CDatanet: An R package to simulate and estimate a Count Data Model with Social Interactions

Elysée Aristide Houndetoungan

2021-02-16

1 Introduction

There is a large and growing literature on peer effects in economics.[1] Recent contributions include, among others, models for discrete data, including binary (e.g., Lee et al., 2014; Liu, 2019), ordered (e.g., Boucher et al., 2018), and multinomial (e.g., Guerra and Mohnen, 2017). To my knowledge, however, there are no existing models for count variables with microeconomic foundations, despite these variables being prevalent in survey data (e.g., number of times teenagers have taken drugs, number of times teenagers have drunk alcohol, frequency of participation in sport activities, frequency of participation in extracurricular activities, frequency of restaurant visits, number of physician visits).

I present an incomplete information game with social interactions in which the outcome takes unbounded integer values. I show that it is important to account for the counting nature of the dependent variable. Assuming that this variable is continuous significantly underestimates the peer effects.

I also present an easy-to-use R package—named **CDatanet**—for implementing the model. This note is a supplement for the package manual. I provide examples with simulated data to show how to use the package. The package is located on GitHub at github.com/ahoundetoungan/CDatanet. All the results of the note can be replicated following the code provided.

The remainder of the paper is organized as follows. Section 2 briefly introduces the model. Sections 3 and 4 provide examples in which the network matrix is exogenous and endogenous respectively. They present a Monte Carlo study to assess the performance of the estimator of the model parameters. They also discuss how to control for the endogeneity of the network. Section 5 concludes.

2 Model

Let $\mathcal{V} = \{1, \dots, n\}$ be a set of n players indexed by i and y_i the observed integer outcome of player i (e.g., the number of cigarettes smoked per day or per week). I denote by $\boldsymbol{G}$ the adjacency matrix such that $G_{ij} = \dfrac{1}{n_i}$ if i knows j and $G_{ij} = 0$ otherwise, where n_i the number of friends of i. Let also $\boldsymbol{X}$ be the matrix of explanatory variables.

As in Liu (2019), I assume that there is a latent variable y_i^* which determines the observed count variable y_i . This latent variable y_i^* and the observed y_i are linked as follow.

Let $(a_q)_{q\in\mathbb{N}}$ be a sequence given by $a_0 = -\infty$, $a_1 \in \mathbb{R}$, and $a_q = a_1 + \gamma(q-1)$ for $q \in \mathbb{N}^*$ and $\gamma \in \mathbb{R}_+^*$. If $y_i^* \in (a_q, a_{q+1}]$ then $y_i = q$.

The first order conditions of the game equilibrium implies that

[1] For recent reviews, see Boucher and Fortin (2016), De Paula (2017) and Bramoullé et al. (2019).

$$y_i^* = \lambda \sum_{i=1}^{n} \sum_{r=0}^{\infty} r G_{ij} p_{ir} + \boldsymbol{x}_i' \boldsymbol{\beta} + \varepsilon_i$$

where p_{ir} is the probability of $y_i = r$, λ is the peer effect and $\varepsilon_i \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$.

The game equilibrium is given by

$$p_{ir} = \Phi\left(\frac{\lambda \sum_{i=1}^{n} \sum_{r=0}^{\infty} r G_{ij} p_{ir} + \boldsymbol{x}_i' \boldsymbol{\beta} - a_q}{\sigma_\varepsilon}\right) - \Phi\left(\frac{\lambda \sum_{i=1}^{n} \sum_{r=0}^{\infty} r G_{ij} p_{ir} + \boldsymbol{x}_i' \boldsymbol{\beta} - a_{q+1}}{\sigma_\varepsilon}\right)$$

where Φ is the probability density function of $\mathcal{N}(0, 1)$.

To estimate $\boldsymbol{\theta} = (\lambda, \boldsymbol{\beta}', \sigma_\varepsilon)'$, I rely on the Nested Partial Likelihood (NPL) method proposed by Aguirregabiria and Mira (2007). I refer the reader to Houndetoungan (2020) for more details. The estimation method is implemented in the package **CDatanet**.

3 Examples with exogenous network

In this section, I present examples to show how to use the package. I simulate data following the model by assuming that the network matrix is exogenous. I then show how to estimate the model parameters using functions provided by **CDatanet**. I also use Monte Carlo simulations to assess the performance of the estimator of the model parameters.

3.1 Data simulation

Given the adjacency matrix, the explanatory variables and $\boldsymbol{\theta}$, the function `simCDnet` can used to simulate data. I assume that there are `M` sub-networks. The number of individuals in each sub-network is randomly chosen between 100 and 1000. I assume three exogenous variables (including the intercept) and I control for the contextual effects.

```
set.seed(2020)
library(CDatanet)
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
print(nvec)
```

```
## [1] 682 455 657 529 222
```

```
n      <- sum(nvec)
print(n)
```

```
## [1] 2545
```

```
# Parameters
lambda <- 0.4              # peer effects
beta   <- c(2, -1.9, 0.8)  # own effects
gamma  <- c(1.5, -1.2)     # contextual effects
sigma  <- 1.5              # standard deviation of epsilon
theta  <- c(lambda, beta, gamma, sigma)

# X
data   <- data.frame(x1 = rnorm(n, 1, 1), x2 =  rexp(n, 0.4))
```

To simulate the network matrix, I assume that the number of friends of each individual is randomly chosen between 0 and 30.

```
# Network
Glist  <- list()
for (m in 1:M) {
  nm           <- nvec[m]
  Gm           <- matrix(0, nm, nm)
  max_d        <- 30
  for (i in 1:nm) {
    tmp        <- sample((1:nm)[-i], sample(0:max_d, 1))
    Gm[i, tmp] <- 1
  }
  rs           <- rowSums(Gm); rs[rs == 0] <- 1
  Gm           <- Gm/rs
  Glist[[m]]   <- Gm
}
```

I can now simulate the count data. The output of `simCDnet` includes `yst` the vector of y_i^*, `y` the vector of y_i, `yb` the vector of $\sum_{r=0}^{\infty} rp_{ir}$, `Gyb` the vector of $\sum_{i=1}^{n}\sum_{r=0}^{\infty} rG_{ij}p_{ir}$ and `iteration` as the number of iterations performed to find the fixed point `yb`.

```
ytmp    <- simCDnet(formula = ~ x1 + x2 | x1 + x2, Glist = Glist, theta = theta,
                    data = data)
names(ytmp)
```

```
## [1] "yst"       "y"         "yb"        "Gyb"       "iteration"
```

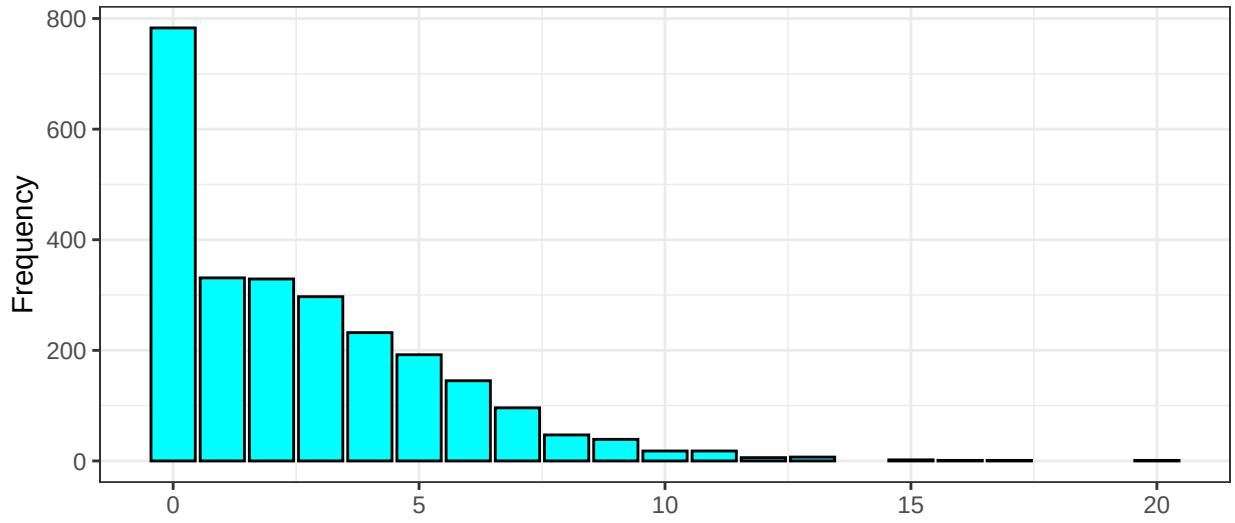
```
y       <- ytmp$y
# Add y to data
data$y  <- y
# Summarize y
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   2.000   2.617   4.000  20.000
```

```
table(y)
```

```
## y
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  15  16  17  20
## 783 331 329 297 232 192 145  96  47  39  18  18   6   7   2   1   1   1
```

```
# Plot data histogram
library(ggplot2)
print(ggplot(data = data.frame(y = y), aes(x = y)) +
        geom_bar(color = "black", fill = "cyan") +
        theme_bw() + xlab("") + ylab("Frequency"))
```



3.2 Estimation

Using the simulated data, I estimate the count data model as well as the spatial autoregressive Tobit (SART) and the spatial autoregressive (SAR) models. The SART model assumes that the dependent variable is continuous and left-censored at 0. This model can be used to model count data when data contain many zeros. However, the SART model does not account for the integer nature of the dependent variable. This can lead to biased estimates. In addition, the SART model also does not account for the integer nature of the dependent variable and does not assume that the data are censored. One can compare the peer effects using the count data model to those when the dependent variable is assumed continuous. The outputs of the functions that implement these models have a `summary class` to summarize the results and provide marginal effects.[2]

```
#  Count data
CD   <- CDnetNPL(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                  optimizer = "nlm",
                 data = data, npl.ctr = list(print = FALSE, maxit = 5e3))
summary(CD)
```

```
## Count data Model with Social Interactions
##
## Method: Nested pseudo-likelihood (NPL)
## Iteration:  12
##
## Network:
## Number of groups         :  5
## Sample size              :  2545
## Average number of friends:  14.98075
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.377313   0.075603    4.99 6.02e-07 ***
## (Intercept)  1.940142   0.155886   12.45   <2e-16 ***
## x1          -1.836252   0.037055  -49.55   <2e-16 ***
## x2           0.804280   0.012901   62.34   <2e-16 ***
## G: x1        1.587260   0.112376   14.12   <2e-16 ***
## G: x2       -1.211202   0.074873  -16.18   <2e-16 ***
```

[2]Note that the estimates from the SAR model can be directly interpreted as marginal effects.

```
##
## Marginal Effects:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.283665   0.056942    4.98 6.31e-07 ***
## (Intercept)  1.458607   0.117014   12.47   <2e-16 ***
## x1          -1.380502   0.033611  -41.07   <2e-16 ***
## x2           0.604661   0.011963   50.55   <2e-16 ***
## G: x1        1.193309   0.086218   13.84   <2e-16 ***
## G: x2       -0.910587   0.057630  -15.80   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## sigma:  1.503368
## log pseudo-likelihood:  -3650.167
```

```
# SART
SART   <- SARTML(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                optimizer = "nlm", data = data, print = FALSE)
summary(SART)
```

```
## SART Model
##
## Method: Maximum Likelihood (ML)
##
## Network:
## Number of groups         :  5
## Sample size              :  2545
## Average number of friends:  14.98075
## Proportion of zeros      :  0.3076621
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.266098   0.063163    4.21 2.52e-05 ***
## (Intercept)  2.526843   0.163713   15.43   <2e-16 ***
## x1          -1.929519   0.041139  -46.90   <2e-16 ***
## x2           0.829310   0.014717   56.35   <2e-16 ***
## G: x1        1.539243   0.119308   12.90   <2e-16 ***
## G: x2       -1.157105   0.068444  -16.91   <2e-16 ***
##
## Marginal Effects:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.192637   0.045700    4.22 2.49e-05 ***
## (Intercept)  1.829269   0.121043   15.11   <2e-16 ***
## x1          -1.396845   0.026602  -52.51   <2e-16 ***
## x2           0.600366   0.009641   62.27   <2e-16 ***
## G: x1        1.114311   0.085623   13.01   <2e-16 ***
## G: x2       -0.837668   0.048974  -17.10   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## sigma:  1.630561
## log likelihood:  -3811.928
```

```
#SAR
SAR  <- SARML(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
```

```
              optimizer = "nlm", data = data, print = FALSE)
summary(SAR)
```

```
## SAR Model
##
## Method: Maximum Likelihood (ML)
##
## Network:
## Number of groups         :  5
## Sample size              :  2545
## Average number of friends:  14.98075
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.163839   0.043641    3.75 0.000174 ***
## (Intercept)  2.532121   0.117490   21.55   <2e-16 ***
## x1          -1.386272   0.027828  -49.82   <2e-16 ***
## x2           0.706970   0.010947   64.58   <2e-16 ***
## G: x1        1.084204   0.085506   12.68   <2e-16 ***
## G: x2       -0.738526   0.046377  -15.92   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## sigma:  1.382775
## log likelihood:  -4436.1
```

Using the count data model, the peer effect is estimated at 0.38, which is close to the true value, 0.4. By replicating this experience several times, the average of the estimates is likely to be almost equal to the true value (see next section). Note that this parameter is not interpretable. Instead, one should interpret the marginal effect: *increasing the expected y_j of the peers by one implies an increase of 0.28 in the expected y_i*. However, the SART and the SAR model underestimate thos impact at 0.19 and 0.16, respectively.

3.3 Monte Carlo Simulations

In this section, I conduct Monte Carlo simulations to assess the performance of the estimator. I use the same settings as in Section 3.1 except that the number of individuals in each sub-network is set to 500. I build the following Monte Carlo function that simulates the data, estimates the three models and returns the estimates.

```
fMC <- function(s) {
  # Groups' size
  M      <- 5
  nvec   <- rep(500, 5);
  n      <- sum(nvec)
  # Parameters
  lambda <- 0.4
  beta   <- c(2, -1.9, 0.8)
  gamma  <- c(1.5, -1.2)
  sigma  <- 1.5
  theta  <- c(lambda, beta, gamma, sigma)

  # X
  data   <- data.frame(x1 = rnorm(n, 1, 1), x2 =  rexp(n, 0.4))

  # Network
  Glist  <- list()
```

```
  for (m in 1:M) {
    nm           <- nvec[m]
    Gm           <- matrix(0, nm, nm)
    max_d        <- 30
    for (i in 1:nm) {
      tmp        <- sample((1:nm)[-i], sample(0:max_d, 1))
      Gm[i, tmp] <- 1
    }
    rs           <- rowSums(Gm); rs[rs == 0] <- 1
    Gm           <- Gm/rs
    Glist[[m]]   <- Gm
  }

  # y
  ytmp   <- simCDnet(formula = ~ x1 + x2 | x1 + x2, Glist = Glist, theta = theta,
                     data = data)
  y      <- ytmp$y
  data$y <- y

  #  Models
  CD     <- CDnetNPL(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                     optimizer = "nlm",
                     data = data, npl.ctr = list(print = FALSE, maxit = 5e3))
  SART   <- SARTML(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                   optimizer = "nlm", data = data, print = FALSE)
  SAR    <- SARML(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                  optimizer = "nlm", data = data, print = FALSE)
  c(CD$estimate, SART$estimate, SAR$estimate)
}
```

I run 1000 times the Monte Carlo function `fMC` and compute the average of each estimate. To make it faster, I run the replications in parallel using **doParallel** package (Corporation and Weston, 2019).

```
library(doParallel)
n.cores  <- 32
replic   <- 1000 #Number of replications
out.mc   <- mclapply(1:replic, fMC, mc.cores = n.cores)
out.mc   <- apply(t(do.call(cbind, out.mc)), 2, mean)
out.mc   <- cbind(theta, out.mc[1:7], out.mc[8:14], out.mc[15:21])
colnames(out.mc) <- c("TrueValue", "CoutData", "SART", "SAR")
print(out.mc)
```

```
##             TrueValue   CoutData       SART        SAR
## lambda            0.4  0.4014868  0.2073720  0.1379745
## (Intercept)       2.0  1.9967916  2.7235037  2.6711174
## x1               -1.9 -1.8990528 -1.9996477 -1.4179726
## x2                0.8  0.7998454  0.8260788  0.7005770
## G: x1             1.5  1.4982137  1.3591351  0.9242272
## G: x2            -1.2 -1.1994937 -1.0790096 -0.6846019
## sigma             1.5  1.4974801  1.6256067  1.3795906
```

The estimator of the count data model seems unbiased while the SART and the SAR models underestimate the peer effects.

4 Examples with endogenous network

Peer effects estimation is generally based on the assumption of exogeneity of the adjacency matrix. This means that the probability of link formation is not correlated to the error term in the count data model. Such an assumption is strong as the link formation probability may depend on unobserved characteristics (e.g., gregariousness) that also influence the outcome. In this section, I simulate data with endogenous network (which violates the assumption). I then show that the peer effect is overestimated when one does not control for the endogeneity of the network. I also present a method to control for the endogeneity. Finally, I use Monte Carlo simulations to prove that this method performs well.

4.1 Data simulation

I assume three sub-networks. The number of individuals in each sub-network is randomly chosen between 100 and 500.

```
rm(list = ls())
set.seed(2020)
# Groups' size
M      <- 3 # Number of sub-groups
nvec   <- round(runif(M, 100, 500))
print(nvec)
```

```
## [1] 359 258 347
```

```
n      <- sum(nvec)
print(n)
```

```
## [1] 964
```

```
# Parameters
lambda <- 0.4               # peer effects
beta   <- c(2, -1.9, 0.8)   # own effects
gamma  <- c(1.5, -1.2)      # contextual effects
sigma  <- 1.5               # standard deviation of epsilon
theta  <- c(lambda, beta, gamma, sigma)

# X
data   <- data.frame(x1 = rnorm(n, 1, 1), x2 =  rexp(n, 0.4))
```

The network matrix follows the dyadic linking model presented in Houndetoungan (2020). Let $\boldsymbol{A}$ be the matrix of links such that $A_{ij} = 1$ if i knows j and $A_{ij} = 0$. In other words, $A_{ij} = 1$ if $G_{ij} = \frac{1}{n_i}$. For he network formation model, it is easier to work with $\boldsymbol{A}$ (binary data). However, for the count data model, I use $\boldsymbol{G}$ as row-normalized equivalent of $\boldsymbol{A}$.

The probability of link formation, P_{ij} depends on $\Delta x_{1ij} = |x_{1i} - x_{1j}|$ and $\Delta x_{2ij} = |x_{2i} - x_{2j}|$ (observed dyad-specific variables) as well as on unobserved individual-level attributes μ_i and μ_j. Typically, let a^*_{ij} defined by

$$a^*_{ij} = \Delta \boldsymbol{x}'_{ij}\bar{\boldsymbol{\beta}} + \mu_i + \mu_j + \varepsilon^*_{ij},$$

where $\varepsilon^*_{ij} \sim$ *logistic* distribution. I assume that $A_{ij} = 1$ if $a^*_{ij} > 0$. In that case, P_{ij} is given by

$$P_{ij} = \frac{\exp\left(\Delta \boldsymbol{x}'_{ij}\bar{\boldsymbol{\beta}} + \mu_i + \mu_j\right)}{1 + \exp\left(\Delta \boldsymbol{x}'_{ij}\bar{\boldsymbol{\beta}} + \mu_i + \mu_j\right)}.$$

I also assume that μ_i is random. If i is from the s-th sub-network, then $\mu_i \sim \mathcal{N}\left(u_{\mu s}, \sigma^2_{\mu s}\right)$. The mean and the variance of μ_i vary across the sub-networks. This is a way to control for the sub-network heterogeneity as fixed effects in the link formation probability.

```
# Parameter for network model
betanet  <- c(-2.8, -1.5)     # beta
Glist    <- list()            # adjacency matrix row normalized
Network  <- list()            # adjacency matrix row non-normalized
dX       <- matrix(0, 0, 2)   # observed dyad-specific variables
mu       <- list()            # unobserved individual-level attribute
uu       <- runif(M, -1, 1)   # mean of uu in each sub-network
sigma2u  <- runif(M, 0.5, 4) # variance of uu in each sub-network

# Network
for (m in 1:M) {
  nm           <- nvec[m]
  mum          <- rnorm(nm, uu[m], sqrt(sigma2u[m]))
  Z1           <- matrix(0, nm, nm)
  Z2           <- matrix(0, nm, nm)

  for (i in 1:nm) {
    for (j in 1:nm) {
      Z1[i, j] <- abs(data$x1[i] - data$x1[j])
      Z2[i, j] <- abs(data$x2[i] - data$x2[j])
    }
  }

  Gm           <- 1*((Z1*betanet[1] + Z2*betanet[2] +
                        kronecker(mum, t(mum), "+") + rlogis(nm^2)) > 0)
  diag(Gm)     <- 0

  diag(Z1)     <- NA
  diag(Z2)     <- NA
  Z1           <- Z1[!is.na(Z1)]
  Z2           <- Z2[!is.na(Z2)]

  dX           <- rbind(dX, cbind(Z1, Z2))

  Network[[m]] <- Gm
  rs           <- rowSums(Gm); rs[rs == 0] <- 1
  Gm           <- Gm/rs
  Glist[[m]]   <- Gm
  mu[[m]]      <- mum
}
mu            <- unlist(mu)
```

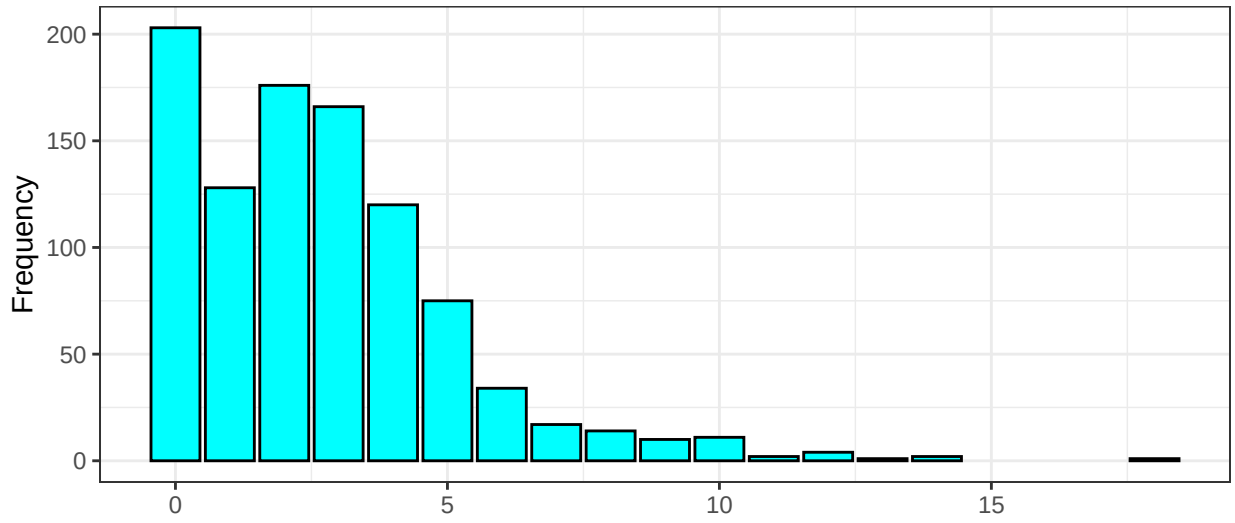
To simulate the dependent variable, I use $\boldsymbol{\mu}$, the vector of μ_i as additional explanatory variable in the count variable model. I set that $Cor(\mu_i, \varepsilon_i) = \rho$ and $Cor(\sum_{j=1}^{n} G_{ij}\mu_i, \varepsilon_i) = \bar{\rho}$. In that case, $\varepsilon_i = \rho\sigma_\varepsilon\tilde{\mu}_i + \bar{\rho}\sigma_\varepsilon\bar{\tilde{\mu}}_i + \tilde{\nu}_i$, where $\tilde{\mu}_i = \dfrac{\mu_i - u_{\mu s}}{\sigma_{\mu s}}$, $\bar{\tilde{\mu}}_i = \sum_{j=1}^{n} G_{ij}\tilde{\mu}_j$ is the average of $\tilde{\mu}_i$ among i's friends and $\tilde{\nu}_i$ is the new error term following normal distribution of variance $\bar{\sigma}_\varepsilon^2 = \left(1 - \rho^2 - \bar{\rho}^2\right)\sigma_\varepsilon^2$.

```
tmu      <- (mu - rep(uu, nvec))/sqrt(rep(sigma2u, nvec))
data$tmu <- tmu
rho      <- 0.24
rhobar   <- 0.18
thetanet <- c(lambda, beta, sigma*rho, gamma, sigma*rhobar,
              sigma*(1 - rho - rhobar))
```

```
ytmp     <- simCDnet(formula = ~ x1 + x2 + tmu | x1 + x2 + tmu,
                     Glist = Glist, theta = thetanet, data = data)
y        <- ytmp$y
# Add y to data
data$y   <- y
# Summarize y
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   2.000   2.694   4.000  18.000
```

```
# Plot data histogram
print(ggplot(data = data.frame(y = y), aes(x = y)) +
        geom_bar(color = "black", fill = "cyan") +
        theme_bw() + xlab("") + ylab("Frequency"))
```



In real-life, $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ are not observed and they are included in the error term ε_i. This implies that the network is endogenous.

4.2 Estimation

When the endogeneity of the network is not taken into account, the peer effects are overestimated. The reason is that the parameter λ also captures other effects. The model considers any common shock on $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ as peer effects. In the example below, the peer effects are estimated at 0.49 when I do not control for the endogeneity of the network.

```
# Count data model
CDexo <- CDnetNPL(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                  optimizer = "optim", data = data, npl.ctr = list(print = FALSE))
summary(CDexo)
```

```
## Count data Model with Social Interactions
##
## Method: Nested pseudo-likelihood (NPL)
## Iteration:  22
##
## Network:
## Number of groups         :  3
## Sample size              :  964
```

```
## Average number of friends:  27.30809
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.487503   0.037720   12.92   <2e-16 ***
## (Intercept)  1.825234   0.141787   12.87   <2e-16 ***
## x1          -1.947378   0.042677  -45.63   <2e-16 ***
## x2           0.803085   0.015453   51.97   <2e-16 ***
## G: x1        1.705766   0.076044   22.43   <2e-16 ***
## G: x2       -1.201023   0.026497  -45.33   <2e-16 ***
##
## Marginal Effects:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.410015   0.032023   12.80   <2e-16 ***
## (Intercept)  1.535114   0.118774   12.92   <2e-16 ***
## x1          -1.637843   0.042362  -38.66   <2e-16 ***
## x2           0.675435   0.014767   45.74   <2e-16 ***
## G: x1        1.434635   0.067551   21.24   <2e-16 ***
## G: x2       -1.010121   0.025133  -40.19   <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## sigma:  0.9333558
## log pseudo-likelihood:  -1139.806
```

To correct the endogeneity, $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ should be included in the model as additional explanatory variables. However, these variables are not observed.

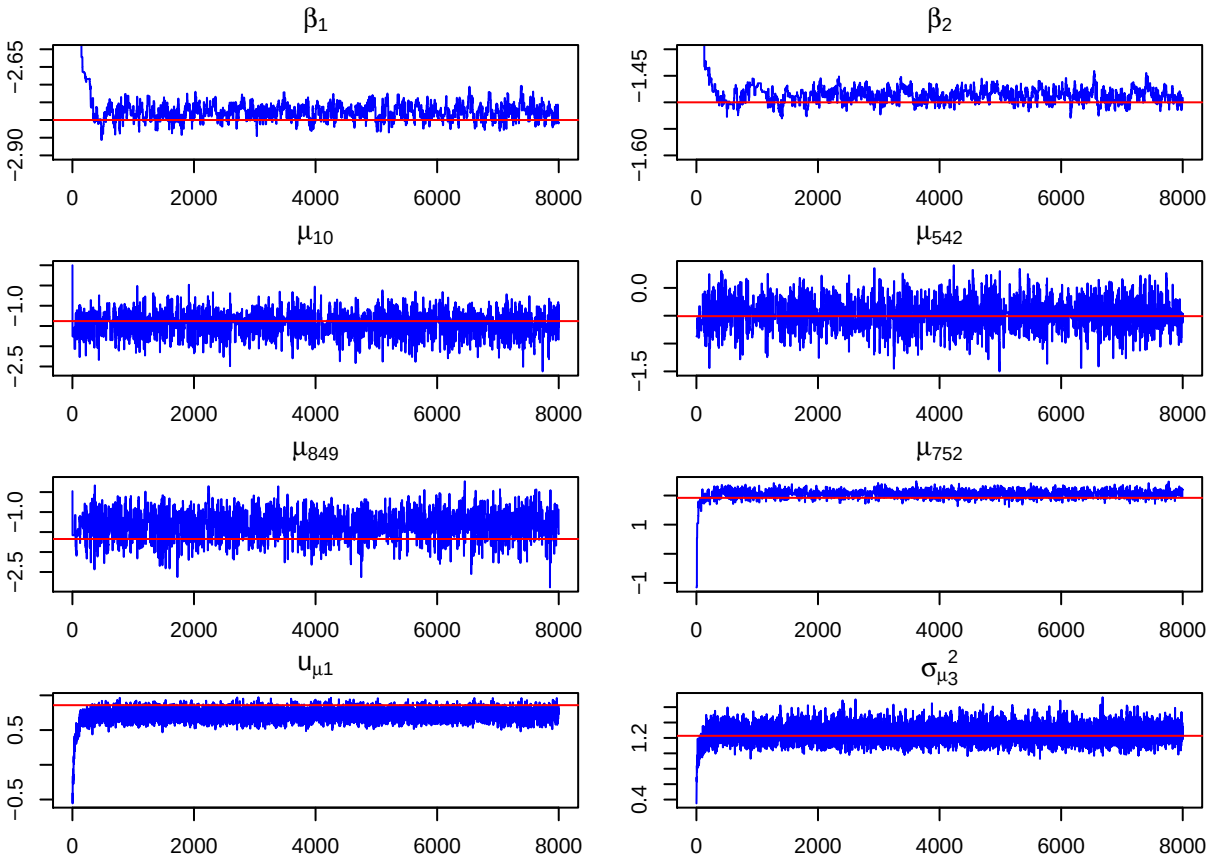
I use Markov Chain Monte Carlo (MCMC) to estimate the parameters of the dyadic linking model (these include the unobserved individual-level attributes μ_i). The estimation can be done using the function `netformation`.

```
# Dyadic linking model
net <- netformation(network =  Network, formula = ~ dX, fixed.effects = TRUE,
                    mcmc.ctr = list(burnin = 1000, iteration = 5000))
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
##
##
## The program successfully executed
##
## ********SUMMARY********
## n.obs           :  314890
## n.links         :  26325
## K               :  2
## Fixed effects   :  Yes
## Burnin          :  1000
## Iteration       :  6000
##
## Elapsed time    :  0  HH  7  mm  56  ss
##
## Average acceptance rate
##                     beta:  0.274125
##                       mu:  0.2697619
```

I randomly choose some parameters and present their posterior distribution.

```
# plot simulations
par(mfrow = c(4,2), mar = c(2, 2, 2, 1.9))
plot(net$posterior$beta[,1], type = "l", ylim = c(-2.9, -2.6), col = "blue",
     main = bquote(beta[1]), ylab = "")
abline(h = betanet[1], col = "red")

plot(net$posterior$beta[,2], type = "l", ylim = c(-1.6, -1.4), col = "blue",
     main = bquote(beta[2]), ylab = "")
abline(h = betanet[2], col = "red")

plot(net$posterior$mu[,10], type = "l", col = "blue",
     main = bquote(mu[10]), ylab = "")
abline(h = mu[10], col = "red")

plot(net$posterior$mu[,542], type = "l", col = "blue",
     main = bquote(mu[542]), ylab = "")
abline(h = mu[542], col = "red")

plot(net$posterior$mu[,849], type = "l", col = "blue",
     main = bquote(mu[849]), ylab = "")
abline(h = mu[849], col = "red")

plot(net$posterior$mu[,752], type = "l", col = "blue",
     main = bquote(mu[752]), ylab = "")
abline(h = mu[752], col = "red")

plot(net$posterior$uu[,1], type = "l", col = "blue",
     main = bquote(u[mu][1]), ylab = "")
abline(h = uu[1], col = "red")

plot(net$posterior$sigmamu2[,3], type = "l", col = "blue",
     main =  bquote(sigma[mu][3]^2), ylab = "")
abline(h = sigma2u[3], col = "red")
```



It stands out that the MCMC converges quickly and the simulations (plotted in blue) are pretty close to the true values (red line).

Using simulations from the posterior distribution, I can construct good estimators for $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ and use them as additional explanatory variables. In the next example, I use the simulation with the highest posterior density to estimate $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$.

```
t           <- which.max(net$posterior$log.density)
print(t)
```

```
## [1] 1683
```

```
muest       <- net$posterior$mu[t,]
uuest       <- net$posterior$uu[t,]
sigma2uest  <- net$posterior$sigmamu2[t,]
tmuest      <- (muest - rep(uuest, nvec))/sqrt(rep(sigma2uest, nvec))
data$tmuest <- tmuest
CDendo      <- CDnetNPL(formula = y ~ x1 + x2 + tmuest, contextual = TRUE,
                        Glist = Glist, optimizer = "optim", data = data,
                        npl.ctr = list(print = FALSE))
summary(CDendo)
```

```
## Count data Model with Social Interactions
##
## Method: Nested pseudo-likelihood (NPL)
## Iteration:  19
##
## Network:
## Number of groups         :  3
```

```
## Sample size              :  964
## Average number of friends:  27.30809
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.396270   0.039176   10.12   <2e-16 ***
## (Intercept)  1.872723   0.142741   13.12   <2e-16 ***
## x1          -1.934278   0.040658  -47.57   <2e-16 ***
## x2           0.806577   0.014741   54.72   <2e-16 ***
## tmuest       0.342133   0.035266    9.70   <2e-16 ***
## G: x1        1.602878   0.076345   21.00   <2e-16 ***
## G: x2       -1.208013   0.026067  -46.34   <2e-16 ***
## G: tmuest    0.280042   0.079627    3.52 0.000437 ***
##
## Marginal Effects:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.333358   0.033127   10.06   <2e-16 ***
## (Intercept)  1.575409   0.119909   13.14   <2e-16 ***
## x1          -1.627191   0.040496  -40.18   <2e-16 ***
## x2           0.678525   0.014293   47.47   <2e-16 ***
## tmuest       0.287816   0.030274    9.51   <2e-16 ***
## G: x1        1.348404   0.067268   20.05   <2e-16 ***
## G: x2       -1.016228   0.024838  -40.91   <2e-16 ***
## G: tmuest    0.235582   0.066863    3.52 0.000426 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## sigma:  0.880119
## log pseudo-likelihood:  -1094.926
```

The coefficients of $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ are significant. This confirms that the network is endogenous. Moreover, the peer effects are estimated at 0.40, which is much better than the previous estimate.

The standard errors computed in this example are not valid because they assume that $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ are observed. To correct the standard error, I replicate simulations of $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$ with replacement from the posterior distribution in order to take into account the variance of the MCMC.

```
fendo    <- function(s) {
  t         <- sample(3001:8000, 1)
  datat     <- data
  mus       <- net$posterior$mu[t,]
  uus       <- rep(net$posterior$uu[t,], nvec)
  sus       <- rep(net$posterior$sigmamu2[t,], nvec)
  tmu       <- (mus - uus)/sqrt(sus)
  datat$tmu <- tmu

  CDnet    <- CDnetNPL(formula = y ~ x1 + x2 + tmu, contextual = TRUE,
                       Glist = Glist, optimizer = "optim", data = datat,
                       npl.ctr = list(print = FALSE))

  summary(CDnet, Glist = Glist, data = datat)
}

n.cores    <- 32
replic     <- 1000 #Number of replications
```

```r
out.endo   <- mclapply(1:replic, fendo, mc.cores = n.cores)
# The output of out.endo is a list of objects from "summary.CDnetNPL" class
# Let's set the class of out.endo as "summary.CDnetNPLs"
class(out.endo) <- "summary.CDnetNPLs"
# I can now summarize the results using print
# Object from "summary.CDnetNPL" has a print method
print(out.endo)
```

```
## Count data Model with Social Interactions
##
## Method: Replication of Nested pseudo-likelihood (NPL)
## Replication:  1000
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.402133   0.040005   10.05   <2e-16 ***
## (Intercept)  1.864045   0.146209   12.75   <2e-16 ***
## x1          -1.935345   0.041119  -47.07   <2e-16 ***
## x2           0.806629   0.014940   53.99   <2e-16 ***
## tmu          0.337809   0.037415    9.03   <2e-16 ***
## G: x1        1.611186   0.078309   20.57   <2e-16 ***
## G: x2       -1.212639   0.026483  -45.79   <2e-16 ***
## G: tmu       0.255580   0.088880    2.88  0.00403  **
##
## Marginal Effects:
##              Estimate Std. Error t value Pr(>|t|)
## lambda       0.338501   0.033852   10.00   <2e-16 ***
## (Intercept)  1.569115   0.123193   12.74   <2e-16 ***
## x1          -1.629105   0.040575  -40.15   <2e-16 ***
## x2           0.678992   0.014350   47.32   <2e-16 ***
## tmu          0.284351   0.031994    8.89   <2e-16 ***
## G: x1        1.356236   0.068747   19.73   <2e-16 ***
## G: x2       -1.020757   0.025080  -40.70   <2e-16 ***
## G: tmu       0.215120   0.074660    2.88  0.00396  **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## sigma:  0.8792079
## log pseudo-likelihood:
##       min      mean       max
## -1104.369 -1094.693 -1085.772
```

The standard errors are slightly greater than the previous estimates.

4.3 Monte Carlo Simulations

One simulation is not sufficient to confirm that the method used to correct the endogeneity performs well. In this section, I use Monte Carlo simulations to assess the performance of this method. I compare the estimates when the network is assumed exogenous to these when I control for the endogeneity.

I build the following Monte Carlo function which simulates data based on endogenous network, and estimates the model by assuming that the network is exogenous and then endogenous.

```r
fMCendo <- function(s) {
  # parameters
```

```r
M         <- 3
nvec      <- rep(500, 3)
n         <- sum(nvec)

lambda    <- 0.4              # peer effects
beta      <- c(2, -1.9, 0.8)  # own effects
gamma     <- c(1.5, -1.2)     # contextual effects
sigma     <- 1.5              # standard deviation of epsilon
theta     <- c(lambda, beta, gamma, sigma)

data      <- data.frame(x1 = rnorm(n, 1, 1), x2 =  rexp(n, 0.4))

# Network
betanet   <- c(-2.8, -1.5)    # beta
Glist     <- list()           # adjacency matrix row normalized
Network   <- list()           # adjacency matrix row non-normalized
dX        <- matrix(0, 0, 2)  # observed dyad-specific variables
mu        <- list()           # unobserved individual-level attribute
uu        <- runif(M, -1, 1)  # mean of uu in each sub-network
sigma2u   <- runif(M, 0.5, 4) # variance of uu in each sub-network

for (m in 1:M) {
  nm            <- nvec[m]
  mum           <- rnorm(nm, uu[m], sqrt(sigma2u[m]))
  Z1            <- matrix(0, nm, nm)
  Z2            <- matrix(0, nm, nm)

  for (i in 1:nm) {
    for (j in 1:nm) {
      Z1[i, j] <- abs(data$x1[i] - data$x1[j])
      Z2[i, j] <- abs(data$x2[i] - data$x2[j])
    }
  }

  Gm            <- 1*((Z1*betanet[1] + Z2*betanet[2] +
                         kronecker(mum, t(mum), "+") + rlogis(nm^2)) > 0)
  diag(Gm)      <- 0

  diag(Z1)      <- NA
  diag(Z2)      <- NA
  Z1            <- Z1[!is.na(Z1)]
  Z2            <- Z2[!is.na(Z2)]

  dX            <- rbind(dX, cbind(Z1, Z2))

  Network[[m]] <- Gm
  rs            <- rowSums(Gm); rs[rs == 0] <- 1
  Gm            <- Gm/rs
  Glist[[m]]    <- Gm
  mu[[m]]       <- mum
}
mu              <- unlist(mu)
```

```
  # Data
  tmu      <- (mu - rep(uu, nvec))/sqrt(rep(sigma2u, nvec))
  data$tmu <- tmu
  rho      <- 0.24
  rhobar   <- 0.18
  thetanet <- c(lambda, beta, sigma*rho, gamma, sigma*rhobar,
                sigma*(1 - rho - rhobar))
  ytmp     <- simCDnet(formula = ~ x1 + x2 + tmu | x1 + x2 + tmu,
                       Glist = Glist, theta = thetanet, data = data)
  y        <- ytmp$y
  data$y   <- y

  # Count data model
  CDexo <- CDnetNPL(formula = y ~ x1 + x2, contextual = TRUE, Glist = Glist,
                    optimizer = "optim", data = data, npl.ctr = list(print = FALSE))
  # Dyadic linking model
  net   <- netformation(network =  Network, formula = ~ dX, fixed.effects = TRUE,
                        mcmc.ctr = list(burnin = 1000, iteration = 2000), print = FALSE)
  # Endogeneity
  t           <- which.max(net$posterior$log.density)
  muest       <- net$posterior$mu[t,]
  uuest       <- net$posterior$uu[t,]
  sigma2uest  <- net$posterior$sigmamu2[t,]
  tmuest      <- (muest - rep(uuest, nvec))/sqrt(rep(sigma2uest, nvec))
  data$tmuest <- tmuest
  CDendo      <- CDnetNPL(formula = y ~ x1 + x2 + tmuest, contextual = TRUE,
                          Glist = Glist, optimizer = "optim", data = data,
                          npl.ctr = list(print = FALSE))
  c(CDexo$estimate, CDendo$estimate)
}
```

I now run 1000 times the function `fMCendo` and compute the average of both estimators.

```
n.cores               <- 5
replic                <- 1000 #Number of replications
out.mcendo            <- mclapply(1:replic, fMCendo, mc.cores = n.cores)
out.mcendo            <- apply(t(do.call(cbind, out.mcendo)), 2, mean)
out.endo              <- cbind(thetanet, c(out.mcendo[c(1:4, NA, 5:6, NA, 7)]),
                               out.mcendo[8:16])
rownames(out.endo)    <- names(out.mcendo[8:16])
colnames(out.endo)    <- c("TrueValue", "Endo-Not-Controlled", "Endo-Controlled")
print(out.endo)
```

```
##             TrueValue Endo-Not-Controlled Endo-Controlled
## lambda           0.40           0.5201007       0.4186179
## (Intercept)      2.00           1.7957977       1.9745290
## x1              -1.90          -1.9077019      -1.8984809
## x2               0.80           0.8019128       0.8002151
## tmuest           0.36                  NA       0.3344702
## G: x1            1.50           1.6644998       1.5205194
## G: x2           -1.20          -1.1805244      -1.2008929
## G: tmuest        0.27                  NA       0.2286079
## sigma            0.87           0.9303622       0.8780438
```

The results prove that the method used to correct the endogeneity performs fairly well. The average of the

estimates is 0.42 while the true value is 0.40. There is still a small bias. This is certainly due to the number of iterations of the MCMC which is low. Indeed, I use the iteration with the highest posterior density to estimate $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$. Since the number of iterations is low, the estimates may not be very good: the more the number of iterations, the better the simulation with highest posterior density for approximating $\tilde{\mu}_i$ and $\bar{\tilde{\mu}}_i$.

In contrast, the peer effects are overestimated when the network is assumed exogenous. The average of the estimates is 0.52.

5 Conclusion

This paper provides technical details on the package **CDatanet**. It shows with simple and practical examples how to use the package. It provides comparison of the model with the SART and the SAR models. It also shows with Monte Carlo simulations how the model performs when the network matrix is exogenous and endogenous. Simulation results prove that the peer effects are overestimated when the network is endogenous. Lastly, it presents a way to control for the endogeneity of the network.

References

Aguirregabiria, V. and Mira, P. (2007). Sequential estimation of dynamic discrete games. *Econometrica*, 75(1):1–53.

Boucher, V., Dedewanou, F. A., and Dufays, A. (2018). Peer-induced beliefs regarding college participation. .

Boucher, V. and Fortin, B. (2016). Some challenges in the empirics of the effects of networks. *The Oxford Handbook on the Economics of Networks*, pages 277–302.

Bramoullé, Y., Djebbari, H., and Fortin, B. (2019). Peer effects in networks: A survey. *Annual Review of Economics, forthcoming*.

Corporation, M. and Weston, S. (2019). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.15.

De Paula, A. (2017). Econometrics of network models. In *Advances in economics and econometrics: Theory and applications, eleventh world congress*, pages 268–323. Cambridge University Press Cambridge.

Guerra, J.-A. and Mohnen, M. (2017). Multinomial choice with social interactions: occupations in victorian london. *Documento CEDE*, (2017-47).

Houndetoungan, E. A. (2020). A count data model with social interactions. *Available at SSRN 3721250*.

Lee, L.-f., Li, J., and Lin, X. (2014). Binary choice models with social network under heterogeneous rational expectations. *Review of Economics and Statistics*, 96(3):402–417.

Liu, X. (2019). Simultaneous equations with binary outcomes and social interactions. *Econometric Reviews*, 38(8):921–937.