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

Vincent Boucher and Elysée Aristide Houndetoungan

2021-03-10

1 Instrumental Variable 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 replication, 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. The following code provides an example using a single group of 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            <- 2
# prior distribution
prior         <- runif(sum(N*(N-1)))
prior         <- vec.to.mat(prior, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(prior)
# normalise
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simSARnet(~ X, contextual = FALSE, Glist = G0norm,
                                     theta = c(alpha, beta, se))
y             <- y$y
# generate instruments
instr         <- sim.IV(prior, 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:

```
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
## -6.47278 -1.37743  0.04286  1.35378  6.85486
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.86704    0.81885   7.165 1.22e-12 ***
## X1           1.00862    0.01060  95.179  < 2e-16 ***
## X2           1.48062    0.01976  74.948  < 2e-16 ***
## G1y          0.21986    0.03864   5.690 1.53e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.033 on 1496 degrees of freedom
## Multiple R-Squared: 0.9086,  Adjusted R-squared: 0.9084
## Wald test:  4950 on 3 and 1496 DF,  p-value: < 2.2e-16
```

```
# 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
## -6.6664018 -1.3556028  0.0002022  1.3825419  6.7349762
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.75563    1.27785   1.374     0.17
## X1           1.01081    0.01066  94.863  < 2e-16 ***
## X2           1.47842    0.01985  74.491  < 2e-16 ***
## G1y          0.41719    0.06091   6.850 1.08e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.042 on 1496 degrees of freedom
## Multiple R-Squared: 0.9078,  Adjusted R-squared: 0.9077
```

```
## Wald test:  4912 on 3 and 1496 DF,  p-value: < 2.2e-16
```

2 Bayesian estimator without network formation model

The Bayesian estimator is neatly packed in the function `mcmcSAR`. Below, we provide a simple example using simulated data.

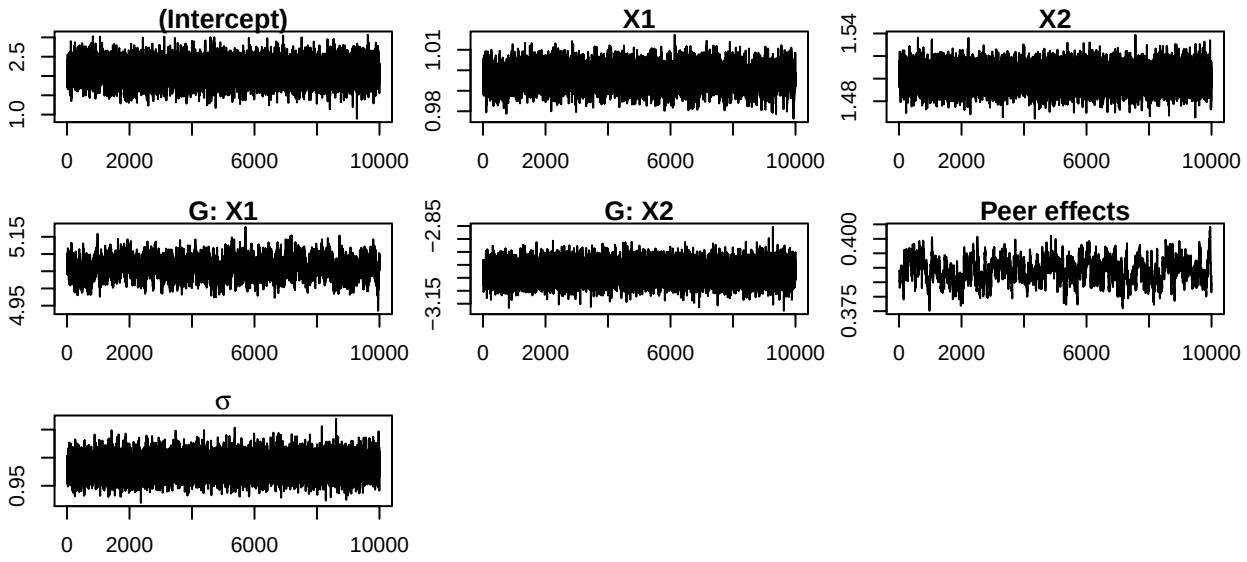
```
library(PartialNetwork)
set.seed(123)
# EXAMPLE I: WITHOUT NETWORK FORMATION MODEL
# Number of groups
M             <- 50
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(2,1,1.5)
# contextual effects
gamma         <- c(5,-3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# prior distribution
prior         <- runif(sum(N*(N-1)))
prior         <- vec.to.mat(prior, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(prior)
# normalize
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simSARnet(~ 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]))
# 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 = 1e4)
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  10000
## Elapsed time           :  0  HH  0  mm  19  ss
##
## Peer effects acceptance rate: 0.4408
```

```r
summary(out.none1)
```

```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 10000
## Burn-in: 5000
##
## 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)   2.1118016 0.271394727  1.5717356  2.6351136    +
## X1            0.9970161 0.005205190  0.9865430  1.0072497    +
## X2            1.5008418 0.009781627  1.4816417  1.5198629    +
## G: X1         5.0614928 0.027667969  5.0067518  5.1141512    +
## G: X2        -3.0282521 0.036447003 -3.0988105 -2.9569765    -
## Peer effects  0.3889236 0.004357742  0.3806162  0.3970502    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 0.9841574
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4408
```

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



```r
# 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))
# 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 = 1e4)
```

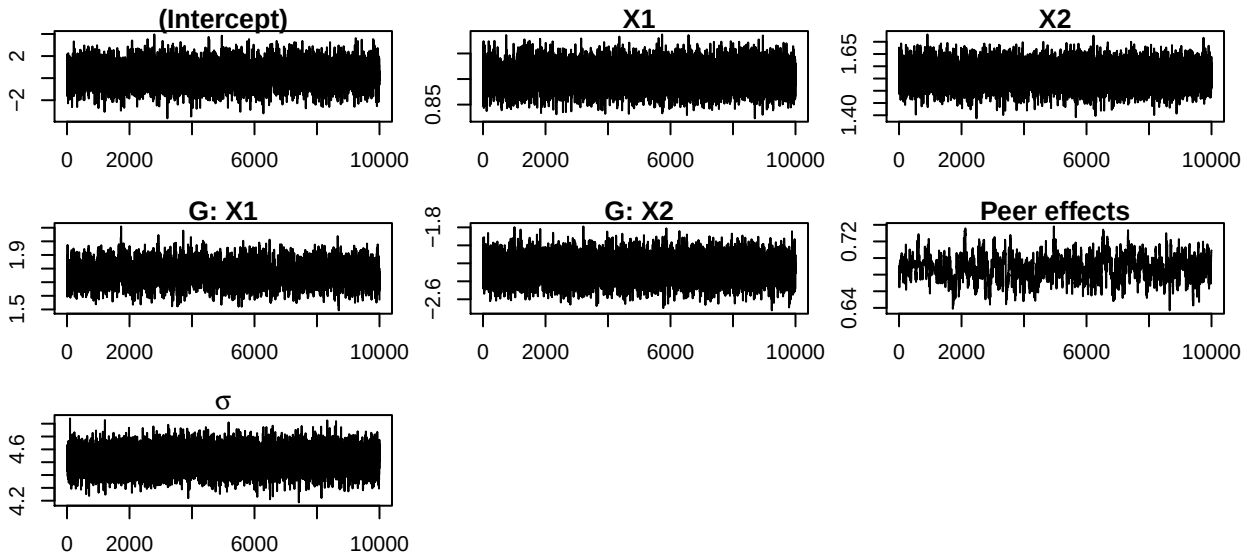
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  10000
## Elapsed time           :  0  HH  0  mm  19  ss
##
## Peer effects acceptance rate: 0.4279
```

```r
summary(out.none2.1) # the peer effets seem overestimated
```

```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 10000
## Burn-in: 5000
##
## 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)   0.3540124 0.97724818 -1.5376099  2.2696241
## X1            0.9055897 0.02345388  0.8598567  0.9518323    +
## X2            1.5624688 0.04408718  1.4757709  1.6481278    +
## G: X1         1.7656994 0.07626751  1.6170669  1.9150605    +
## G: X2        -2.2618536 0.12504432 -2.5074629 -2.0183899    -
## Peer effects  0.6888221 0.01472819  0.6589845  0.7169222    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 4.510032
## Number of groups: 50
## Total sample size: 1500
##
```

```
## Peer effects acceptance rate: 0.4279
```

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




```
out.none2.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = G0.obs,
                        G0 = G0.start,   data = dataset,
                        mlinks = list(dnetwork = prior), iteration = 1e4)
```

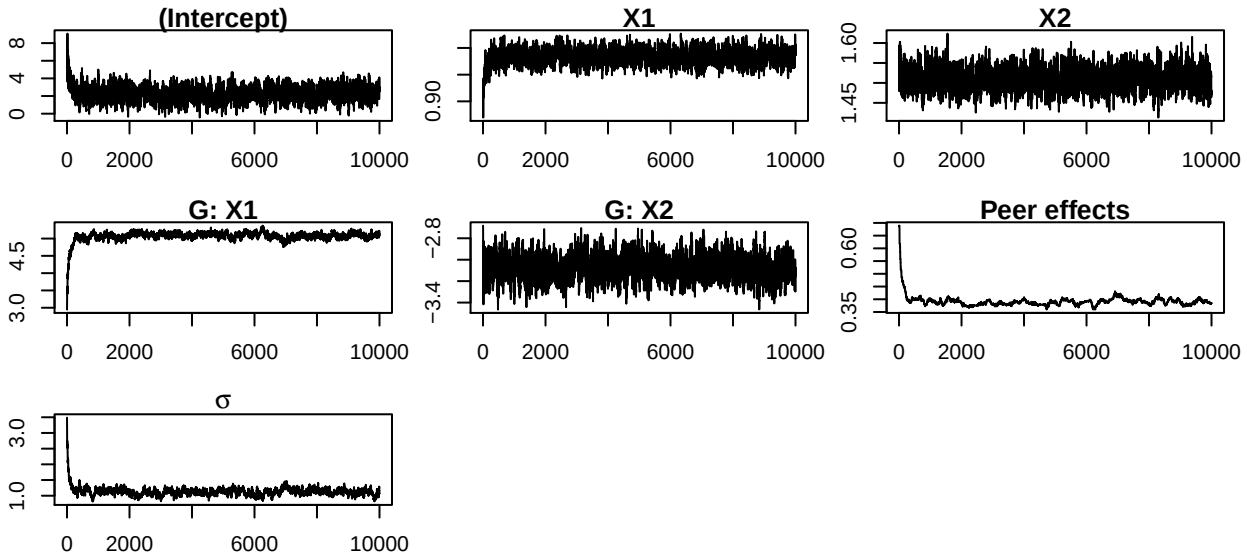
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY**************
## Number of group        :  50
## Iteration              :  10000
## Elapsed time           :  0  HH  14  mm  39  ss
##
## Peer effects acceptance rate: 0.4391
```

```
summary(out.none2.2)
```

```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 10000
## Burn-in: 5000
##
## Percentage of Observed Network Data: 59.93333%
## Network formation model: none
##
## Network Sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
```

```
##                    Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)   2.2619499 0.78056350  0.6939214  3.7656750    +
## X1            0.9830539 0.01329172  0.9579202  1.0099397    +
## X2            1.5121786 0.02655993  1.4606520  1.5639759    +
## G: X1         5.0790383 0.08039342  4.9102771  5.2320206    +
## G: X2        -3.0705144 0.10485527 -3.2734008 -2.8588156    -
## Peer effects  0.3914802 0.01111285  0.3717470  0.4159471    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 1.119265
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4391
```

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



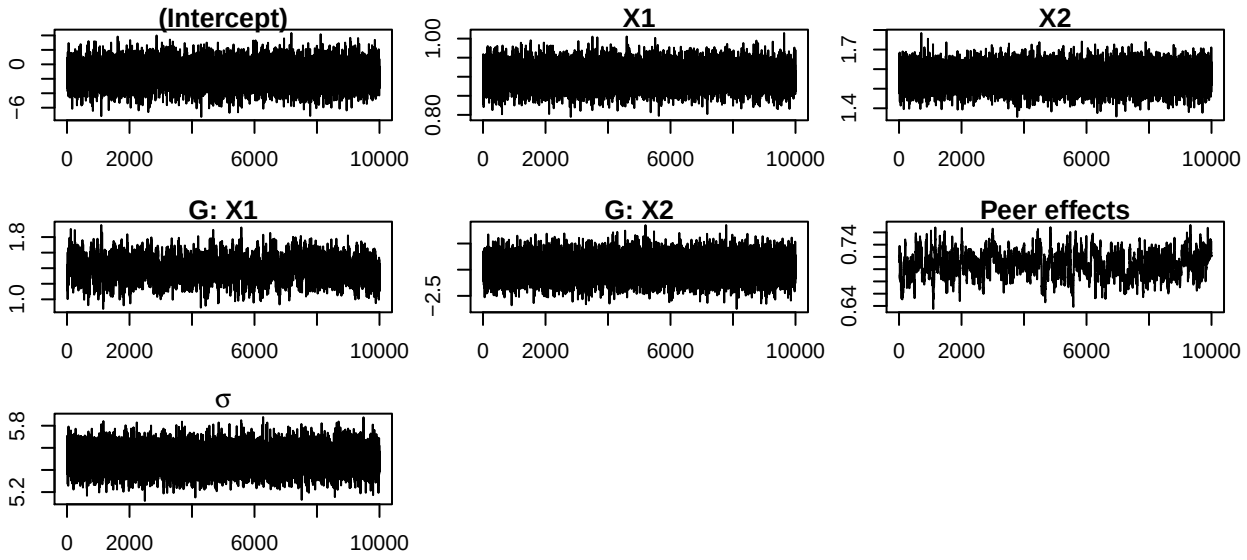

```
# Example I-3: When only the network distribution is available
# Simulate a fictitious network and use as true network
G0.tmp       <- sim.network(prior)
out.none3.1  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                        G0 = G0.tmp, data = dataset, iteration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

```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  10000
## Elapsed time           :  0  HH  0  mm  17  ss
##
## Peer effects acceptance rate: 0.4485
```

```r
summary(out.none3.1)  # the peer effets seem overestimated
```

```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 10000
## Burn-in: 5000
##
## 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)  -1.3953493 1.57920637 -4.5156518  1.7156168
## X1            0.9016752 0.02834729  0.8453817  0.9576844    +
## X2            1.5593761 0.05494652  1.4527582  1.6651459    +
## G: X1         1.3927847 0.14137802  1.1136030  1.6695040    +
## G: X2        -1.9650616 0.20938632 -2.3777562 -1.5449322    -
## Peer effects  0.7082433 0.02072956  0.6671947  0.7466054    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 5.497975
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4485
```

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



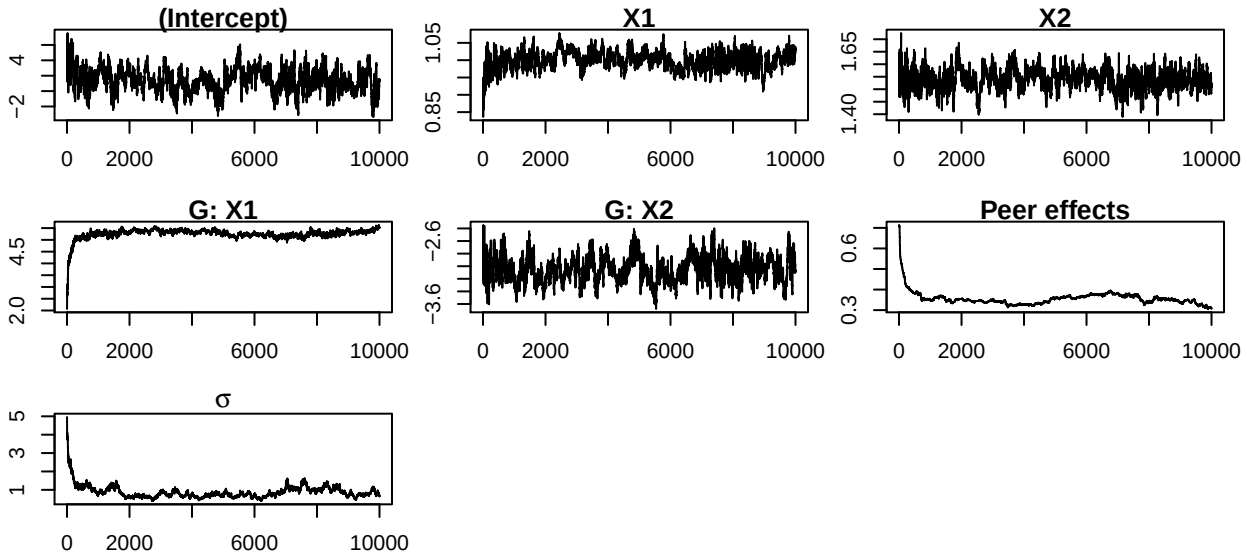
```r
out.none3.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        G0 = G0.tmp, data = dataset, mlinks = list(dnetwork = prior),
                        iteration = 1e4)
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  10000
## Elapsed time         :  0  HH  35  mm  13  ss
##
## Peer effects acceptance rate: 0.4398
```

```r
summary(out.none3.2)
```

```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 10000
## Burn-in: 5000
##
## 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)   1.4353650 1.46477033 -1.5503191  4.3337132
## X1            0.9974188 0.02360383  0.9503897  1.0407444    +
## X2            1.5380656 0.03819647  1.4592263  1.6113430    +
## G: X1         5.2457355 0.11176786  5.0382931  5.4842302    +
## G: X2        -3.0464637 0.20220364 -3.4406450 -2.6393638    -
## Peer effects  0.3568191 0.01844493  0.3133373  0.3858118    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 0.870009
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.4398
```

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



3 Bayesian estimator with logit model as network formation model

```
# 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))
# 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
covar         <- as.matrix(cbind("Const" = 1,
                                 "dX1"   = mat.to.vec(X1.mat),
                                 "dX2"   = mat.to.vec(X2.mat)))
ynet          <- covar %*% 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::simSARnet(~ 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]))
```

```
# 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
out.logi2.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = G0.obs,
                        G0 = G0.start,   data = dataset,
                        mlinks = list(model = "logit", covariates = covar),
                        iteration = 2e4)
```

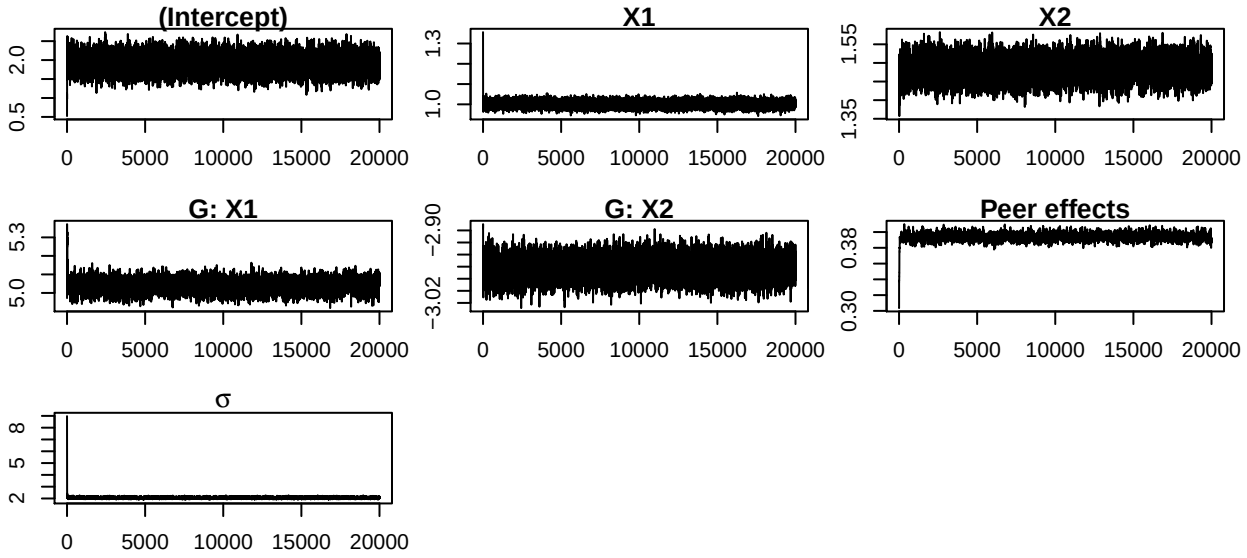
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  0  HH  31  mm  41  ss
##
## Peer effects acceptance rate: 0.438
## rho acceptance rate         : 0.27185
```

```
summary(out.logi2.2) # the peer effets seem overestimated
```

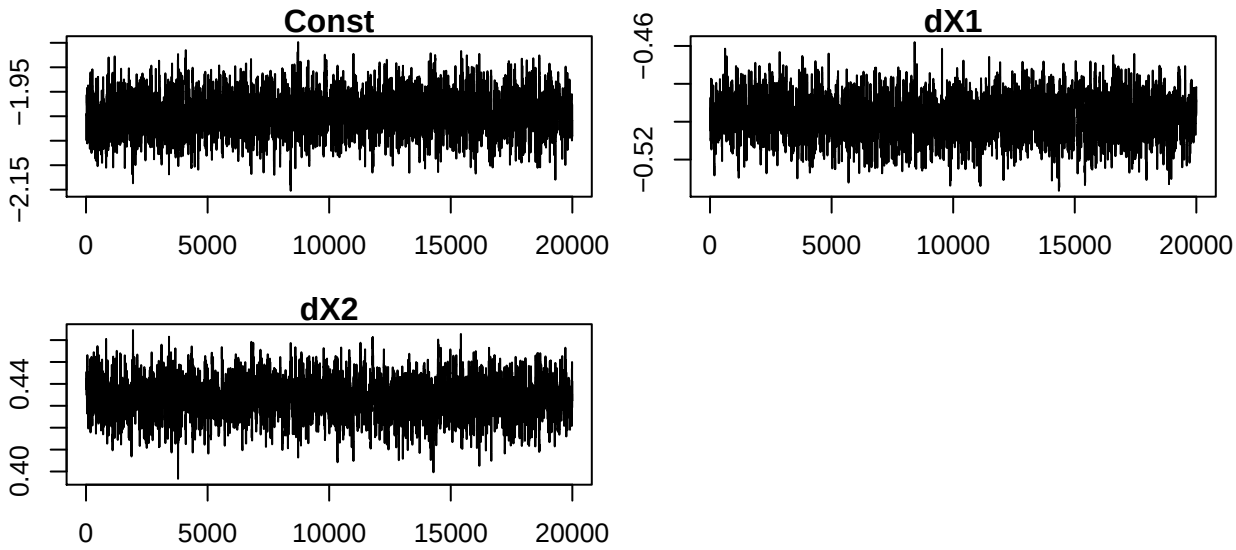
```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of Observed Network Data: 59.86437%
## Network formation model: logit
##
## Network Sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##             Mean   Std.Error     Inf CI     Sup CI Sign
## Const -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
```

```
plot(out.logi2.2, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```




```
plot(out.logi2.2, plot.type = "sim", which.parms = "rho", mar = c(3, 2.1, 1, 1))
```




```
# 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 + covar, family = binomial(link = "logit"))
slogestim    <- summary(logestim)
estimates    <- list("rho"     = logestim$coefficients,
                     "var.rho" = slogestim$cov.unscaled,
                     "N"       = N)
mlinks       <- list(model = "logit", covariates = covar, estimates = estimates)
out.logi3.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        data = dataset, mlinks = mlinks,citeration = 2e4)
```

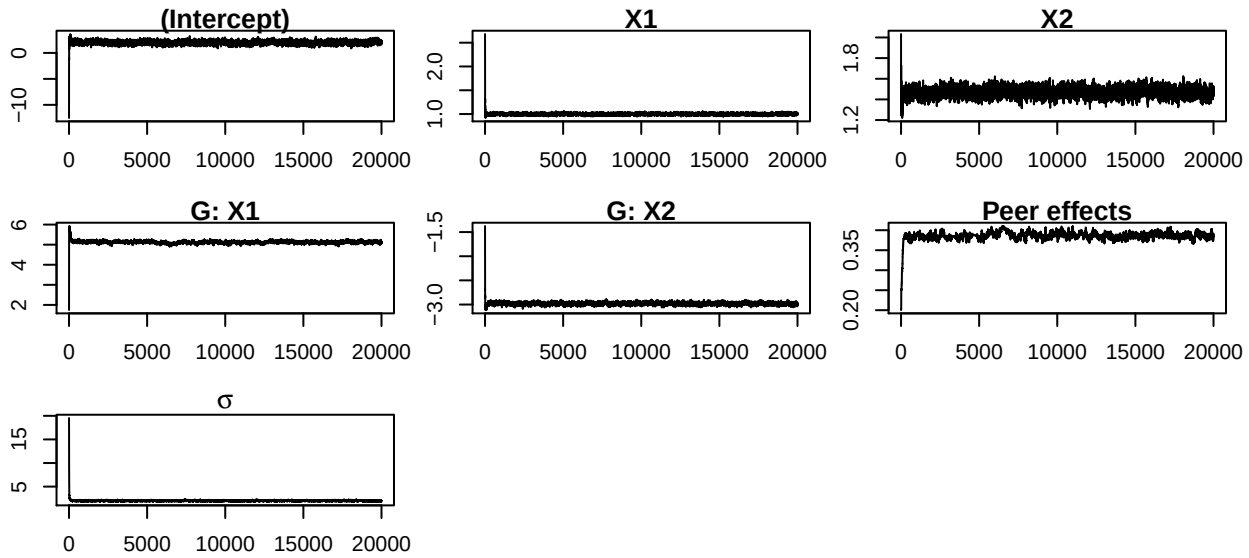
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  1  HH  18  mm  54  ss
##
## Peer effects acceptance rate: 0.4446
## rho acceptance rate         : 0.27905
```

```
summary(out.logi3.2)
```

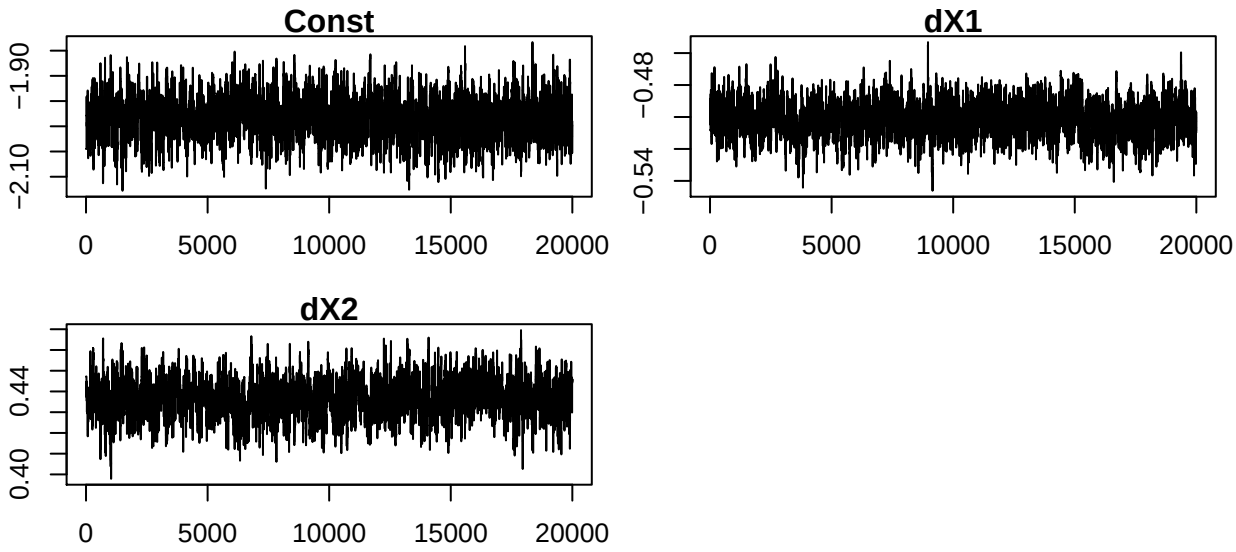
```
## Bayesian estimation of SAR model
##
## Formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of Observed Network Data: 0%
## Network formation model: logit
```

```
##
## Network Sampling
## Method: Gibbs sampler
## Update per block: No
##
## Network formation model
##             Mean   Std.Error     Inf CI     Sup CI Sign
## Const -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
```

```
plot(out.logi3.2, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```




```
plot(out.logi3.2, plot.type = "sim", which.parms = "rho", mar = c(3, 2.1, 1, 1))
```



4 Bayesian estimator with latent space model as network formation model

4.1 ARD, Breza et al. (2020)

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

```
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
Prior       <- sim.dnetwork(gennu, gend, genzeta, genz)
# Adjacency matrix
G           <- sim.network(Prior)
# 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
```

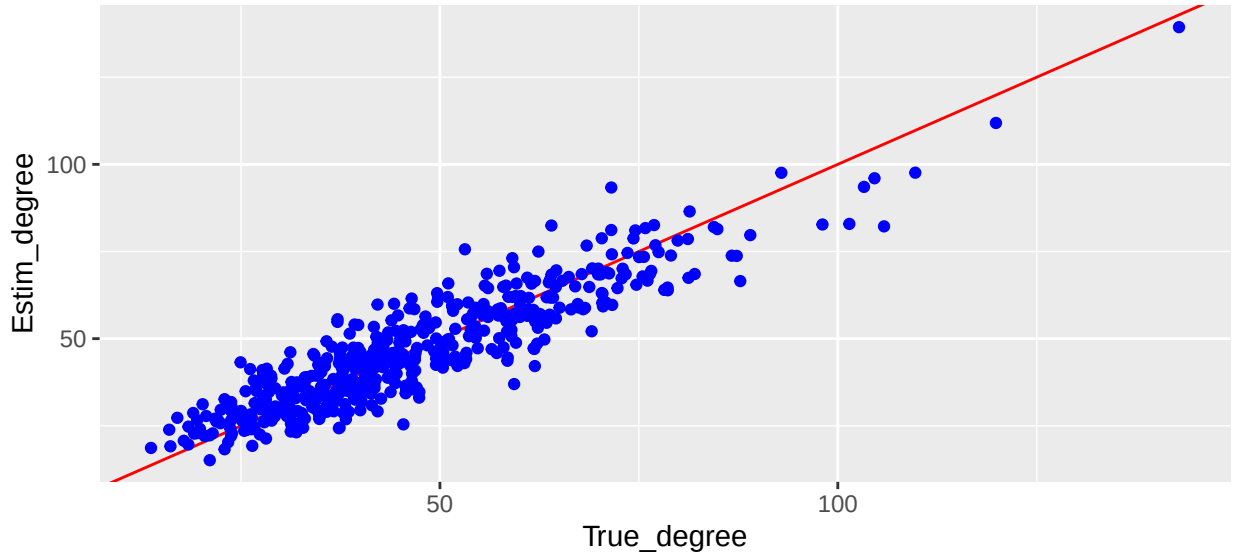
```
densityatz  <- matrix(0, N, K)
for(k in 1:K){
  densityatz[,k] <- dvMF(genz, genv[k,]*geneta[k])
}
trait            <- matrix(0, N, K)
for(k in 1:K){
  tmp       <- exp(geneta[k] + logCpvMF(3, geneta[k]))
  trait[,k] <- runif(N, 0.5*tmp, tmp) < densityatz[,k]
}
# Build ADR
ARD         <- G %*% trait
# generate b
genb        <- numeric(K)
for(k in 1:K){
  genb[k]   <- sum(G[,trait[,k]==1])/sum(G)
}
```

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

```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## ********SUMMARY********
## n               :  500
## K               :  12
## Dimension       :  3
## Iteration       :  5000
## Elapsed time    :  0  HH  1  mm  37  ss
##
## Average acceptance rate
##                       z:  0.441114
##                       d:  0.4399656
##                       b:  0.4408667
##                     eta:  0.4401833
##                    zeta:  0.4346
```

```
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")
```



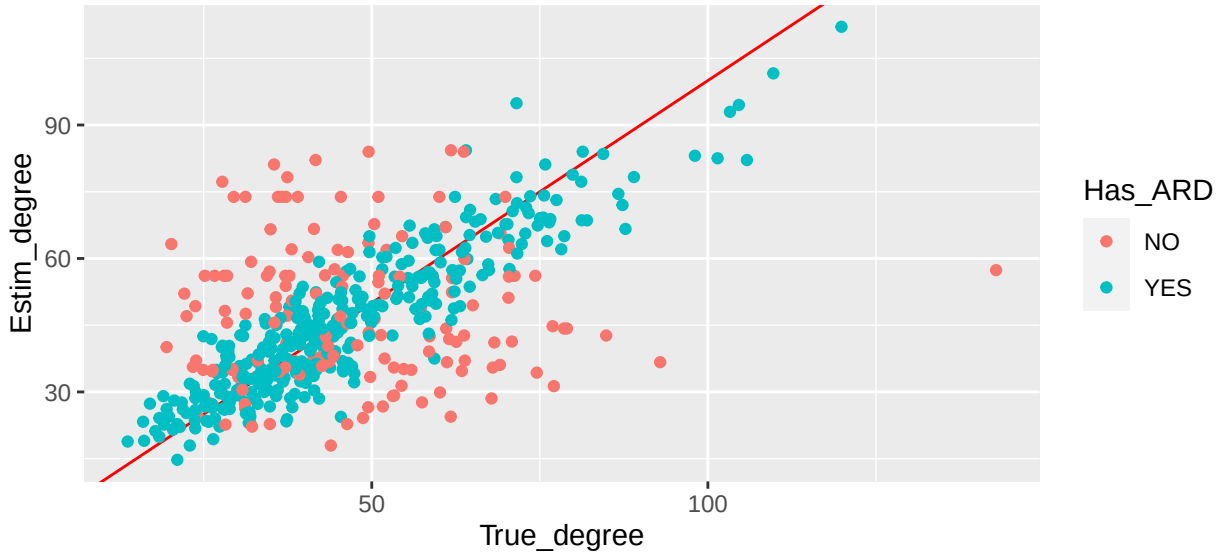

```
# Example2: ARD is observed for 70% population
# sample with ARD
n          <- round(0.7*N)
# individual with ARD
iselect    <- sort(sample(1:N, n, replace = FALSE))
ARDs       <- ARD[iselect,]
traits     <- trait[iselect,]
# initialization
d0         <- d0[iselect]; z0 <- z0[iselect,]
start      <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0, "zeta" = zeta0)
# MCMC
estim.ard2 <- mcmcARD(Y = ARDs, traitARD = traits, start = start, fixv = vfixcolumn,
                      consb = bfixcolumn, iteration = 5000)
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## ********SUMMARY********
## n              :  350
## K              :  12
## Dimension      :  3
## Iteration      :  5000
## Elapsed time   :  0  HH  1  mm  19  ss
##
## Average acceptance rate
##                      z:  0.4406074
##                      d:  0.4398063
##                      b:  0.44005
##                    eta:  0.4391333
##                   zeta:  0.4444
```

```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)
```

```
## ARD non observed on the entire population
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## Average link probabilities estimated
## Iteration             : 2500
## Elapsed time          : 0  HH  0  mm  20  ss
```

```r
# estimated degre
estd          <- rep(NA, N)
estd[hasARD]  <- colMeans(tail(estim.ard2$simulations$d, 2500))
estd[!hasARD] <- 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()
```



4.2 Estimating the SAR model with ARD

```r
library(PartialNetwork)
set.seed(123)
M             <- 50
N             <- rep(30, M)
genzeta       <- 3.5
mu            <- -1.35
sigma         <- 0.37
K             <- 12    # number of traits
P             <- 3     # Sphere dimension
```

```r
# In this loop, we generate data for the latent space model and
# estimate the latent space model in the M sub-networks
estimates     <- list()
list.trait    <- list()
prior         <- list()
G0            <- list()
for (m in 1:M) {
  # 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
  Probabilities <- sim.dnetwork(gennu, gend, genzeta, genz)
  prior[[m]]    <- Probabilities
  # Adjacency matrix
  G        <- sim.network(Probabilities)
  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)
  for(k in 1:K){
    tmp       <- exp(geneta[k] + logCpvMF(3, geneta[k]))
    trait[,k] <- runif(N[m], 0.5*tmp, tmp) < 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
  }

  # 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      <- 1:8
```

```
  b0[bfixcolumn]  <- genb[bfixcolumn]
  v0[vfixcolumn,] <- genv[vfixcolumn,]
  start           <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0,
                          "zeta" = zeta0)
  # MCMC
  estimates[[m]]  <- mcmcARD(Y = ARD, traitARD = trait, start = start, fixv = vfixcolumn,
                             sim.d = FALSE, sim.zeta = FALSE, consb = bfixcolumn,
                             iteration = 1000)
}
## Simulate X and y
# 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::simSARnet(~ 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]))
```

```
mlinks       <- 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)
```

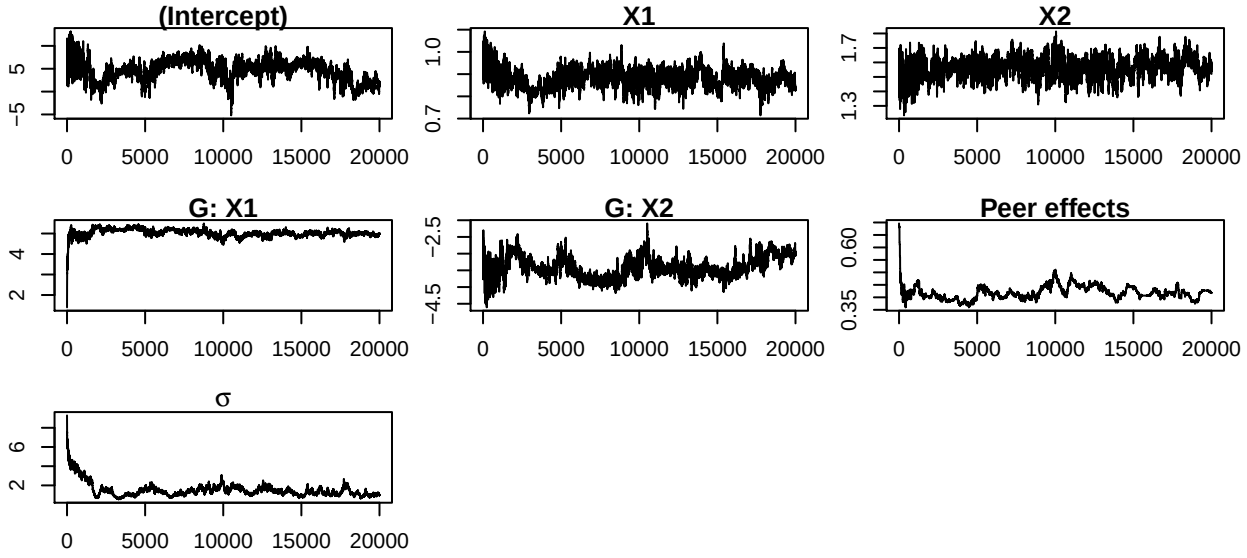
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY**************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  3  HH  46  mm  59  ss
##
## Peer effects acceptance rate: 0.44765
## rho acceptance rate         : 0.269653
```

```
summary(out.lspa1)
```

```
## Bayesian estimation of SAR model
##
## 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
##
## Network Sampling
## Method: Gibbs sampler
## Update per block: No
##
## Outcome model
##                     Mean  Std.Error     Inf CI     Sup CI Sign
## (Intercept)   4.2312311 2.28009318 -0.5356115  7.7744242
## X1            0.8783798 0.03669234  0.8054005  0.9520716    +
## X2            1.5642391 0.06601355  1.4283316  1.6876608    +
## G: X1         4.9632904 0.10738933  4.7252445  5.1508732    +
## G: X2        -3.3404467 0.29341367 -3.8209487 -2.7600029    -
## Peer effects  0.4250888 0.02276889  0.3841628  0.4727297    +
## ---
## Significance level: 95%
## ' ' = non signif.  '+' = signif. positive  '-' = signif. negative
##
## Error standard-deviation: 1.336564
## Number of groups: 50
## Total sample size: 1500
##
## Peer effects acceptance rate: 0.44765
## rho acceptance rate         : 0.269653
```

```
plot(out.lspa1, plot.type = "sim", mar = c(3, 2.1, 1, 1))
```



References

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.