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

Vincent Boucher and Elysée Aristide Houndetoungan

2021-04-16

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.

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            <- 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)
```

```r
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
## -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
```

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
## -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` (see the help page of the function in the package, using `?mcmcSAR`, for more details on the function). Below, we provide a simple example using simulated data.

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.

```
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]))
```

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

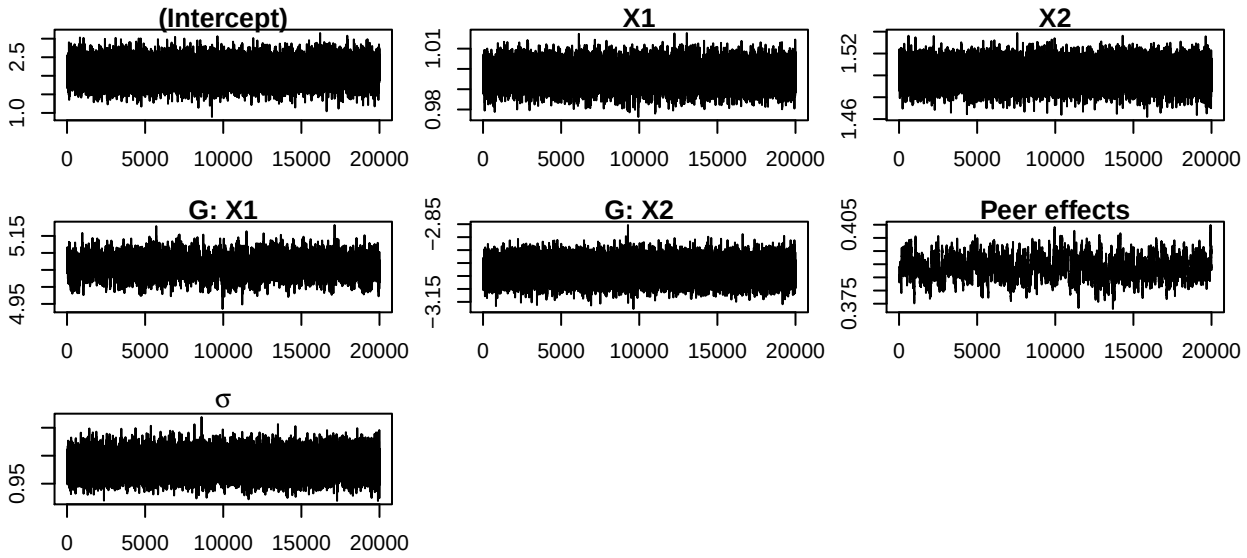
```r
# Example I-1: When the network is fully observed
out.none1     <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                         G0 = G0, data = dataset, iteration = 2e4)
```

```
## 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  0  mm  23  ss
##
## Peer effects acceptance rate: 0.44065
```

```r
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
```

```r
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)
```

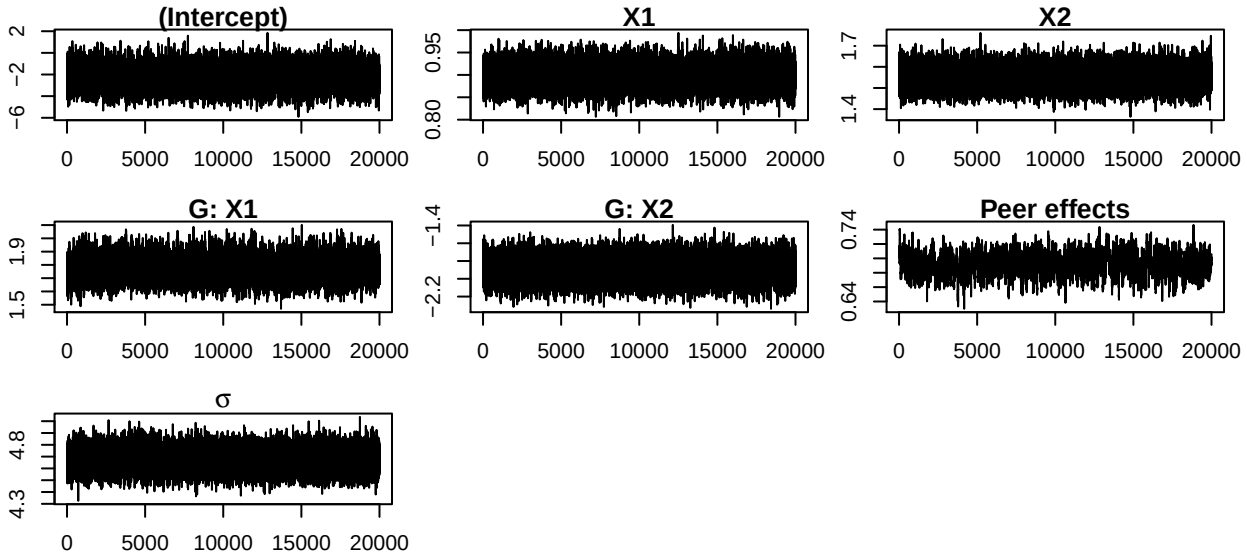
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  0  HH  0  mm  23  ss
##
## Peer effects acceptance rate: 0.4317
```

```
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 = prior), iteration = 2e4)
```

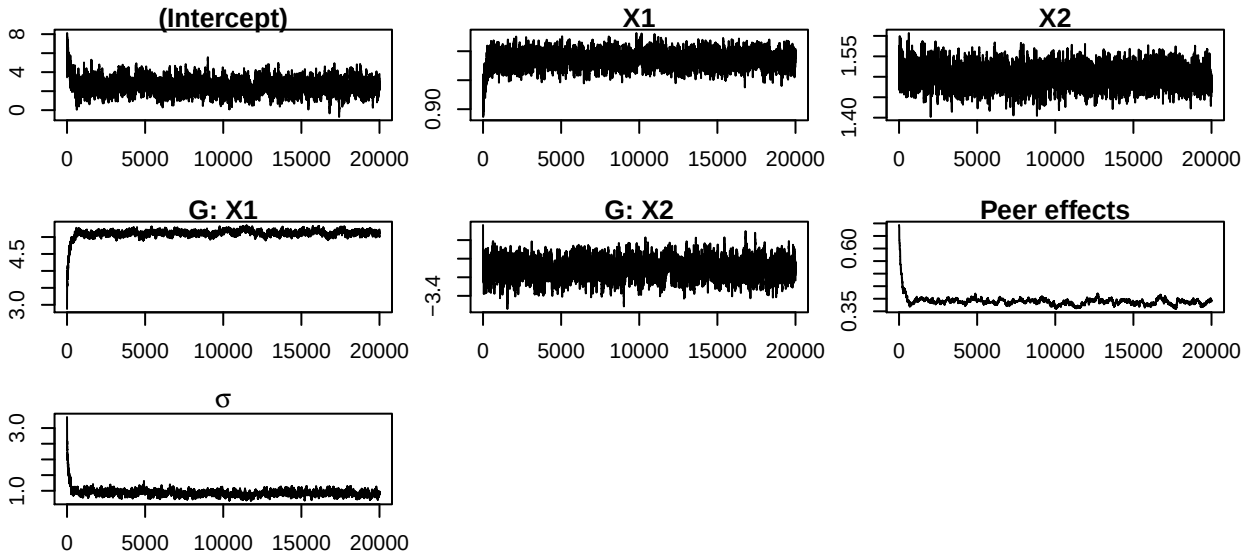
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
```

```
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  0  HH  41  mm  37  ss
##
## Peer effects acceptance rate: 0.4353
```

```
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(prior)
out.none3.1  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                        G0 = G0.tmp, data = dataset, iteration = 2e4)
```

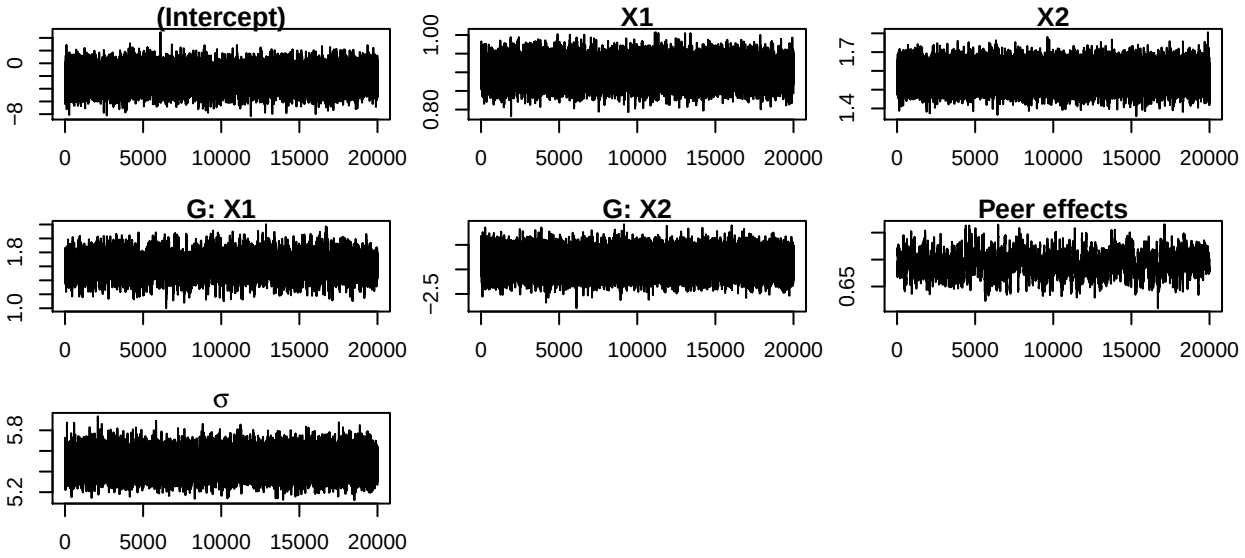
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  0  HH  0  mm  23  ss
##
## Peer effects acceptance rate: 0.44445
```

```
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
```

```
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).

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

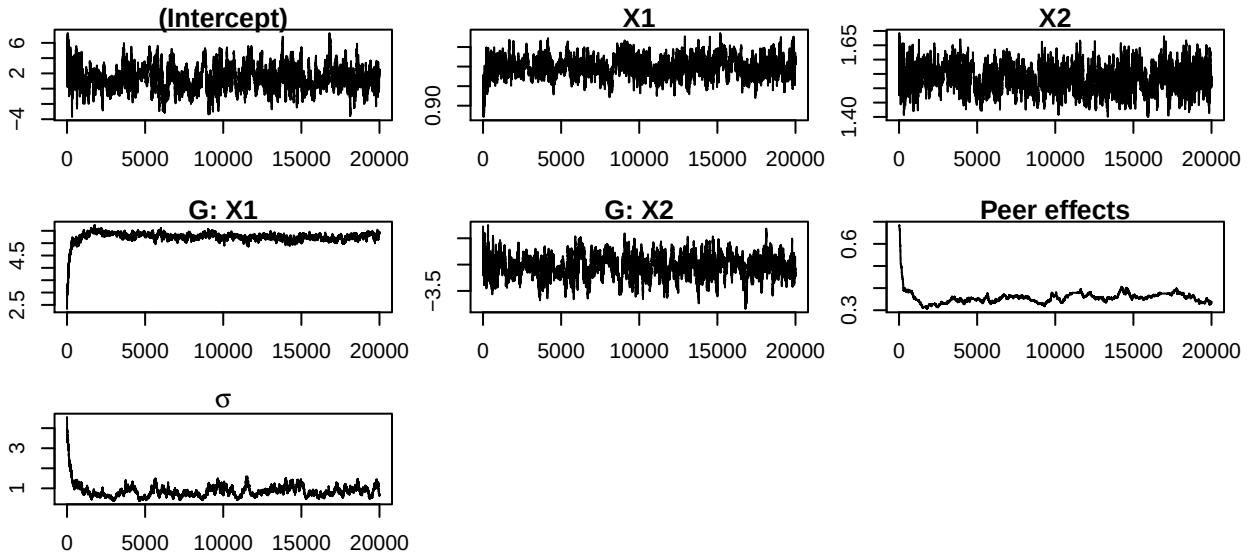
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  1  HH  42  mm  2  ss
```

```
##
## Peer effects acceptance rate: 0.42995
```

```
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 Bayesian estimator with logit model as network formation model

For this example, we assume that links are generated using a simple logit model.

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))
# 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]))
```

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

```
# 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(covar))
```

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

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

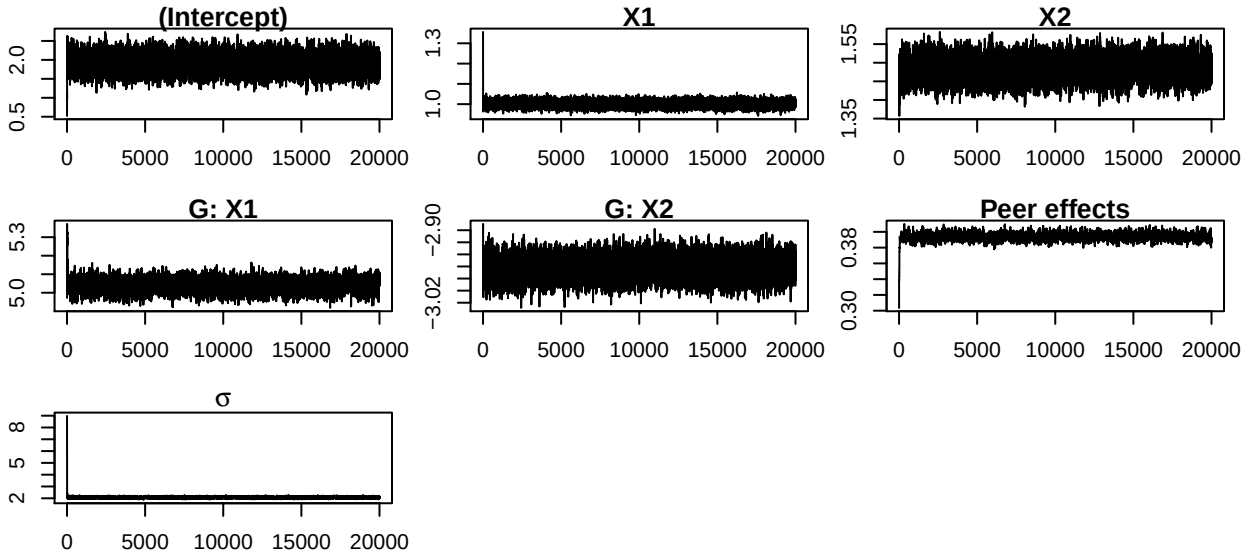
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  0  HH  33  mm  1  ss
##
## Peer effects acceptance rate: 0.438
## rho acceptance rate         : 0.27185
```

```
summary(out.logi2.2)
```

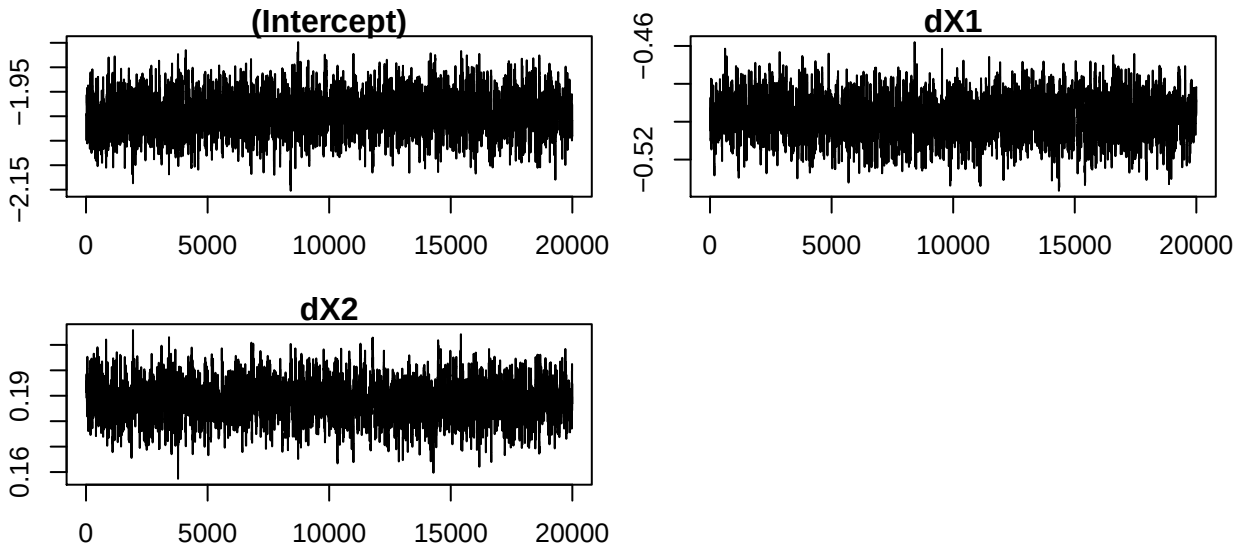
```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
## Number of steps performed: 20000
## Burn-in: 10000
##
## Percentage of observed network data: 59.86437%
## Network formation model: logit
```

```
## 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
```

```
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 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.

```
# 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", mlinks.formula = ~ dX1 + dX2,
                     mlinks.data = as.data.frame(covar), estimates = estimates)
```

```
out.logi3.2  <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "none",
                        data = dataset, mlinks = mlinks, iteration = 2e4)
```

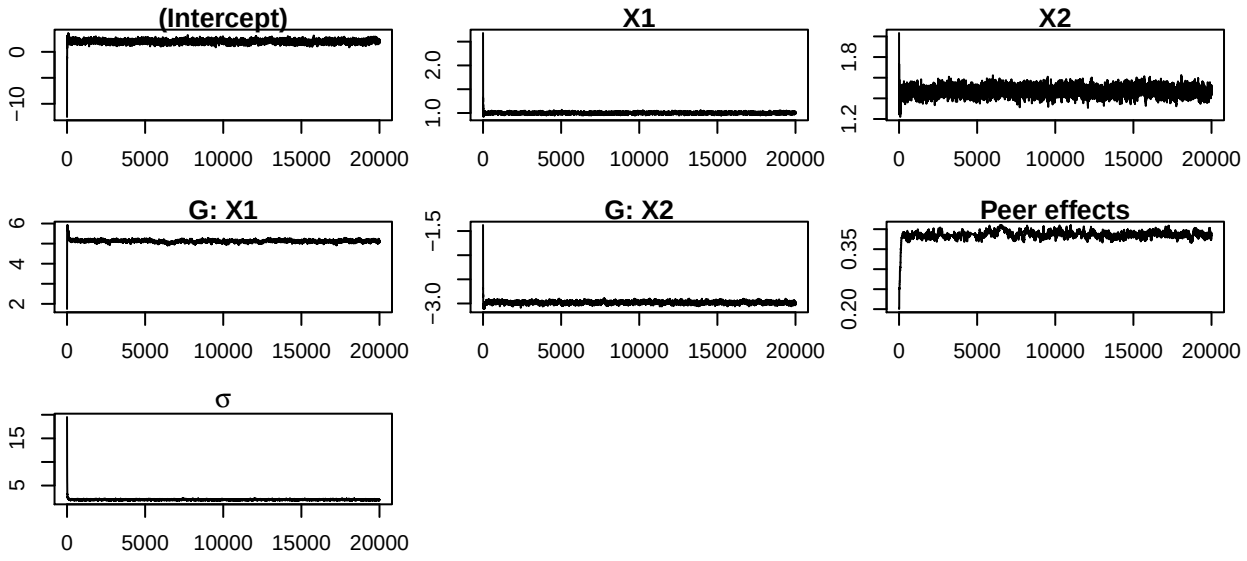
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY**************
## Number of group        :  50
## Iteration              :  20000
## Elapsed time           :  1  HH  20  mm  2  ss
##
## Peer effects acceptance rate: 0.4446
## rho acceptance rate         : 0.27905
```

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

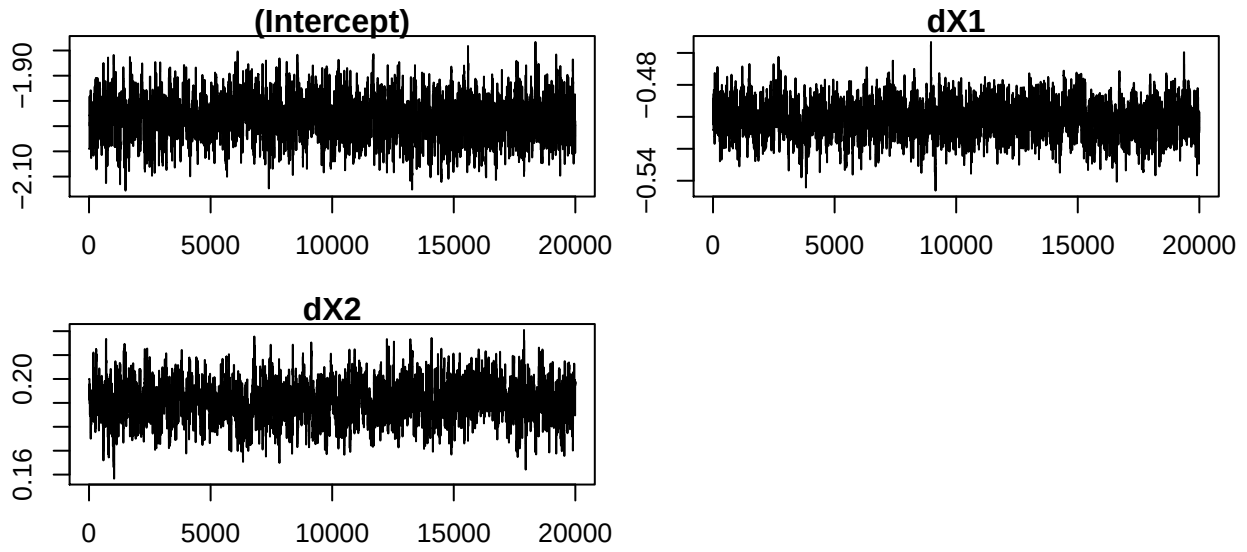
```
## Bayesian estimation of SAR model
##
## Outcome model's formula = y ~ X1 + X2 | X1 + X2
## Method: MCMC
```

```
## 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
```

```
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)

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).

```
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)
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

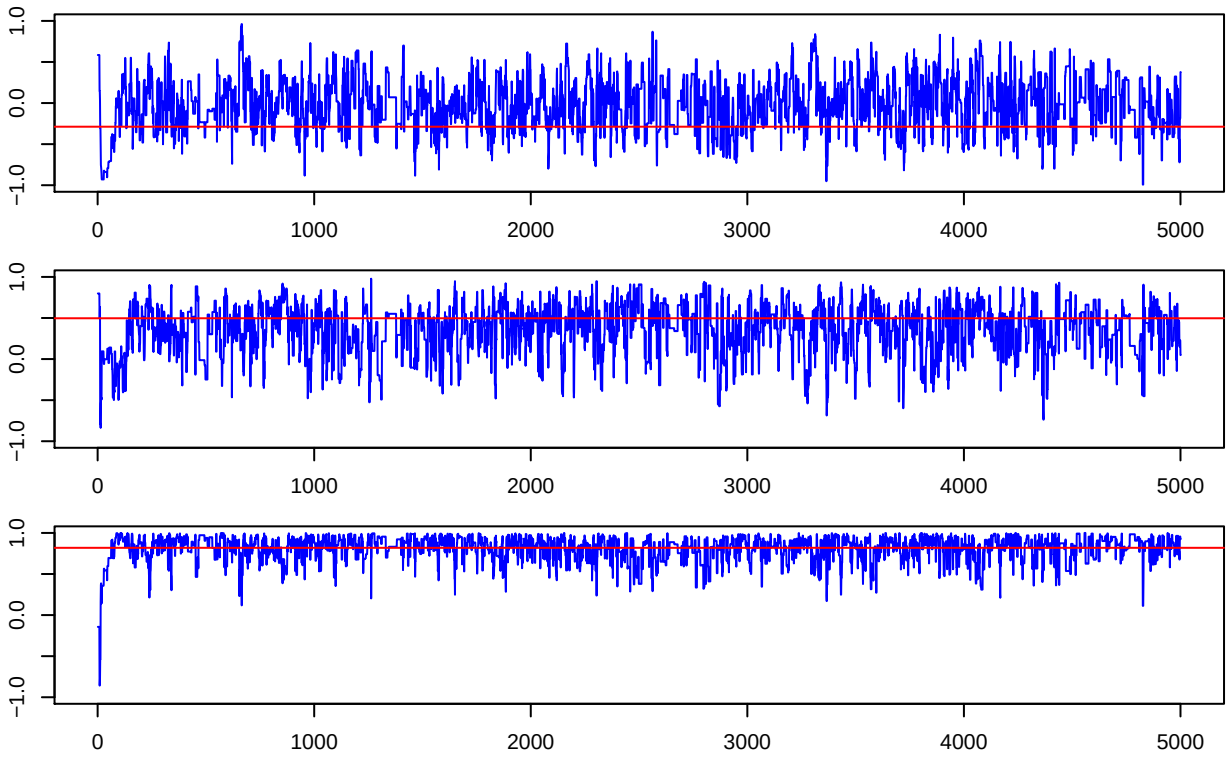
```
# Example1: ARD is observed for the whole population
# initialization
d0     <- exp(rnorm(N)); b0 <- exp(rnorm(K)); eta0 <- rep(1,K)
zeta0  <- 2; z0 <- matrix(rvMF(N, rep(0,P)), N); v0 <- matrix(rvMF(K,rep(0, P)), K)
# We should fix some vk and bk
vfixcolumn      <- 1:5
bfixcolumn      <- c(3, 7, 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`

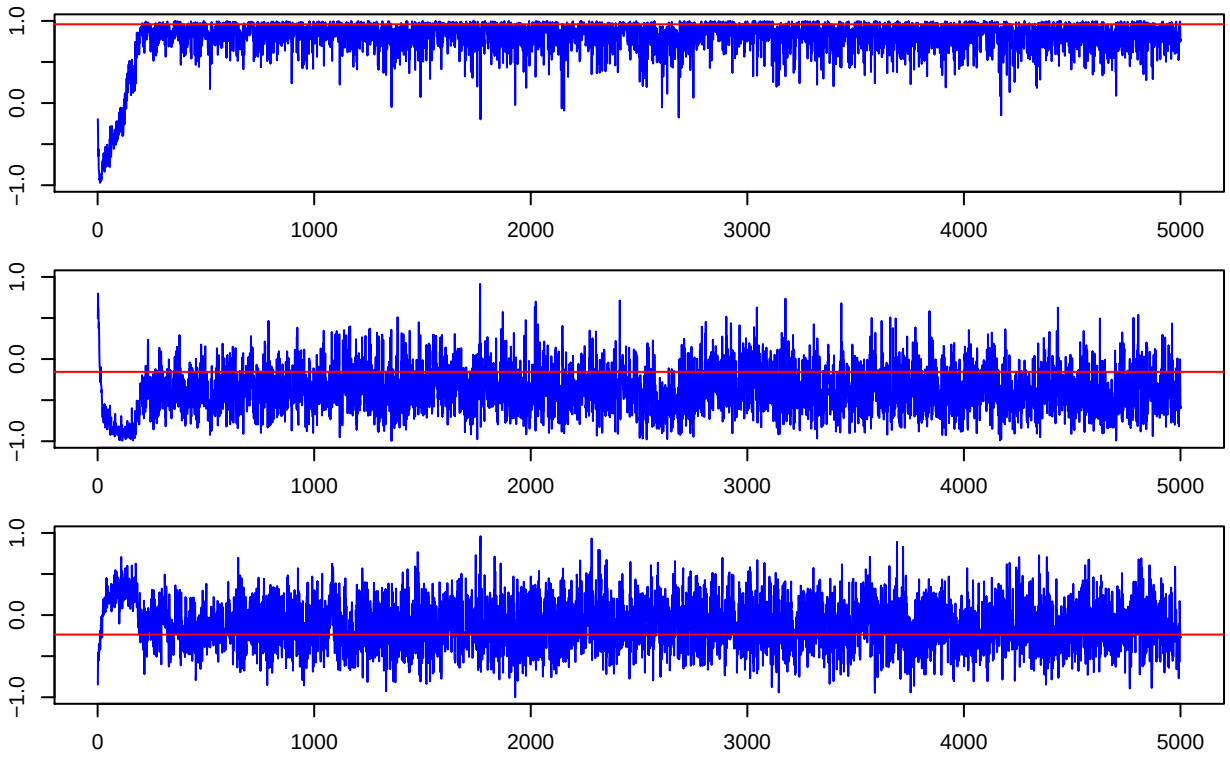
```
# MCMC
estim.ard1      <- mcmcARD(Y = ARD, traitARD = trait, start = start, fixv = vfixcolumn,
                           consb = bfixcolumn, iteration = 5000)
```

```
## 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  18  ss
##
## Average acceptance rate
##                        z:  0.441398
##                        d:  0.4399204
##                        b:  0.4392833
##                      eta:  0.4397833
##                     zeta:  0.4402
```

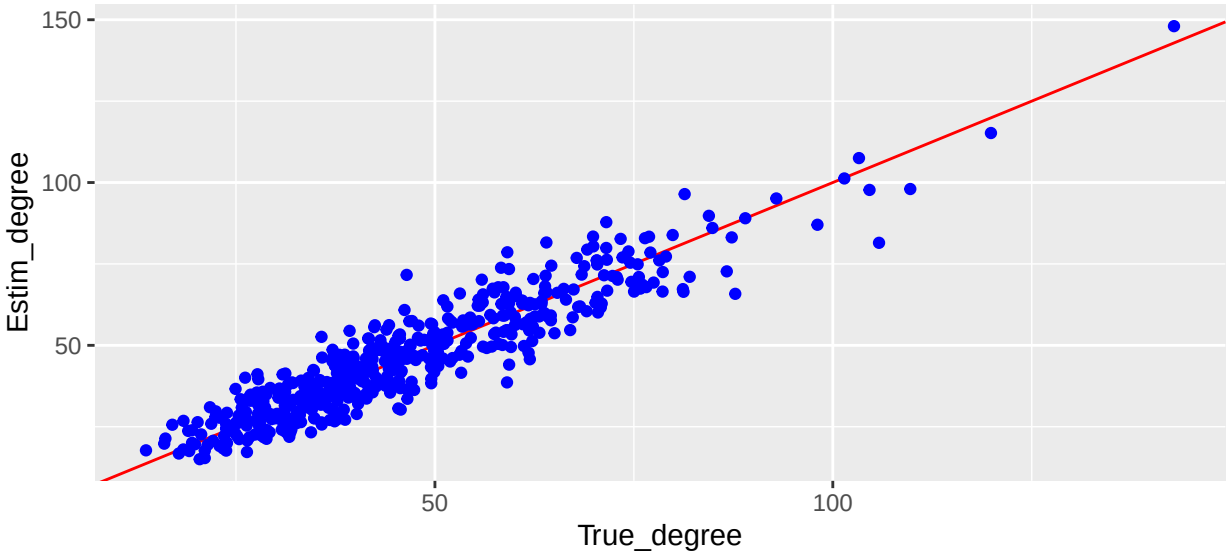
```
# plot coordinates of individual 123
i     <- 123
zi    <- estim.ard1$simulations$z[i,,]
par(mfrow = c(3, 1), mar = c(2.1, 2.1, 1, 1))
invisible(lapply(1:3, function(x) {
  plot(zi[x,], type = "l", ylab = "", col = "blue", ylim = c(-1, 1))
  abline(h = genz[i, x], col = "red")
}))
```




```
# plot coordinates of the trait 8
k     <- 8
vk    <- estim.ard1$simulations$v[k,,]
par(mfrow = c(3, 1), mar = c(2.1, 2.1, 1, 1))
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")
}))
```




```
# 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

```
# 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)
```

The estimation can be performed using the function `mcmcARD`

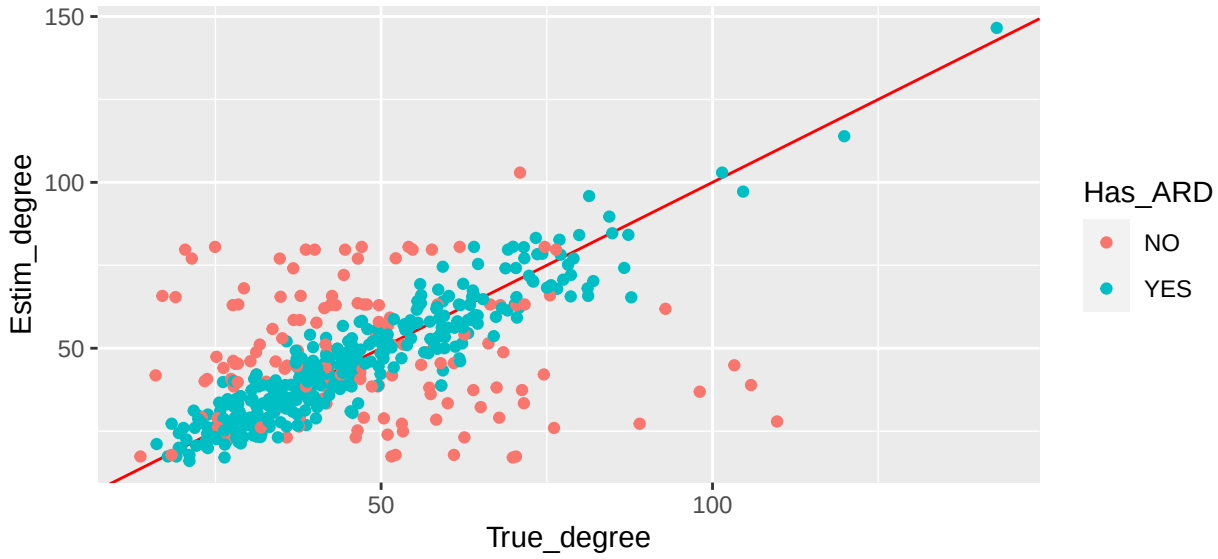
```
# MCMC
estim.ard2 <- mcmcARD(Y = ARDs, traitARD = traits, start = start, fixv = vfixcolumn,
                      consb = bfixcolumn, iteration = 5000)
```

```
## 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  0  mm  52  ss
##
## Average acceptance rate
##                       z:  0.4419703
##                       d:  0.4401291
##                       b:  0.4406667
##                     eta:  0.44055
##                    zeta:  0.4422
```

```
# 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  13  ss
```

```
# 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()
```



4.2 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.

```
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()
prior         <- 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
  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)
  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
  }

  ##############################################################################################
  #######                                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::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]))
```

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

```
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)
```

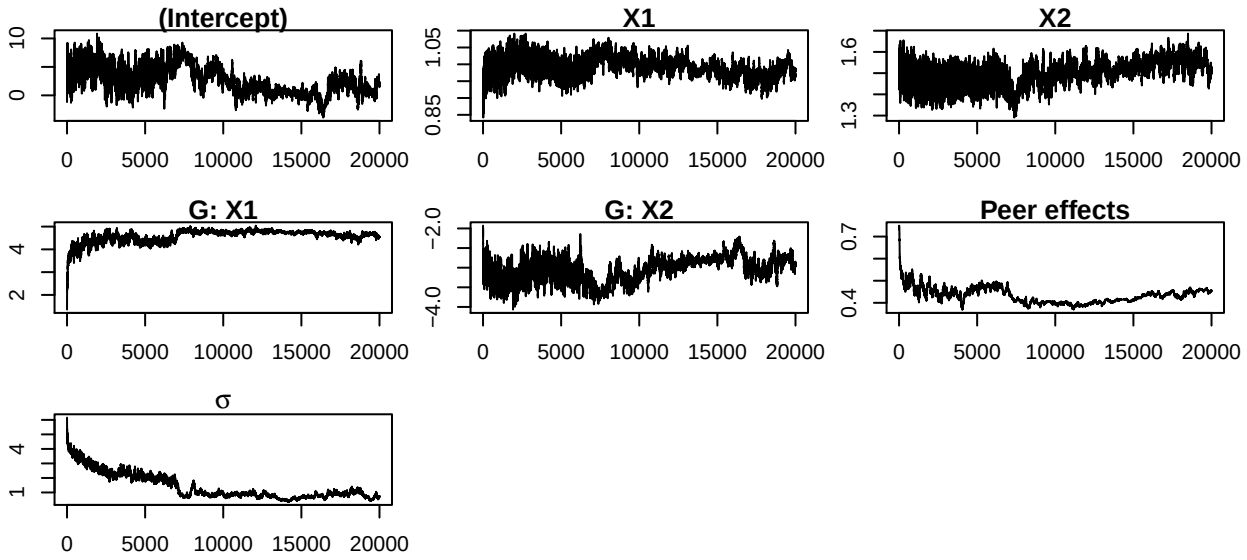
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
##
## *************SUMMARY*************
## Number of group        :  30
## Iteration              :  20000
## Elapsed time           :  7  HH  49  mm  17  ss
##
## Peer effects acceptance rate: 0.43145
## rho acceptance rate         : 0.2697867
```

```
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
```

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



4.3 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% $\sim$ 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()
prior         <- list()
GO            <- 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
  Probabilities <- sim.dnetwork(gennu, gend, genzeta, genz)
  prior[[m]]    <- Probabilities
  # Adjacency matrix
  G        <- sim.network(Probabilities)
  GO[[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
  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 GO
GOnorm        <- norm.network(GO)
# simulate dependent variable use an external package
y             <- CDatanet::simSARnet(~ X, contextual = TRUE, Glist = GOnorm,
                                     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`.

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

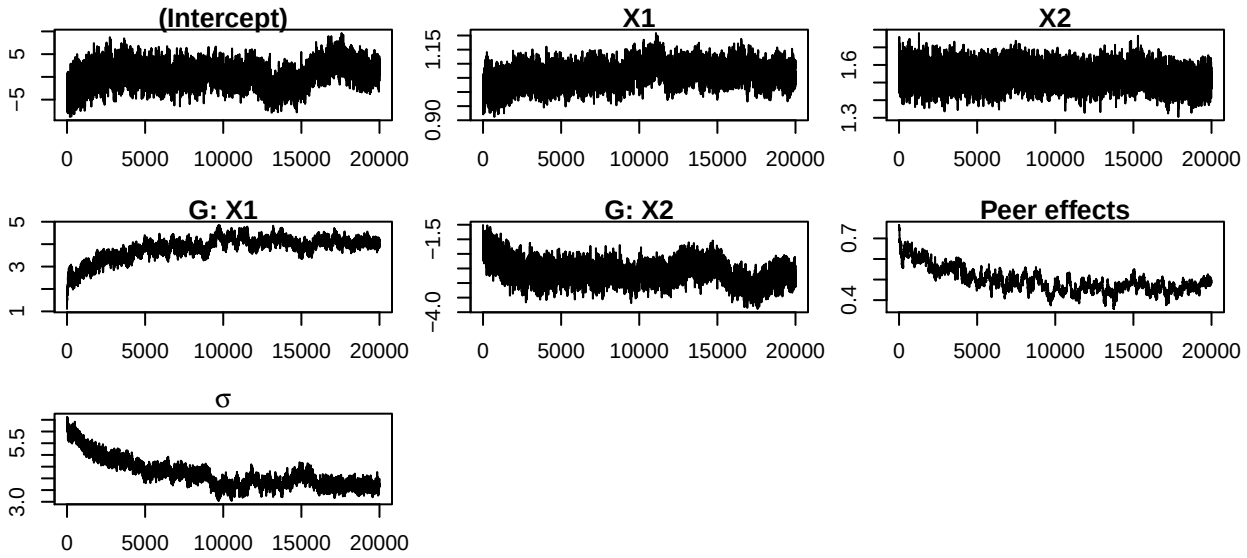
```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
## The program successfully executed
```

```
##
## *************SUMMARY*************
## Number of group        :  30
## Iteration              :  20000
## Elapsed time           :  7  HH  51  mm  0  ss
##
## Peer effects acceptance rate: 0.44075
## rho acceptance rate         : 0.2705733
```

```
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
```

```
plot(out.lspa2, 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.