,
                          G0.obs = G0.obs, data = dataset, mlinks = mlinks,
                          iteration = 2e4)
summary(out.selb2)
```

```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 55.2%
## Network formation model: logit
## Formula = ~dX1 + dX2
##
## Network sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept) -1.9731148 0.032176581 -2.0391133 -1.9122950    -
## dX1         -0.4832792 0.008757863 -0.4999554 -0.4656951    -
## dX2          0.1859662 0.006253833  0.1738966  0.1982889    +
##
## Outcome model
##                   Mean   Std.Error     Inf CI     Sup CI Sign
## (Intercept)   1.9856192 0.214097653  1.5685888  2.4056519    +
## X1            1.0082459 0.013817963  0.9815052  1.0352525    +
## X2            1.5042882 0.027471214  1.4507508  1.5578727    +
```

```
## G: X1          4.9559929 0.033201835  4.8912507  5.0211436    +
## G: X2         -2.9850242 0.016120415 -3.0165679 -2.9532613    -
## Peer effects  0.4054351 0.004686395  0.3963591  0.4148558    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 2.034204
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4377
## rho acceptance rate         : 0.27595
```

It is also possible to use selected sample and the weight to estimate `rho` and `var.rho` using the function `glm`. These estimates can be provided to `mlinks` in `estimates`. In this case `rho` will not be updated using the observed part of the network.

References

Boucher, V. and Houndetoungan, A. (2022). *Estimating peer effects using partial network data.* Centre de recherche sur les risques les enjeux économiques et les politiques.

Bramoullé, Y., Djebbari, H., and Fortin, B. (2009). Identification of peer effects through social networks. *Journal of econometrics*, 150(1):41–55.

Breza, E., Chandrasekhar, A. G., McCormick, T. H., and Pan, M. (2020). Using aggregated relational data to feasibly identify network structure without network data. *American Economic Review*, 110(8):2454–84.

Manski, C. F. and Lerman, S. R. (1977). The estimation of choice probabilities from choice based samples. *Econometrica: Journal of the Econometric Society*, pages 1977–1988.