Example of confounding homophily with diffusion

George G. Vega Yon

May 24, 2016

```
rm(list=ls())

library(Matrix) # To handle sparse matrices
library(spdep)  # To run SAR model
```

```
## Loading required package: sp
```

```
library(spatialprobit) # To run SAR probit
```

```
## Loading required package: mvtnorm

## Loading required package: tmvtnorm

## Loading required package: stats4

## Loading required package: gmm

## Loading required package: sandwich
```

```
# Parameters for the simulation
set.seed(121)
n <- 5e2
```

Data generating process

- The latent variable follows $L \sim N(0, 2)$
- The graph is homophily-based as $\Pr\left(i \to j\right) = 1 - \|L_i - L_j\|$
- The behavior is based only on L (no diffusion): $\Pr\left(y = 1\right) = \Phi(L)$

```
# Latent variable (could be 'interest on implementing policy')
L <- rnorm(n, sd = 2)

# Generating random graph
# i and j are connected with probability proportional to the euclidean
# distance, this is
# Pr( i -> j) ~ 1 - |L[i] - L[j]|
A <- matrix(0, ncol=n, nrow=n)
for (i in 1:n)
  for (j in 1:n)
    if ((i != j) && (1 - abs(L[i] - L[j])) > runif(1))
      A[i,j] <- 1

# Computing density and taking a look
sum(A)/(n*(n-1))
```

```
## [1] 0.1469018
```

```
A[1:10,1:10]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    0    1    1    0    0    0    0    0     0
##  [2,]    0    0    1    1    0    0    0    0    0     0
##  [3,]    1    1    0    0    0    0    1    1    0     0
##  [4,]    0    1    1    0    0    0    0    1    0     0
##  [5,]    0    0    0    0    0    0    0    0    0     0
##  [6,]    0    0    0    0    0    0    0    0    0     0
##  [7,]    0    1    1    1    0    0    0    0    0     0
##  [8,]    0    1    1    1    0    0    0    0    0     0
##  [9,]    0    0    0    0    0    0    0    0    0     0
## [10,]    0    0    0    0    0    0    0    0    0     0
```

```r
# library(igraph)
# ig <- graph_from_adjacency_matrix(A)
# plot(ig, vertex.size=1, layout=layout_with_fr)

# Now generating behavior using the same variable.
# Pr(y=1) = Pr(L > 0) = Phi(L)
y <- as.integer(pnorm(L, sd=2) > runif(n))
```

Logit model

```
# Now we run a logit, ---------------------------------------------------------

# the weighting matrix W is in fact the exposure
# level to the adoption y.
W <- A/(rowSums(A) + 1e-15)
summary(glm(y ~ I(W %*% y), family=binomial()))
```

```
##
## Call:
## glm(formula = y ~ I(W %*% y), family = binomial())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1066  -0.8222  -0.3967   0.8461   2.2723
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.5029     0.2541   -9.85   <2e-16 ***
## I(W %*% y)    5.0255     0.4743   10.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.18  on 499  degrees of freedom
## Residual deviance: 531.33  on 498  degrees of freedom
## AIC: 535.33
##
## Number of Fisher Scoring iterations: 4
```

```
# Controling for Latent variable (which we can't)
summary(glm(y ~ I(W %*% y) + L, family=binomial()))
```

```
##
## Call:
## glm(formula = y ~ I(W %*% y) + L, family = binomial())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7854  -0.8333  -0.1629   0.8306   2.2854
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.8268     1.5221   1.200  0.23006
## I(W %*% y)   -3.9182     3.1183  -1.257  0.20893
## L             1.4691     0.5116   2.872  0.00408 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.18  on 499  degrees of freedom
## Residual deviance: 509.08  on 497  degrees of freedom
## AIC: 515.08
##
## Number of Fisher Scoring iterations: 6
```

SAR-probit

```r
# Now we run a SAR-probit ------------------------------------------------------

# Dropping 'missing'
i <- which(rowSums(W)>0)
y <- y[i]
L <- L[i]

W <- methods::as(W[i,][,i], "dgCMatrix")

if (n <= 5e2) {
  sar_probit <- sarprobit(y~1, W = W, showProgress=FALSE)
  summary(sar_probit)

  # Controling for Latent variable (which we can't)
  sar_probitL <- sarprobit(y~L, W = W, showProgress=FALSE)
  summary(sar_probitL)
}
```

```
## Warning in sn2listw(df): 174 is not an origin
```

```
## --------MCMC spatial autoregressive probit--------
## Execution time  = 51.991 secs
##
## N draws          =   1000, N omit (burn-in)=    100
## N observations  =    497, K covariates    =      1
## # of 0 Y values =    260, # of 1 Y values =    237
## Min rho          = -1.000, Max rho          =  1.000
## --------------------------------------------------
##
##              Estimate  Std. Dev   p-level  t-value Pr(>|z|)
## (Intercept) -0.005307  0.044995  0.443000  -0.118    0.906
## rho           0.911665  0.034306  0.000000  26.575   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Warning in sn2listw(df): 174 is not an origin
```

```
## --------MCMC spatial autoregressive probit--------
## Execution time  = 52.161 secs
##
## N draws          =   1000, N omit (burn-in)=    100
## N observations  =    497, K covariates    =      2
## # of 0 Y values =    260, # of 1 Y values =    237
## Min rho          = -1.000, Max rho          =  1.000
## --------------------------------------------------
##
##             Estimate Std. Dev  p-level t-value Pr(>|z|)
## (Intercept) -0.04926  0.06778  0.23700  -0.727  0.46769
## L            0.53673  0.16482  0.00000   3.256  0.00121 **
## rho         -0.12036  0.41901  0.39400  -0.287  0.77405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

SAR

```r
# Running a SAR model ----------------------------------------------------------

summary(lagsarlm(y~1, listw = mat2listw(W), zero.policy = TRUE))
```

```
## Warning in sn2listw(df): 174 is not an origin
```

```
##
## Call:lagsarlm(formula = y ~ 1, listw = mat2listw(W), zero.policy = TRUE)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.856565 -0.343007 -0.070679  0.357768  0.929321
##
## Type: lag
## Regions with no neighbours included:
##  174
## Coefficients: (asymptotic standard errors)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.070679   0.032111  2.2011  0.02773
##
## Rho: 0.85733, LR test value: 153.43, p-value: < 2.22e-16
## Asymptotic standard error: 0.052464
##     z-value: 16.341, p-value: < 2.22e-16
## Wald statistic: 267.04, p-value: < 2.22e-16
##
## Log likelihood: -283.4698 for lag model
## ML residual variance (sigma squared): 0.17854, (sigma: 0.42254)
## Number of observations: 497
## Number of parameters estimated: 3
## AIC: 572.94, (AIC for lm: 724.37)
## LM test for residual autocorrelation
## test value: 0.082133, p-value: 0.77443
```

```r
# Controling for Latent variable (which we can't)
summary(lagsarlm(y~L, listw = mat2listw(W), zero.policy = TRUE))
```

```
## Warning in sn2listw(df): 174 is not an origin
```

```
##
## Call:lagsarlm(formula = y ~ L, listw = mat2listw(W), zero.policy = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99340 -0.33335  0.03691  0.33370  0.93012
##
## Type: lag
## Regions with no neighbours included:
##  174
## Coefficients: (asymptotic standard errors)
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) 0.362260   0.078055  4.6411 3.466e-06
## L           0.108863   0.022998  4.7336 2.206e-06
##
```

```
## Rho: 0.26523, LR test value: 2.2974, p-value: 0.12959
## Asymptotic standard error: 0.15522
##     z-value: 1.7087, p-value: 0.087507
## Wald statistic: 2.9196, p-value: 0.087507
##
## Log likelihood: -271.0382 for lag model
## ML residual variance (sigma squared): 0.17403, (sigma: 0.41717)
## Number of observations: 497
## Number of parameters estimated: 4
## AIC: 550.08, (AIC for lm: 550.37)
## LM test for residual autocorrelation
## test value: 0.1308, p-value: 0.71761
```