Package 'CDatanet'

February 17, 2021

Type Package

Title Modeling Count Data with Peer Effects

Version 0.0.1

Date 2021-02-08

Description

Likelihood-based estimation and data generation from a class of models used to estimate peer effects on count data by controlling for the network endogeneity. This class includes count data models with social interactions (Houndetoungan 2020; <doi:10.2139/ssrn.3721250>), spatial tobit models (Xu and Lee 2015; <doi:10.1016/j.jeconom.2015.05.004>), and spatial linear-in-means models (Lee 2004; <doi:10.1111/j.1468-0262.2004.00558.x>).

License GPL-3

Encoding UTF-8

BugReports https://github.com/ahoundetoungan/CDatanet/issues

URL https://github.com/ahoundetoungan/CDatanet

LazyData true

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), Formula, formula.tools, ddpcr, Matrix

LinkingTo Rcpp, RcppArmadillo, RcppProgress

RoxygenNote 7.1.1

Suggests ggplot2, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Elysée Aristide Houndetoungan [cre, aut]

Maintainer Elysée Aristide Houndetoungan <ariel92and@gmail.com>

Repository CRAN

Date/Publication 2021-02-17 20:00:03 UTC

R topics documented:

`CDatanet-package` *The CDatanet package*

Description

The **CDatanet** package implements the count data model with social interactions and the dyadic linking model developed in Houndetoungan (2020). It also simulates data from the count data model and implements the Spatial Autoregressive Tobit model (LeSage, 2000; Xu and Lee, 2015) for left censored data and the Spatial Autoregressive Model (Lee, 2004). To make the computations faster **CDatanet** uses `C++` through the **Rcpp** package (Eddelbuettel et al., 2011).

Author(s)

Maintainer: Elysée Aristide Houndetoungan `<ariel92and@gmail.com>`

References

Atchade, Y. F., & Rosenthal, J. S. (2005). On adaptive markov chain monte carlo algorithms, *Bernoulli*, 11(5), 815-828, doi: 10.3150/bj/1130077595

Eddelbuettel, D., François, R., Allaire, J., Ushey, K., Kou, Q., Russel, N., ... & Bates, D., 2011, **Rcpp**: Seamless R and `C++` integration, *Journal of Statistical Software*, 40(8), 1-18, doi: 10.18637/jss.v040.i08

Houndetoungan, E. A., 2020, A Count Data Model with Social Interactions. Available at SSRN 3721250, doi: 10.2139/ssrn.3721250

Lee, L. F., 2004, Asymptotic distributions of quasi-maximum likelihood estimators for spatial autoregressive models. Econometrica, 72(6), 1899-1925, doi: 10.1111/j.14680262.2004.00558.x

LeSage, J. P., 2000, Bayesian estimation of limited dependent variable spatial autoregressive models, *Geographical Analysis*, 32(1), 19-35, doi: 10.1111/j.15384632.2000.tb00413.x

Xu, X., & Lee, L. F., 2015, Maximum likelihood estimation of a spatial autoregressive Tobit model, Journal of Econometrics, 188(1), 264-280, doi: 10.1016/j.jeconom.2015.05.004

See Also

simCDnet, CDnetNPL, SARML, SARTML and netformation.

CDnetNPL *Estimate Count Data Model with Social Interactions using NPL Method*

Description

Estimate Count Data Model with Social Interactions using NPL Method

Usage

```
CDnetNPL(
  formula,
  contextual,
  Glist,
  theta0 = NULL,
  yb0 = NULL,
  optimizer = "optim",
  npl.ctr = list(),
  opt.ctr = list(),
  data
)
```

Arguments

formula	an object of class formula: a symbolic description of the model. The formula should be as for example y ~ x1 + x2 | x1 + x2 where y is the endogenous vector, the listed variables before the pipe, x1, x2 are the individual exogenous variables and the listed variables after the pipe, x1, x2 are the contextual observable variables. Other formulas may be y ~ x1 + x2 for the model without contextual effects, y ~ -1 + x1 + x2 | x1 + x2 for the model without intercept or y ~ x1 + x2 | x2 + x3 to allow the contextual variable to be different from the individual variables.
contextual	(optional) logical; if true, this means that all individual variables will be set as contextual variables. Set the the formula as y ~ x1 + x2 and contextual as TRUE is equivalent to set the formula as y ~ x1 + x2 | x1 + x2.
Glist	the adjacency matrix or list sub-adjacency matrix.
theta0	(optional) starting value of $\theta = (\lambda, \beta, \gamma, \sigma)$. The parameter γ should be removed if the model does not contain contextual effects (see details).
yb0	(optional) expectation of y.
optimizer	is either nlm (referring to the nlm function) or optim (referring to the optim function). At every step of the NPL method, the estimation is performed using nlm or optim. Other arguments of these functions such as, control and method can be defined through the argument opt.ctr.

`npl.ctr`	list of controls for the NPL method (see details).
`opt.ctr`	list of arguments of nlm or optim (the one set in `optimizer`) such as control, method, ...
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `CDnetNPL` is called.

Details

Model:

Following Houndetoungan (2020), the count data $\mathbf{y}$ is generated from a latent variable $\mathbf{y}^*$. The latent variable is given for all i as

$$y_i^* = \lambda \mathbf{g}_i \bar{\mathbf{y}} + \mathbf{x}_i' \beta + \mathbf{g}_i \mathbf{X} \gamma + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$.
The count variable y_i is then define by the next (greater or equal) non negative integer to y_i^*; that is $y_i = 0$ if $y_i^* \leq 0$ and $y_i = q + 1$ if $q < y_i^* \leq q + 1$, where q is a non-negative integer.

`npl.ctr`**:**

The model parameters is estimated using the Nested Partial Likelihood (NPL) method. This approach starts with a guess of θ and $\bar{y}$ and constructs iteratively a sequence of θ and $\bar{y}$. The solution converges when the L_1 distance between two consecutive θ and $\bar{y}$ is less than a tolerance. The argument `npl.ctr` is an optional list which contain

- tol the tolerance of the NPL algorithm (default 1e-4),
- maxit the maximal number of iterations allowed (default 500),
- print a boolean indicating if the estimate should be printed at each step.

`codedata`**:**

The class of the output of this function is `CDnetNPL`. This class has a summary and print methods to summarize and print the results. The adjacency matrix and the data are needed to summarize the results. However, in order to save memory, the function does not return these objects. Instead, it returns `codedata` which contains among others, the `formula` and the names of these objects passed through the argument `Glist` and `data` (if provided). `codedata` will be used to get access to the adjacency matrix and the data. Therefore, it is important to have the adjacency matrix and the data (or the variables) available in `.GlobalEnv`. Otherwise, it will be necessary to provide them to the summary function.

Value

A list consisting of:

`M`	number of sub-networks.
`n`	number of individuals in each network.
`iteration`	number of iterations performed by the NPL algorithm.
`estimate`	NPL estimator.

likelihood	pseudo-likelihood value.
yb	ybar (see details), expectation of y.
Gyb	average of the expectation of y among friends.
steps	step-by-step output as returned by the optimizer.
codedata	list of formula, name of the object Glist, number of friends in the network and name of the object data (see details).

See Also

simCDnet, SARML and SARTML.

Examples

```
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
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
X      <- cbind(rnorm(n, 1, 1), 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
}


# data
data    <- data.frame(x1 = X[,1], x2 =  X[,2])

rm(list = ls()[!(ls() %in% c("Glist", "data", "theta"))])
```

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

y       <- ytmp$y

# plot histogram
hist(y, breaks = max(y))

data    <- data.frame(yt = y, x1 = data$x1, x2 = data$x2)
rm(list = ls()[!(ls() %in% c("Glist", "data"))])

out   <- CDnetNPL(formula = yt ~ x1 + x2, contextual = TRUE, Glist = Glist, data = data)
summary(out)
```

`netformation` *Estimate Network Formation Model*

Description

Estimate Network Formation Model

Usage

```
netformation(
  network,
  formula,
  data,
  fixed.effects = TRUE,
  init = list(),
  mcmc.ctr = list(),
  print = TRUE
)
```

Arguments

`network`	matrix or list of sub-matrix of social interactions containing 0 and 1, where links are represented by 1
`formula`	an object of class formula: a symbolic description of the model. The `formula` should be as for example `~ x1 + x2` where `x1`, `x2` are explanatory variable of links formation
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `netformation` is called.
`fixed.effects`	boolean indicating if sub-network heterogeneity as fixed effects should be included.

`init`	(optional) list of starting values containing `beta`, an K-dimensional vector of the explanatory variables parameter, `mu` an n-dimensional vector of unobserved parameters, `sigmau2` the vector of the variances of `mu` in each sub-network (or single variance if `fixed.effects = FALSE`) and `uu` the vector of the of the means of `mu` in each sub-network (or single mean if `fixed.effects = FALSE`), where K is the number of explanatory variables and n is the number of individuals.
`mcmc.ctr`	(optional) list of MCMC control (see detail).
`print`	boolean indicating if the estimation progression should be printed.

Details

The network formation model can be used to control the network endogeneity in the count data, Tobit, and SAR models (see the vignette for an example with the count data model).
The MCMC control `mcmc.ctr` should be a list containing

- jscalebeta a K-dimensional vector of `beta` jumping scales.
- jscalemu an n-dimensional vector of `mu` jumping scales.
- burnin the number of iterations in the burn-in.
- iteration the number of simulations.
- tbeta target of `beta`.
- tmu target of `mu`.

The burn-in is replicated three times. The estimation is performed for each component of `beta` during the two firsts burn-in. The simulation from the second burn-in are used to compute covariance of `beta`. From the third burn-in and the remaining steps of the MCMC, all the components in `beta` are jointly simulated.
As `mu` dimension is large, the simulation is performed for each component.
The jumping scale are also updated during the MCMC following Atchadé and Rosenthal (2005).

Value

A list consisting of:

`n`	number of individuals in each network.
`n.obs`	number of observations.
`n.links`	number of links.
`K`	number of explanatory variables.
`posterior`	list of simulations from the posterior distribution and the posterior density.
`acceptance.rate`	acceptance rate of beta and mu.
`mcmc.ctr`	returned list of MCMC control.
`init`	returned list of starting values.

Examples

```
M            <- 5 # Number of sub-groups
nvec         <- round(runif(M, 100, 500))
beta         <- c(1, -1)
Glist        <- list()
dX           <- matrix(0, 0, 2)
mu           <- list()
uu           <- runif(M, -5, 5)
sigma2u      <- runif(M, 0.5, 16)
for (m in 1:M) {
  n          <- nvec[m]
  mum        <- rnorm(n, uu[m], sqrt(sigma2u[m]))
  X1         <- rnorm(n)
  X2         <- rbinom(n, 1, 0.2)
  Z1         <- matrix(0, n, n)
  Z2         <- matrix(0, n, n)

  for (i in 1:n) {
    for (j in 1:n) {
      Z1[i, j] <- abs(X1[i] - X1[j])
      Z2[i, j] <- 1*(X2[i] == X2[j])
    }
  }

  Gm           <- 1*((Z1*beta[1] + Z2*beta[2] +
                        kronecker(mum, t(mum), "+") + rlogis(n^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))
  Glist[[m]]   <- Gm
  mu[[m]]      <- mum
}

mu  <- unlist(mu)
out <- netformation(network =  Glist, formula = ~ dX, fixed.effects = T,
                    mcmc.ctr = list(burin = 1000, iteration = 5000))


# plot simulations
plot(out$posterior$beta[,1], type = "l")
abline(h = beta[1], col = "red")
plot(out$posterior$beta[,2], type = "l")
abline(h = beta[2], col = "red")

k <- 2
plot(out$posterior$sigmamu2[,2], type = "l")
```

```
abline(h = sigma2u[2], col = "red")


i <- 10
plot(out$posterior$mu[,i], type = "l",
     ylim = c(min(out$posterior$mu[,i], mu[i]), max(out$posterior$mu[,i], mu[i])))
abline(h = mu[i], col = "red")

plot(out$posterior$uu[,k] , type = "l",
     ylim = c(min(out$posterior$uu[,k], uu[k]), max(out$posterior$uu[,k], uu[k])))
abline(h = uu[k], col = "red")
```

SARML *Estimate SAR model*

Description

Estimate SAR model

Usage

```
SARML(
  formula,
  contextual,
  Glist,
  lambda0 = NULL,
  optimizer = "optim",
  opt.ctr = list(),
  print = TRUE,
  cov = TRUE,
  data
)
```

Arguments

formula	an object of class formula: a symbolic description of the model. The formula should be as for example y ~ x1 + x2 | x1 + x2 where y is the endogenous vector, the listed variables before the pipe, x1, x2 are the individual exogenous variables and the listed variables after the pipe, x1, x2 are the contextual observable variables. Other formulas may be y ~ x1 + x2 for the model without contextual effects, y ~ -1 + x1 + x2 | x1 + x2 for the model without intercept or y ~ x1 + x2 | x2 + x3 to allow the contextual variable to be different from the individual variables.
contextual	(optional) logical; if true, this means that all individual variables will be set as contextual variables. Set the formula as y ~ x1 + x2 and contextual as TRUE is equivalent to set the formula as y ~ x1 + x2 | x1 + x2.
Glist	the adjacency matrix or list sub-adjacency matrix.

`lambda0`	(optional) starting value of λ. The parameter γ should be removed if the model does not contain contextual effects (see details).
`optimizer`	is either `nlm` (referring to the function nlm) or `optim` (referring to the function optim). Other arguments of these functions such as, the control values and the method can be defined through the argument `opt.ctr`.
`opt.ctr`	list of arguments of nlm or optim (the one set in `optimizer`) such as control, method, ...
`print`	a boolean indicating if the estimate should be printed at each step.
`cov`	a boolean indicating if the covariance should be computed.
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `mcmcARD` is called.

Details

Model:

The variable $\mathbf{y}$ is given for all i as

$$y_i = \lambda \mathbf{g}_i y + \mathbf{x}_i'\beta + \mathbf{g}_i \mathbf{X}\gamma + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$.

`codedata`**:**

The class of the output of this function is `SAR`. This class has a summary and print methods to summarize and print the results. In order to save memory, the function does not return neither the adjacency matrix nor the data. Instead, it returns `codedata` which contains among others, the `formula` and the name of the adjacency matrix passed through the argument `Glist`.

Value

A list consisting of:

`M`	number of sub-networks.
`n`	number of individuals in each network.
`estimate`	Maximum Likelihood (ML) estimator.
`likelihood`	likelihood value.
`cov`	covariance matrix of the estimate.
`optimization`	output as returned by the optimizer.
`codedata`	list of formula, name of the object `Glist`, number of friends in the network and name of the object `data` (see details).

See Also

`CDnetNPL` and `SARTML`.

Examples

```
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
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
X      <- cbind(rnorm(n, 1, 1), 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
}


# data
data    <- data.frame(x1 = X[,1], x2 =  X[,2])

rm(list = ls()[!(ls() %in% c("Glist", "data", "theta"))])

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

y       <- ytmp$y

# plot histogram
hist(y, breaks = max(y))

data    <- data.frame(yt = y, x1 = data$x1, x2 = data$x2)
rm(list = ls()[!(ls() %in% c("Glist", "data"))])

out     <- SARML(formula = yt ~ x1 + x2, contextual = TRUE,
                 Glist = Glist, optimizer = "optim", data = data)
```

```
summary(out)
```

SARTML *Estimate SART model*

Description

Estimate SART model

Usage

```
SARTML(
  formula,
  contextual,
  Glist,
  theta0 = NULL,
  optimizer = "optim",
  opt.ctr = list(),
  print = TRUE,
  cov = TRUE,
  data
)
```

Arguments

`formula`	an object of class formula: a symbolic description of the model. The `formula` should be as for example `y ~ x1 + x2 | x1 + x2` where `y` is the endogenous vector, the listed variables before the pipe, `x1`, `x2` are the individual exogenous variables and the listed variables after the pipe, `x1`, `x2` are the contextual observable variables. Other formulas may be `y ~ x1 + x2` for the model without contextual effects, `y ~ -1 + x1 + x2 | x1 + x2` for the model without intercept or `y ~ x1 + x2 | x2 + x3` to allow the contextual variable to be different from the individual variables.
`contextual`	(optional) logical; if true, this means that all individual variables will be set as contextual variables. Set the `formula` as `y ~ x1 + x2` and `contextual` as `TRUE` is equivalent to set the formula as `y ~ x1 + x2 | x1 + x2`.
`Glist`	the adjacency matrix or list sub-adjacency matrix.
`theta0`	(optional) starting value of $\theta = (\lambda, \beta, \gamma, \sigma)$. The parameter γ should be removed if the model does not contain contextual effects (see details).
`optimizer`	is either `nlm` (referring to the function nlm) or `optim` (referring to the function optim). Other arguments of these functions such as, the control values and the method can be defined through the argument `opt.ctr`.
`opt.ctr`	list of arguments of nlm or optim (the one set in `optimizer`) such as control, method, ...
`print`	a boolean indicating if the estimate should be printed at each step.

cov	a boolean indicating if the covariance should be computed.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which SARTML is called.

Details

Model:

The left-censored variable $\mathbf{y}$ is generated from a latent variable $\mathbf{y}^*$. The latent variable is given for all i as

$$y_i^* = \lambda \mathbf{g}_i y + \mathbf{x}_i'\beta + \mathbf{g}_i \mathbf{X}\gamma + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$.
The count variable y_i is then define that is $y_i = 0$ if $y_i^* \leq 0$ and $y_i = y_i^*$ otherwise.

codedata:

The class of the output of this function is SARTML. This class has a summary and print methods to summarize and print the results. The adjacency matrix is needed to summarize the results. However, in order to save memory, the function does not return it. Instead, it returns codedata which contains the formula and the name of the adjacency matrix passed through the argument Glist. codedata will be used to get access to the adjacency matrix. Therefore, it is important to have the adjacency matrix available in .GlobalEnv. Otherwise it will be necessary to provide the adjacency matrix to the summary and print functions.

Value

A list consisting of:

M	number of sub-networks.
n	number of individuals in each network.
estimate	Maximum Likelihood (ML) estimator.
likelihood	likelihood value.
cov	covariance matrix of the estimate.
optimization	output as returned by the optimizer.
codedata	list of formula, name of the object Glist, number of friends in the network, name of the object data, and number of zeros in y (see details).

See Also

CDnetNPL and SARML.

Examples

```
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
```

```
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
X      <- cbind(rnorm(n, 1, 1), 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
}


# data
data    <- data.frame(x1 = X[,1], x2 =  X[,2])

rm(list = ls()[!(ls() %in% c("Glist", "data", "theta"))])

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

y       <- ytmp$y

# plot histogram
hist(y)

opt.ctr <- list(method  = "Nelder-Mead",
                control = list(abstol = 1e-16, abstol = 1e-11, maxit = 5e3))
data    <- data.frame(yt = y, x1 = data$x1, x2 = data$x2)
rm(list = ls()[!(ls() %in% c("Glist", "data"))])

out     <- SARTML(formula = yt ~ x1 + x2, optimizer = "nlm",
                  contextual = TRUE, Glist = Glist, data = data)
summary(out)
```

`simCDnet` *Simulate data from Count Data Model with Social Interactions*

Description

Simulate data from Count Data Model with Social Interactions

Usage

```
simCDnet(formula, contextual, Glist, theta, tol = 1e-15, maxit = 500, data)
```

Arguments

`formula`	an object of class formula: a symbolic description of the model. The `formula` should be as for example `y ~ x1 + x2 | x1 + x2` where `y` is the endogenous vector, the listed variables before the pipe, `x1`, `x2` are the individual exogenous variables and the listed variables after the pipe, `x1`, `x2` are the contextual observable variables. Other formulas may be `y ~ x1 + x2` for the model without contextual effects, `y ~ -1 + x1 + x2 | x1 + x2` for the model without intercept or `y ~ x1 + x2 | x2 + x3` to allow the contextual variable to be different from the individual variables.
`contextual`	(optional) logical; if true, this means that all individual variables will be set as contextual variables. Set the `formula` as `y ~ x1 + x2` and `contextual` as `TRUE` is equivalent to set the formula as `y ~ x1 + x2 | x1 + x2`.
`Glist`	the adjacency matrix or list sub-adjacency matrix.
`theta`	the parameter value as $\theta = (\lambda, \beta, \gamma, \sigma)$. The parameter γ should be removed if the model does not contain contextual effects (see details).
`tol`	the tolerance value used in the Fixed Point Iteration Method to compute the expectancy of `y`. The process stops if the L_1 distance between two consecutive values of the expectancy of `y` is less than `tol`.
`maxit`	the maximal number of iterations in the Fixed Point Iteration Method.
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `mcmcARD` is called.

Details

Following Houndetoungan (2020), the count data $\mathbf{y}$ is generated from a latent variable $\mathbf{y}^*$. The latent variable is given for all i as

$$y_i^* = \lambda \mathbf{g}_i \bar{\mathbf{y}} + \mathbf{x}_i' \beta + \mathbf{g}_i \mathbf{X} \gamma + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$.
The count variable y_i is then define by the next (greater or equal) non negative integer to y_i^*; that is $y_i = 0$ if $y_i^* \leq 0$ and $y_i = q + 1$ if $q < y_i^* \leq q + 1$, where q is a non-negative integer.

Value

A list consisting of:

`yst`	ys (see details), the latent variable.
`y`	the observed count data.
`yb`	ybar (see details), the expectation of y.
`Gyb`	the average of the expectation of y among friends.
`iteration`	number of iterations performed by sub-network in the Fixed Point Iteration Method.

See Also

`CDnetNPL`.

Examples

```
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
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
X      <- cbind(rnorm(n, 1, 1), 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
}


# data
data    <- data.frame(x1 = X[,1], x2 =  X[,2])
```

```
rm(list = ls()[!(ls() %in% c("Glist", "data", "theta"))])

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

y       <- ytmp$y

# plot histogram
hist(y, breaks = max(y))
```

---

simSARnet *Simulate data from the linear-in-mean Model with Social Interactions*

Description

Simulate data from the linear-in-mean Model with Social Interactions

Usage

```
simSARnet(formula, contextual, Glist, theta, data)
```

Arguments

formula	an object of class formula: a symbolic description of the model. The formula should be as for example y ~ x1 + x2 | x1 + x2 where y is the endogenous vector, the listed variables before the pipe, x1, x2 are the individual exogenous variables and the listed variables after the pipe, x1, x2 are the contextual observable variables. Other formulas may be y ~ x1 + x2 for the model without contextual effects, y ~ -1 + x1 + x2 | x1 + x2 for the model without intercept or y ~ x1 + x2 | x2 + x3 to allow the contextual variable to be different from the individual variables.
contextual	(optional) logical; if true, this means that all individual variables will be set as contextual variables. Set the formula as y ~ x1 + x2 and contextual as TRUE is equivalent to set the formula as y ~ x1 + x2 | x1 + x2.
Glist	the adjacency matrix or list sub-adjacency matrix.
theta	the parameter value as $\theta = (\lambda, \beta, \gamma, \sigma)$. The parameter γ should be removed if the model does not contain contextual effects (see details).
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which mcmcARD is called.

Details

The variable $\mathbf{y}$ is given for all i as

$$y_i = \lambda \mathbf{g}_i y + \mathbf{x}_i'\beta + \mathbf{g}_i \mathbf{X}\gamma + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$.

Value

A list consisting of:

y	the observed count data.
Gy	the average of y among friends.

See Also

CDnetNPL.

Examples

```
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
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
X      <- cbind(rnorm(n, 1, 1), 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
}
```

```
# data
data    <- data.frame(x1 = X[,1], x2 =  X[,2])

rm(list = ls()[!(ls() %in% c("Glist", "data", "theta"))])

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

# plot histogram
hist(y)
```

`simTobitnet` *Simulate data from the Tobit Model with Social Interactions*

Description

Simulate data from the Tobit Model with Social Interactions

Usage

```
simTobitnet(formula, contextual, Glist, theta, tol = 1e-15, maxit = 500, data)
```

Arguments

`formula`	an object of class formula: a symbolic description of the model. The `formula` should be as for example `y ~ x1 + x2 | x1 + x2` where `y` is the endogenous vector, the listed variables before the pipe, `x1`, `x2` are the individual exogenous variables and the listed variables after the pipe, `x1`, `x2` are the contextual observable variables. Other formulas may be `y ~ x1 + x2` for the model without contextual effects, `y ~ -1 + x1 + x2 | x1 + x2` for the model without intercept or `y ~ x1 + x2 | x2 + x3` to allow the contextual variable to be different from the individual variables.
`contextual`	(optional) logical; if true, this means that all individual variables will be set as contextual variables. Set the `formula` as `y ~ x1 + x2` and `contextual` as `TRUE` is equivalent to set the formula as `y ~ x1 + x2 | x1 + x2`.
`Glist`	the adjacency matrix or list sub-adjacency matrix.
`theta`	the parameter value as $\theta = (\lambda, \beta, \gamma, \sigma)$. The parameter γ should be removed if the model does not contain contextual effects (see details).
`tol`	the tolerance value used in the Fixed Point Iteration Method to compute `y`. The process stops if the L_1 distance between two consecutive values of `y` is less than `tol`.
`maxit`	the maximal number of iterations in the Fixed Point Iteration Method.
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `mcmcARD` is called.

Details

The left-censored variable $\mathbf{y}$ is generated from a latent variable $\mathbf{y}^*$. The latent variable is given for all i as

$$y_i^* = \lambda \mathbf{g}_i y + \mathbf{x}_i'\beta + \mathbf{g}_i \mathbf{X}\gamma + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$.
The count variable y_i is then define that is $y_i = 0$ if $y_i^* \leq 0$ and $y_i = y_i^*$ otherwise.

Value

A list consisting of:

`yst`	ys (see details), the latent variable.
`y`	the observed count data.
`Gy`	the average of y among friends.
`iteration`	number of iterations performed by sub-network in the Fixed Point Iteration Method.

See Also

CDnetNPL.

Examples

```
# Groups' size
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
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
X      <- cbind(rnorm(n, 1, 1), 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
}


# data
data    <- data.frame(x1 = X[,1], x2 =  X[,2])

rm(list = ls()[!(ls() %in% c("Glist", "data", "theta"))])

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

y       <- ytmp$y

# plot histogram
hist(y)
```

---

summary.CDnetNPL *Summarize Count Data Model with Social Interactions*

Description

Summary and print methods for the class CDnetNPL as returned by the function CDnetNPL.

Usage

```
## S3 method for class 'CDnetNPL'
summary(object, cov.ctr = list(), Glist, data, ...)

## S3 method for class 'summary.CDnetNPL'
print(x, ...)

## S3 method for class 'CDnetNPL'
print(x, ...)

## S3 method for class 'summary.CDnetNPLs'
print(x, ...)
```

Arguments

object	an object of class CDnetNPL, output of the function CDnetNPL.
cov.ctr	list of control values for the covariance containing two integers, R and S. The covariance summations from 0 to infinity. But the summed elements decreases exponentially. The summations are approximated by summations from 0 to R. The covariance also requires computing $\Phi(x)-\Phi(x-1)$, where Φ is the normal' probability density function. This is done using important sampling, where S numbers are generated form the uniform distribution.

Glist	the adjacency matrix or list sub-adjacency matrix. If missing make, sure that the object provided to the function CDnetNPL is available in .GlobalEnv (see detail - codedata section of CDnetNPL).
data	dataframe containing the explanatory variables. If missing make, sure that the object provided to the function CDnetNPL is available in .GlobalEnv (see detail - codedata section of CDnetNPL).
...	further arguments passed to or from other methods.
x	an object of class summary.CDnetNPL, output of the function summary.CDnetNPL, class summary.CDnetNPLs, list of outputs of the function summary.CDnetNPL (when the model is estimated many times to control for the endogeneity) or class CDnetNPL of the function CDnetNPL.

Value

A list consisting of:

M	number of sub-networks.
n	number of individuals in each network.
iteration	number of iterations performed by the NPL algorithm.
estimate	NPL estimator.
likelihood	pseudo-likelihood value.
yb	ybar (see details), expectation of y.
Gyb	average of the expectation of y among friends.
steps	step-by-step output as returned by the optimizer.
cov	covariance matrix of the estimate.
meffects	vector of marginal effects.
cov.me	covariance matrix of the marginal effects.
cov.ctr	returned value of the control values for the covariance.
codedata	list of formula, name of the object Glist, number of friends in the network and name of the object data.

---

summary.SARML *Summarize SAR Model*

Description

Summary and print methods for the class SARML as returned by the function SARML.

Usage

```
## S3 method for class 'SARML'
summary(object, ...)

## S3 method for class 'summary.SARML'
print(x, ...)

## S3 method for class 'SARML'
print(x, ...)

## S3 method for class 'summary.SARMLs'
print(x, ...)
```

Arguments

object	an object of class SARML, output of the function SARML.
...	further arguments passed to or from other methods.
x	an object of class summary.SARML, output of the function summary.SARML or class SARML, output of the function SARML.

Value

A list consisting of:

M	number of sub-networks.
n	number of individuals in each network.
estimate	Maximum Likelihood (ML) estimator.
likelihood	likelihood value.
cov	covariance matrix of the estimate.
optimization	output as returned by the optimizer.
codedata	list of formula, name of the object Glist, number of friends in the network and name of the object data.

---

summary.SARTML *Summarize SART Model*

Description

Summary and print methods for the class SARTML as returned by the function SARTML.

Usage

```
## S3 method for class 'SARTML'
summary(object, Glist, data, ...)

## S3 method for class 'summary.SARTML'
print(x, ...)

## S3 method for class 'SARTML'
print(x, ...)

## S3 method for class 'summary.SARTMLs'
print(x, ...)
```

Arguments

object	an object of class SARTML, output of the function SARTML.
Glist	the adjacency matrix or list sub-adjacency matrix. If missing make, sure that the object provided to the function SARTML is available in .GlobalEnv (see detail - codedata section of SARTML).
data	dataframe containing the explanatory variables. If missing make, sure that the object provided to the function SARTML is available in .GlobalEnv (see detail - codedata section of SARTML).
...	further arguments passed to or from other methods.
x	an object of class summary.SARTML, output of the function summary.SARTML or class SARTML, output of the function SARTML.

Value

A list consisting of:

M	number of sub-networks.
n	number of individuals in each network.
estimate	Maximum Likelihood (ML) estimator.
likelihood	likelihood value.
cov	covariance matrix of the estimate.
optimization	output as returned by the optimizer.
codedata	list of formula, name of the object Glist, number of friends in the network, name of the object data, and number of zeros in y.

Index