Package 'CDatanet'

May 7, 2026

Type Package

Title Econometrics of Network Data

Version 2.2.2

Date 2025-12-01

Description Simulating and estimating peer effect models and network formation models. The class of peer effect models includes linear-in-means models (Lee, 2004; <doi:10.1111/j.1468-0262.2004.00558.x>), Tobit models (Xu and Lee, 2015; <doi:10.1016/j.jeconom.2015.05.004>), and discrete numerical data models (Houndetoungan, 2025; <doi:10.48550/arXiv.2405.17290>). The network formation models include pair-wise regressions with degree heterogeneity (Graham, 2017; <doi:10.3982/ECTA12679>) and exponential random graph models (Mele, 2017; <doi:10.3982/ECTA10400>).

License GPL-3

Language en-US

Encoding UTF-8

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

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

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), Formula, formula.tools, Matrix, matrixcalc, foreach, doRNG, doParallel, parallel

LinkingTo Rcpp, RcppArmadillo, RcppProgress, RcppDist, RcppNumerical, RcppEigen

RoxygenNote 7.3.2

Suggests ggplot2, MASS, knitr, rmarkdown

NeedsCompilation yes

Author Aristide Houndetoungan [cre, aut]

Maintainer Aristide Houndetoungan <ahoundetoungan@ecn.ulaval.ca>

Repository CRAN

Date/Publication 2025-11-09 20:10:06 UTC

Contents

`CDatanet-package` *The CDatanet Package*

Description

The **CDatanet** package simulates and estimates peer effect models and network formation models. The peer effect models include linear-in-means models (Lee, 2004; Lee et al., 2010), Tobit models (Xu and Lee, 2015), and discrete numerical data models (Houndetoungan, 2024). The network formation models include pairwise regressions with degree heterogeneity (Graham, 2017; Yan et al., 2019) and exponential random graph models (Mele, 2017). To enhance computation speed, **CDatanet** uses `C++` via the **Rcpp** package (Eddelbuettel et al., 2011).

Author(s)

Maintainer: Aristide Houndetoungan <ahoundetoungan@ecn.ulaval.ca>

References

Eddelbuettel, D., & Francois, R. (2011). **Rcpp**: Seamless R and `C++` integration. *Journal of Statistical Software*, 40(8), 1-18, doi:10.18637/jss.v040.i08.

Houndetoungan, E. A. (2025). Count Data Models with Heterogeneous Peer Effects. Available at arXiv:2405.17290, doi:10.48550/arXiv.2405.17290.

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.

Lee, L. F., Liu, X., & Lin, X. (2010). Specification and estimation of social interaction models with network structures. The Econometrics Journal, 13(2), 145-176, doi:10.1111/j.1368423X.2010.00310.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.

Graham, B. S. (2017). An econometric model of network formation with degree heterogeneity. *Econometrica*, 85(4), 1033-1063, doi:10.3982/ECTA12679.

Mele, A. (2017). A structural model of dense network formation. *Econometrica*, 85(3), 825-850, doi:10.3982/ECTA10400.

Yan, T., Jiang, B., Fienberg, S. E., & Leng, C. (2019). Statistical inference in a directed network model with covariates. *Journal of the American Statistical Association*, 114(526), 857-868, doi:10.1080/01621459.2018.1448829.

See Also

Useful links:

- https://github.com/ahoundetoungan/CDatanet
- Report bugs at https://github.com/ahoundetoungan/CDatanet/issues

---

`cdnet` *Estimating Count Data Models with Social Interactions under Rational Expectations Using the NPL Method*

Description

`cdnet` estimates count data models with social interactions under rational expectations using the NPL algorithm (see Houndetoungan, 2024).

Usage

```
cdnet(
  formula,
  Glist,
  group,
  Rmax,
  Rbar,
  starting = list(lambda = NULL, Gamma = NULL, delta = NULL),
  Ey0 = NULL,
  ubslambda = 1L,
  optimizer = "fastlbfgs",
  npl.ctr = list(),
  opt.ctr = list(),
  cov = TRUE,
  data
)
```

Arguments

formula	a class object formula: a symbolic description of the model. The formula must be, for example, y ~ x1 + x2 + gx1 + gx2, where y is the endogenous vector, and x1, x2, gx1, and gx2 are control variables, which may include contextual variables (i.e., averages among the peers). Peer averages can be computed using the function peer.avg.
Glist	adjacency matrix. For networks consisting of multiple subnets (e.g., schools), Glist can be a list of subnets, with the m-th element being an $n_m \times n_m$ adjacency matrix, where n_m is the number of nodes in the m-th subnet. For heterogeneous peer effects (i.e., when length(unique(group)) = h > 1), the m-th element must be a list of h^2 $n_m \times n_m$ adjacency matrices corresponding to the different network specifications (see Houndetoungan, 2024, Section 2.1). For heterogeneous peer effects in the case of a single large network (a single school), Glist must be a one-item list (since there is one school). This item must be a list of h^2 network specifications. The order in which the networks are specified is important and must match the order of the groups in sort(unique(group)) (see argument group and examples).
group	a vector indicating the individual groups. The default assumes a common group. For two groups, i.e., length(unique(group)) = 2 (e.g., A and B), four types of peer effects are defined: peer effects of A on A, of A on B, of B on A, and of B on B. In this case, in the argument Glist, the networks must be defined in this order: AA, AB, BA, BB.
Rmax	an integer indicating the theoretical upper bound of y (see model specification in detail).
Rbar	an L-vector, where L is the number of groups. For large Rmax, the cost function is assumed to be semi-parametric (i.e., nonparametric from 0 to $\bar{R}$ and quadratic beyond $\bar{R}$).
starting	(optional) a starting value for $\theta = (\lambda, \Gamma', \delta')$, where λ, Γ, and δ are the parameters to be estimated (see details).
Ey0	(optional) a starting value for $E(y)$.
ubslambda	a positive value indicating the upper bound of $\sum_{s=1}^{S} \lambda_s > 0$.
optimizer	specifies the optimization method, which can be one of: fastlbfgs (L-BFGS optimization method from the **RcppNumerical** package), nlm (from the function nlm), or optim (from the function optim). Arguments for these functions, such as control and method, can be set via the argument opt.ctr.
npl.ctr	a list of controls for the NPL method (see details).
opt.ctr	a list of arguments to be passed to optim_lbfgs from the **RcppNumerical** package, or to nlm or optim (the solver specified in optimizer), such as maxit, eps_f, eps_g, control, method, etc.
cov	a Boolean indicating whether the covariance should be computed.
data	an optional data frame, list, or environment (or an 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 cdnet is called.

Details

Model:

The count variable y_i takes the value r with probability.

$$P_{ir} = F(\sum_{s=1}^{S} \lambda_s \bar{y}_i^{e,s} + \mathbf{z}_i'\Gamma - a_{h(i),r}) - F(\sum_{s=1}^{S} \lambda_s \bar{y}_i^{e,s} + \mathbf{z}_i'\Gamma - a_{h(i),r+1}).$$

In this equation, $\mathbf{z}_i$ is a vector of control variables; F is the distribution function of the standard normal distribution; $\bar{y}_i^{e,s}$ is the average of $E(y)$ among peers using the `s`-th network definition; $a_{h(i),r}$ is the `r`-th cut-point in the cost group $h(i)$.

The following identification conditions have been introduced: $\sum_{s=1}^{S} \lambda_s > 0$, $a_{h(i),0} = -\infty$, $a_{h(i),1} = 0$, and $a_{h(i),r} = \infty$ for any $r \geq R_{\text{max}} + 1$. The last condition implies that $P_{ir} = 0$ for any $r \geq R_{\text{max}} + 1$. For any $r \geq 1$, the distance between two cut-points is $a_{h(i),r+1} - a_{h(i),r} = \delta_{h(i),r} + \sum_{s=1}^{S} \lambda_s$. As the number of cut-points can be large, a quadratic cost function is considered for $r \geq \bar{R}_{h(i)}$, where $\bar{R} = (\bar{R}_1, ..., \bar{R}_L)$. With the semi-parametric cost function, $a_{h(i),r+1} - a_{h(i),r} = \bar{\delta}_{h(i)} + \sum_{s=1}^{S} \lambda_s$.

The model parameters are: $\lambda = (\lambda_1, ..., \lambda_S)'$, Γ, and $\delta = (\delta_1', ..., \delta_L')'$, where $\delta_l = (\delta_{l,2}, ..., \delta_{l,\bar{R}_l}, \bar{\delta}_l)'$ for $l = 1, ..., L$. The number of single parameters in δ_l depends on R_{max} and $\bar{R}_l$. The components $\delta_{l,2}, ..., \delta_{l,\bar{R}_l}$ or/and $\bar{\delta}_l$ must be removed in certain cases.
If $R_{\text{max}} = \bar{R}_l \geq 2$, then $\delta_l = (\delta_{l,2}, ..., \delta_{l,\bar{R}_l})'$.
If $R_{\text{max}} = \bar{R}_l = 1$ (binary models), then $\bar{\delta}_l$ must be empty.
If $R_{\text{max}} > \bar{R}_l = 1$, then $\delta_l = \bar{\delta}_l$.

`npl.ctr`:

The model parameters are estimated using the Nested Partial Likelihood (NPL) method. This approach begins with an initial guess for θ and $E(y)$ and iteratively refines them. The solution converges when the ℓ_1-distance between two consecutive estimates of θ and $E(y)$ is smaller than a specified tolerance.

The argument `npl.ctr` must include the following parameters:

tol the tolerance level for the NPL algorithm (default is 1e-4).

maxit the maximum number of iterations allowed (default is 500).

print a boolean value indicating whether the estimates should be printed at each step.

S the number of simulations performed to compute the integral in the covariance using importance sampling.

Value

A list consisting of:

`info`	a list containing general information about the model.
`estimate`	the NPL estimator.
`Ey`	$E(y)$, the expectation of y.
`GEy`	the average of $E(y)$ across peers.

cov	a list that includes (if cov == TRUE): parms, the covariance matrix, and another list, var.comp, which contains Sigma (Σ) and Omega (Ω), the matrices used to compute the covariance matrix.
details	step-by-step output returned by the optimizer.

References

Houndetoungan, A. (2024). Count Data Models with Heterogeneous Peer Effects. Available at SSRN 3721250, doi:10.2139/ssrn.3721250.

See Also

sart, sar, simcdnet.

Examples

```
set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 200))
n      <- sum(nvec)

# Adjacency matrix
A      <- list()
for (m in 1:M) {
  nm           <- nvec[m]
  Am           <- matrix(0, nm, nm)
  max_d        <- 30 #maximum number of friends
  for (i in 1:nm) {
    tmp        <- sample((1:nm)[-i], sample(0:max_d, 1))
    Am[i, tmp] <- 1
  }
  A[[m]]       <- Am
}
Anorm  <- norm.network(A) #Row-normalization

# X
X      <- cbind(rnorm(n, 1, 3), rexp(n, 0.4))

# Two group:
group  <- 1*(X[,1] > 0.95)

# Networks
# length(group) = 2 and unique(sort(group)) = c(0, 1)
# The networks must be defined as to capture:
# peer effects of `0` on `0`, peer effects of `1` on `0`
# peer effects of `0` on `1`, and peer effects of `1` on `1`
G        <- list()
cums     <- c(0, cumsum(nvec))
for (m in 1:M) {
  tp     <- group[(cums[m] + 1):(cums[m + 1])]
  Am     <- A[[m]]
  G[[m]] <- norm.network(list(Am * ((1 - tp) %*% t(1 - tp)),
```

```
                                Am * ((1 - tp) %*% t(tp)),
                                Am * (tp %*% t(1 - tp)),
                                Am * (tp %*% t(tp))))
}

# Parameters
lambda <- c(0.2, 0.3, -0.15, 0.25)
Gamma  <- c(4.5, 2.2, -0.9, 1.5, -1.2)
delta  <- rep(c(2.6, 1.47, 0.85, 0.7, 0.5), 2)

# Data
data   <- data.frame(X, peer.avg(Anorm, cbind(x1 = X[,1], x2 =  X[,2])))
colnames(data) = c("x1", "x2", "gx1", "gx2")

ytmp   <- simcdnet(formula = ~ x1 + x2 + gx1 + gx2, Glist = G, Rbar = rep(5, 2),
                   lambda = lambda, Gamma = Gamma, delta = delta, group = group,
                   data = data)
y      <- ytmp$y
hist(y, breaks = max(y) + 1)
table(y)

# Estimation
est   <- cdnet(formula = y ~ x1 + x2 + gx1 + gx2, Glist = G, Rbar = rep(5, 2), group = group,
               optimizer = "fastlbfgs", data = data,
               opt.ctr = list(maxit = 5e3, eps_f = 1e-11, eps_g = 1e-11))
summary(est)
```

`homophili.data` *Converting Data between Directed Network Models and Symmetric Network Models.*

Description

`homophili.data` converts the matrix of explanatory variables between directed network models and symmetric network models.

Usage

```
homophili.data(data, nvec, to = c("lower", "upper", "symmetric"))
```

Arguments

`data`	A `matrix` or `data.frame` of the explanatory variables of the network formation model. This corresponds to the `X` matrix in `homophily.fe` or `homophily.re`.
`nvec`	A vector of the number of individuals in the networks.
`to`	Indicates the direction of the conversion. For a matrix of explanatory variables `X` (`n*(n-1)` rows), one can select lower triangular entries (`to = "lower"`) or upper triangular entries (`to = "upper"`). For a triangular `X` (`n*(n-1)/2` rows),

one can convert to a full matrix of `n*(n-1)` rows by using symmetry (`to = "symmetric"`).

Value

The transformed `data.frame`.

`homophily.fe` *Estimating Network Formation Models with Degree Heterogeneity: the Fixed Effect Approach*

Description

`homophily.fe` implements a Logit estimator for a network formation model with homophily. The model includes degree heterogeneity using fixed effects (see details).

Usage

```
homophily.fe(
  network,
  formula,
  data,
  symmetry = FALSE,
  fe.way = 1,
  init = NULL,
  method = c("L-BFGS", "Block-NRaphson", "Mix"),
 ctr = list(maxit.opt = 10000, maxit.nr = 50, eps_f = 1e-09, eps_g = 1e-09, tol = 1e-04),
  print = TRUE
)
```

Arguments

`network`	A matrix or list of sub-matrices 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, for example, `~ x1 + x2`, where `x1` and `x2` are explanatory variables for link formation. If missing, the model is estimated with fixed effects only.
`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 `homophily` is called.
`symmetry`	Indicates whether the network model is symmetric (see details).
`fe.way`	Indicates whether it is a one-way or two-way fixed effect model. The expected value is 1 or 2 (see details).

`init`	(optional) Either a list of starting values containing `beta`, a K-dimensional vector of the explanatory variables' parameters, `mu`, an n-dimensional vector, and `nu`, an n-dimensional vector, where K is the number of explanatory variables and n is the number of individuals; or a vector of starting values for `c(beta, mu, nu)`.
`method`	A character string specifying the optimization method. Expected values are `"L-BFGS"`, `"Block-NRaphson"`, or `"Mix"`. `"Block-NRaphson"` refers to the `Newton-Raphson` method applied to each subnetwork, and `"Mix"` combines the `Newton-Raphson` method for `beta` with the `L-BFGS` method for the fixed effects.
`ctr`	(optional) A list containing control parameters for the solver. For the `optim_lbfgs` method from the **RcppNumerical** package, the list should include `maxit.opt` (corresponding to `maxit` for the `L-BFGS` method), `eps_f`, and `eps_g`. For the `Block-NRaphson` method, the list should include `maxit.nr` (corresponding to `maxit` for the `Newton-Raphson` method) and `tol`.
`print`	A boolean indicating if the estimation progression should be printed.

Details

Let p_{ij} be the probability for a link to go from individual i to individual j. This probability is specified for two-way effect models (`fe.way = 2`) as

$$p_{ij} = F(\mathbf{x}_{ij}'\beta + \mu_i + \nu_j),$$

where F is the cumulative distribution function of the standard logistic distribution. Unobserved degree heterogeneity is captured by μ_i and ν_j. These are treated as fixed effects (see `homophily.re` for random effect models). As shown by Yan et al. (2019), the estimator of the parameter β is biased. A bias correction is necessary but not implemented in this version. However, the estimators of μ_i and ν_j are consistent.

For one-way fixed effect models (`fe.way = 1`), $\nu_j = \mu_j$. For symmetric models, the network is not directed, and the fixed effects need to be one-way.

Value

A list consisting of:

`model.info`	A list of model information, such as the type of fixed effects, whether the model is symmetric, the number of observations, etc.
`estimate`	The maximizer of the log-likelihood.
`loglike`	The maximized log-likelihood.
`optim`	The returned value from the optimization solver, which contains details of the optimization. The solver used is `optim_lbfgs` from the **RcppNumerical** package.
`init`	The returned list of starting values.
`loglike.init`	The log-likelihood at the starting values.

References

Yan, T., Jiang, B., Fienberg, S. E., & Leng, C. (2019). Statistical inference in a directed network model with covariates. *Journal of the American Statistical Association*, 114(526), 857-868, doi:10.1080/01621459.2018.1448829.

See Also

homophily.re.

Examples

```
set.seed(1234)
M            <- 2 # Number of sub-groups
nvec         <- round(runif(M, 20, 50))
beta         <- c(.1, -.1)
Glist        <- list()
dX           <- matrix(0, 0, 2)
mu           <- list()
nu           <- list()
Emunu        <- runif(M, -1.5, 0) # Expectation of mu + nu
smu2         <- 0.2
snu2         <- 0.2
for (m in 1:M) {
  n          <- nvec[m]
  mum        <- rnorm(n, 0.7*Emunu[m], smu2)
  num        <- rnorm(n, 0.3*Emunu[m], snu2)
  X1         <- rnorm(n, 0, 1)
  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(num), "+") + 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
  nu[[m]]      <- num
}
```

```
mu  <- unlist(mu)
nu  <- unlist(nu)

out   <- homophily.fe(network =  Glist, formula = ~ -1 + dX, fe.way = 2)
muhat <- out$estimate$mu
nuhat <- out$estimate$nu
plot(mu, muhat)
plot(nu, nuhat)
```

`homophily.re` *Estimating Network Formation Models with Degree Heterogeneity: the Bayesian Random Effect Approach*

Description

`homophily.re` implements a Bayesian Probit estimator for network formation model with homophily. The model includes degree heterogeneity using random effects (see details).

Usage

```
homophily.re(
  network,
  formula,
  data,
  symmetry = FALSE,
  group.fe = FALSE,
  re.way = 1,
  init = list(),
  iteration = 1000,
  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 variables for 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 `homophily` is called.
`symmetry`	indicates whether the network model is symmetric (see details).
`group.fe`	indicates whether the model includes group fixed effects.

re.way	indicates whether it is a one-way or two-way random effect model. The expected value is 1 or 2 (see details).
init	(optional) list of starting values containing beta, a K-dimensional vector of the explanatory variables parameter, mu, an n-dimensional vector, and nu, an n-dimensional vector, smu2 the variance of mu, and snu2 the variance of nu, where K is the number of explanatory variables and n is the number of individuals.
iteration	the number of iterations to be performed.
print	boolean indicating if the estimation progression should be printed.

Details

Let p_{ij} be a probability for a link to go from the individual i to the individual j. This probability is specified for two-way effect models (re.way = 2) as

$$p_{ij} = F(\mathbf{x}'_{ij}\beta + \mu_i + \nu_j),$$

where F is the cumulative of the standard normal distribution. Unobserved degree heterogeneity is captured by μ_i and ν_j. The latter are treated as random effects (see homophily.fe for fixed effect models).
For one-way random effect models (re.way = 1), $\nu_j = \mu_j$. For symmetric models, the network is not directed and the random effects need to be one way.

Value

A list consisting of:

model.info	list of model information, such as the type of random effects, whether the model is symmetric, number of observations, etc.
posterior	list of simulations from the posterior distribution.
init	returned list of starting values.

See Also

homophily.fe.

Examples

```
set.seed(1234)
library(MASS)
M            <- 4 # Number of sub-groups
nvec         <- round(runif(M, 100, 500))
beta         <- c(.1, -.1)
Glist        <- list()
dX           <- matrix(0, 0, 2)
mu           <- list()
nu           <- list()
cst          <- runif(M, -1.5, 0)
smu2         <- 0.2
snu2         <- 0.2
rho          <- 0.8
```

```
Smunu        <- matrix(c(smu2, rho*sqrt(smu2*snu2), rho*sqrt(smu2*snu2), snu2), 2)
for (m in 1:M) {
  n          <- nvec[m]
  tmp        <- mvrnorm(n, c(0, 0), Smunu)
  mum        <- tmp[,1] - mean(tmp[,1])
  num        <- tmp[,2] - mean(tmp[,2])
  X1         <- rnorm(n, 0, 1)
  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*((cst[m] + Z1*beta[1] + Z2*beta[2] +
                        kronecker(mum, t(num), "+") + rnorm(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
  nu[[m]]      <- num
}

mu  <- unlist(mu)
nu  <- unlist(nu)

out   <- homophily.re(network =  Glist, formula = ~ dX, group.fe = TRUE,
                      re.way = 2, iteration = 1e3)

# plot simulations
plot(out$posterior$beta[,1], type = "l")
abline(h = cst[1], col = "red")
plot(out$posterior$beta[,2], type = "l")
abline(h = cst[2], col = "red")
plot(out$posterior$beta[,3], type = "l")
abline(h = cst[3], col = "red")
plot(out$posterior$beta[,4], type = "l")
abline(h = cst[4], col = "red")

plot(out$posterior$beta[,5], type = "l")
abline(h = beta[1], col = "red")
plot(out$posterior$beta[,6], type = "l")
abline(h = beta[2], col = "red")
```

```
plot(out$posterior$sigma2_mu, type = "l")
abline(h = smu2, col = "red")
plot(out$posterior$sigma2_nu, type = "l")
abline(h = snu2, col = "red")
plot(out$posterior$rho, type = "l")
abline(h = rho, col = "red")

i <- 10
plot(out$posterior$mu[,i], type = "l")
abline(h = mu[i], col = "red")
plot(out$posterior$nu[,i], type = "l")
abline(h = nu[i], col = "red")
```

`meffects` *Marginal Effects for Count Data Models and Tobit Models with Social Interactions*

Description

`meffects` computes marginal effects for count data and Tobit models with social interactions. It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
meffects(model, ...)

## S3 method for class 'cdnet'
meffects(
  model,
  Glist,
  cont.var,
  bin.var,
  type.var,
  Glist.contextual,
  data,
  tol = 1e-10,
  maxit = 500,
  boot = 1000,
  progress = TRUE,
  ncores = 1,
  ...
)

## S3 method for class 'summary.cdnet'
meffects(
  model,
  Glist,
```

```
  cont.var,
  bin.var,
  type.var,
  Glist.contextual,
  data,
  tol = 1e-10,
  maxit = 500,
  boot = 1000,
  progress = TRUE,
  ncores = 1,
  ...
)

## S3 method for class 'sart'
meffects(
  model,
  Glist,
  cont.var,
  bin.var,
  type.var,
  Glist.contextual,
  data,
  tol = 1e-10,
  maxit = 500,
  boot = 1000,
  progress = TRUE,
  ncores = 1,
  ...
)

## S3 method for class 'summary.sart'
meffects(
  model,
  Glist,
  cont.var,
  bin.var,
  type.var,
  Glist.contextual,
  data,
  tol = 1e-10,
  maxit = 500,
  boot = 1000,
  progress = TRUE,
  ncores = 1,
  ...
)
```

Arguments

model	an object of class cdnet (summary.cdnet) or sart (summary.sart), output of the function cdnet or sart, respectively.
...	Additional arguments passed to methods.
Glist	The network matrix used to obtain model. Typically, this is the Glist argument supplied to the function cdnet or sart.
cont.var	A character vector of continuous variable names for which the marginal effects should be computed.
bin.var	A character vector of binary variable names for which the marginal effects should be computed.
type.var	A list indicating "own" and contextual variables that appear in the cont.var and bin.var arguments. The list contains pairs of variable names, with the first element being the "own" variable and the second being the contextual variable. When a variable has no associated contextual variable, only the variable name is included. For example, type.var = list(c("x1", "gx1"), c("x2", "gx2"), "x3") means that gx1 is the contextual variable for x1, gx2 is the contextual variable for x2, and x3 has no contextual variable. This information is used to compute the indirect and total marginal effects for x1, x2, and x3.
Glist.contextual	The network matrix used to compute contextual variables, if any are specified in the type.var argument. For networks consisting of multiple subnets, Glist can be a list of subnets, where the m-th element is an ns*ns adjacency matrix, with ns denoting the number of nodes in the m-th subnet.
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(model), typically the environment from which meffects is called.
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 maximum number of iterations in the fixed-point iteration method.
boot	The number of bootstrap simulations to compute standard errors and confidence intervals.
progress	A logical value indicating whether the progress of the bootstrap simulations should be printed to the console.
ncores	Number of CPU cores (threads) used to run the bootstrap process in parallel.

Value

A list containing:

info General information about the model.

estimate The Maximum Likelihood (ML) estimates of the parameters.

Ey $E(y)$, the expected values of the endogenous variable.

`GEy` The average of $E(y)$ among peers.

`cov` A list containing covariance matrices (if `cov = TRUE`).

`details` Additional outputs returned by the optimizer.

`meffects` A list containing the marginal effects.

Examples

```
#' set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 200))
n      <- sum(nvec)

# Adjacency matrix
A      <- list()
for (m in 1:M) {
  nm           <- nvec[m]
  Am           <- matrix(0, nm, nm)
  max_d        <- 30 #maximum number of friends
  for (i in 1:nm) {
    tmp        <- sample((1:nm)[-i], sample(0:max_d, 1))
    Am[i, tmp] <- 1
  }
  A[[m]]       <- Am
}
Anorm  <- norm.network(A) #Row-normalization

# X
X      <- cbind(rnorm(n, 1, 3), rexp(n, 0.4))

# Two group:
group  <- 1*(X[,1] > 0.95)

# Networks
# length(group) = 2 and unique(sort(group)) = c(0, 1)
# The networks must be defined as to capture:
# peer effects of `0` on `0`, peer effects of `1` on `0`
# peer effects of `0` on `1`, and peer effects of `1` on `1`
G        <- list()
cums     <- c(0, cumsum(nvec))
for (m in 1:M) {
  tp     <- group[(cums[m] + 1):(cums[m + 1])]
  Am     <- A[[m]]
  G[[m]] <- norm.network(list(Am * ((1 - tp) %*% t(1 - tp)),
                              Am * ((1 - tp) %*% t(tp)),
                              Am * (tp %*% t(1 - tp)),
                              Am * (tp %*% t(tp))))
}

# Parameters
lambda <- c(0.2, 0.3, -0.15, 0.25)
Gamma  <- c(4.5, 2.2, -0.9, 1.5, -1.2)
delta  <- rep(c(2.6, 1.47, 0.85, 0.7, 0.5), 2)
```

```
# Data
data   <- data.frame(X, peer.avg(Anorm, cbind(x1 = X[,1], x2 =  X[,2])))
colnames(data) = c("x1", "x2", "gx1", "gx2")

ytmp   <- simcdnet(formula = ~ x1 + x2 + gx1 + gx2, Glist = G, Rbar = rep(5, 2),
                   lambda = lambda, Gamma = Gamma, delta = delta, group = group,
                   data = data)
y      <- ytmp$y
hist(y, breaks = max(y) + 1)
table(y)

# Estimation
est   <- cdnet(formula = y ~ x1 + x2 + gx1 + gx2, Glist = G, Rbar = rep(5, 2), group = group,
               optimizer = "fastlbfgs", data = data,
               opt.ctr = list(maxit = 5e3, eps_f = 1e-11, eps_g = 1e-11))

meffects(est, Glist = G, data = data, cont.var = c("x1", "x2", "gx1", "gx2"),
         type.var = list(c("x1", "gx1"), c("x2", "gx2")), Glist.contextual = Anorm,
         boot = 100, ncores = 2)
```

`norm.network` *Creating Objects for Network Models*

Description

The `vec.to.mat` function creates a list of square matrices from a given vector. Elements of the generated matrices are taken from the vector and placed column-wise or row-wise, progressing from the first matrix in the list to the last. The diagonals of the generated matrices are set to zeros.
The `mat.to.vec` function creates a vector from a given list of square matrices. Elements of the generated vector are taken column-wise or row-wise, starting from the first matrix in the list to the last, excluding diagonal entries.
The `norm.network` function row-normalizes matrices in a given list.

Usage

```
norm.network(W)

vec.to.mat(u, N, normalise = FALSE, byrow = FALSE)

mat.to.vec(W, ceiled = FALSE, byrow = FALSE)
```

Arguments

`W`	A matrix or list of matrices to convert.
`u`	A numeric vector to convert.
`N`	A vector of sub-network sizes such that `length(u) == sum(N * (N - 1))`.

normalise	A boolean indicating whether the returned matrices should be row-normalized (TRUE) or not (FALSE).
byrow	A boolean indicating whether entries in the matrices should be taken by row (TRUE) or by column (FALSE).
ceiled	A boolean indicating whether the given matrices should be ceiled before conversion (TRUE) or not (FALSE).

Value

A vector of size sum(N * (N - 1)) or a list of length(N) square matrices, with matrix sizes determined by N[1], N[2], ...

See Also

simnetwork, peer.avg.

Examples

```
# Generate a list of adjacency matrices
## Sub-network sizes
N <- c(250, 370, 120)
## Rate of friendship
p <- c(0.2, 0.15, 0.18)
## Network data
u <- unlist(lapply(1:3, function(x) rbinom(N[x] * (N[x] - 1), 1, p[x])))
W <- vec.to.mat(u, N)

# Convert W into a list of row-normalized matrices
G <- norm.network(W)

# Recover u
v <- mat.to.vec(G, ceiled = TRUE)
all.equal(u, v)
```

---

peer.avg *Computing Peer Averages*

Description

The peer.avg function computes peer average values using network data (provided as a list of adjacency matrices) and observable characteristics.

Usage

```
peer.avg(Glist, V, export.as.list = FALSE)
```

Arguments

Glist	An adjacency matrix or a list of sub-adjacency matrices representing the network structure.
V	A vector or matrix of observable characteristics.
export.as.list	(optional) A boolean indicating whether the output should be a list of matrices (TRUE) or a single matrix (FALSE).

Value

The matrix product diag(Glist[[1]], Glist[[2]], ...) %*% V, where diag() represents the block diagonal operator.

See Also

simnetwork, vec.to.mat

Examples

```
# Generate a list of adjacency matrices
## Sub-network sizes
N <- c(250, 370, 120)
## Rate of friendship
p <- c(0.2, 0.15, 0.18)
## Network data
u <- unlist(lapply(1:3, function(x) rbinom(N[x] * (N[x] - 1), 1, p[x])))
G <- vec.to.mat(u, N, normalise = TRUE)

# Generate a vector y
y <- rnorm(sum(N))

# Compute G %*% y
Gy <- peer.avg(Glist = G, V = y)
```

---

print.simcdEy *Printing the Average Expected Outcomes for Count Data Models with Social Interactions*

Description

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

Usage

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

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

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

Arguments

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

Value

A list of the same objects in object.

---

remove.ids *Removing Identifiers with NA from Adjacency Matrices Optimally*

Description

The remove.ids function removes identifiers with missing values (NA) from adjacency matrices in an optimal way. Multiple combinations of rows and columns can be deleted to eliminate NAs, but this function ensures that the smallest number of rows and columns are removed to retain as much data as possible.

Usage

```
remove.ids(network, ncores = 1L)
```

Arguments

network	A list of adjacency matrices to process.
ncores	The number of cores to use for parallel computation.

Value

A list containing:

network A list of adjacency matrices without missing values.

id A list of vectors indicating the indices of retained rows and columns for each matrix.

Examples

```
# Example 1: Small adjacency matrix
A <- matrix(1:25, 5)
A[1, 1] <- NA
A[4, 2] <- NA
remove.ids(A)

# Example 2: Larger adjacency matrix with multiple NAs
B <- matrix(1:100, 10)
B[1, 1] <- NA
B[4, 2] <- NA
B[2, 4] <- NA
B[, 8] <- NA
remove.ids(B)
```

---

sar *Estimating Linear-in-mean Models with Social Interactions*

Description

`sar` computes quasi-maximum likelihood estimators for linear-in-mean models with social interactions (see Lee, 2004 and Lee et al., 2010).

Usage

```
sar(
  formula,
  Glist,
  lambda0 = NULL,
  fixed.effects = FALSE,
  optimizer = "optim",
  opt.ctr = list(),
  print = TRUE,
  cov = TRUE,
  cinfo = TRUE,
  data
)
```

Arguments

`formula`	a class object formula: a symbolic description of the model. `formula` must be as, for example, `y ~ x1 + x2 + gx1 + gx2` where `y` is the endogenous vector and `x1`, `x2`, `gx1` and `gx2` are control variables, which can include contextual variables, i.e. averages among the peers. Peer averages can be computed using the function `peer.avg`.
`Glist`	The network matrix. For networks consisting of multiple subnets, `Glist` can be a list of subnets with the `m`-th element being an `ns*ns` adjacency matrix, where `ns` is the number of nodes in the `m`-th subnet.

`lambda0`	an optional starting value of λ.
`fixed.effects`	a Boolean indicating whether group heterogeneity must be included as fixed effects.
`optimizer`	is either `nlm` (referring to the function nlm) or `optim` (referring to the function optim). Arguments for these functions such as, `control` and `method` can be set via the argument `opt.ctr`.
`opt.ctr`	list of arguments of nlm or optim (the one set in `optimizer`) such as `control`, `method`, etc.
`print`	a Boolean indicating if the estimate should be printed at each step.
`cov`	a Boolean indicating if the covariance should be computed.
`cinfo`	a Boolean indicating whether information is complete (`cinfo = TRUE`) or incomplete (`cinfo = FALSE`). In the case of incomplete information, the model is defined under rational expectations.
`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 `sar` is called.

Details

In the complete information model, the outcome y_i for individual i is defined as:

$$y_i = \lambda \bar{y}_i + \mathbf{z}_i'\Gamma + \epsilon_i,$$

where $\bar{y}_i$ represents the average outcome y among individual i's peers, $\mathbf{z}_i$ is a vector of control variables, and $\epsilon_i \sim N(0, \sigma^2)$ is the error term. In the case of incomplete information models with rational expectations, the outcome y_i is defined as:

$$y_i = \lambda E(\bar{y}_i) + \mathbf{z}_i'\Gamma + \epsilon_i,$$

where $E(\bar{y}_i)$ is the expected average outcome of i's peers, as perceived by individual i.

Value

A list consisting of:

`info`	list of general information on the model.
`estimate`	Maximum Likelihood (ML) estimator.
`cov`	covariance matrix of the estimate.
`details`	outputs as returned by the optimizer.

References

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.

Lee, L. F., Liu, X., & Lin, X. (2010). Specification and estimation of social interaction models with network structures. The Econometrics Journal, 13(2), 145-176, doi:10.1111/j.1368423X.2010.00310.x

See Also

sart, cdnet, simsar.

Examples

```
# Groups' size
set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
n      <- sum(nvec)

# Parameters
lambda <- 0.4
Gamma  <- c(2, -1.9, 0.8, 1.5, -1.2)
sigma  <- 1.5
theta  <- c(lambda, Gamma, sigma)

# X
X      <- cbind(rnorm(n, 1, 1), rexp(n, 0.4))

# Network
G      <- 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
  G[[m]]       <- Gm
}

# data
data   <- data.frame(X, peer.avg(G, cbind(x1 = X[,1], x2 =  X[,2])))
colnames(data) <- c("x1", "x2", "gx1", "gx2")

ytmp    <- simsar(formula = ~ x1 + x2 + gx1 + gx2, Glist = G,
                  theta = theta, data = data)
data$y  <- ytmp$y

out     <- sar(formula = y ~ x1 + x2 + gx1 + gx2, Glist = G,
               optimizer = "optim", data = data)
summary(out)
```

`sart` *Estimating Tobit Models with Social Interactions*

Description

`sart` estimates Tobit models with social interactions based on the framework of Xu and Lee (2015). The method allows for modeling both complete and incomplete information scenarios in networks, incorporating rational expectations in the latter case.

Usage

```
sart(
  formula,
  Glist,
  starting = NULL,
  Ey0 = NULL,
  optimizer = "fastlbfgs",
  npl.ctr = list(),
  opt.ctr = list(),
  cov = TRUE,
  cinfo = TRUE,
  data
)
```

Arguments

`formula`	An object of class formula: a symbolic description of the model. The formula must follow the structure, e.g., `y ~ x1 + x2 + gx1 + gx2`, where `y` is the endogenous variable, and `x1`, `x2`, `gx1`, and `gx2` are control variables. Control variables may include contextual variables, such as peer averages, which can be computed using `peer.avg`.
`Glist`	The network matrix. For networks consisting of multiple subnets, `Glist` can be a list, where the `m`-th element is an `ns*ns` adjacency matrix representing the `m`-th subnet, with `ns` being the number of nodes in that subnet.
`starting`	(Optional) A vector of starting values for $\theta = (\lambda, \Gamma, \sigma)$, where: • λ is the peer effect coefficient, • Γ is the vector of control variable coefficients, • σ is the standard deviation of the error term.
`Ey0`	(Optional) A starting value for $E(y)$.
`optimizer`	The optimization method to be used. Choices are: • `"fastlbfgs"`: L-BFGS optimization method from the **RcppNumerical** package, • `"nlm"`: Refers to the nlm function, • `"optim"`: Refers to the optim function.

	Additional arguments for these functions, such as `control` and `method`, can be specified through the `opt.ctr` argument.
`npl.ctr`	A list of controls for the NPL (Nested Pseudo-Likelihood) method (refer to the details in `cdnet`).
`opt.ctr`	A list of arguments to be passed to the chosen solver (`fastlbfgs`, nlm, or optim), such as `maxit`, `eps_f`, `eps_g`, `control`, `method`, etc.
`cov`	A Boolean indicating whether to compute the covariance matrix (`TRUE` or `FALSE`).
`cinfo`	A Boolean indicating whether the information structure is complete (`TRUE`) or incomplete (`FALSE`). Under incomplete information, the model is defined with rational expectations.
`data`	An optional data frame, list, or environment (or object coercible by as.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 `sart` is called.

Details

For a complete information model, the outcome y_i is defined as:

$$\begin{cases} y_i^* = \lambda \bar{y}_i + \mathbf{z}_i'\Gamma + \epsilon_i, \\ y_i = \max(0, y_i^*), \end{cases}$$

where $\bar{y}_i$ is the average of y among peers, $\mathbf{z}_i$ is a vector of control variables, and $\epsilon_i \sim N(0, \sigma^2)$.

In the case of incomplete information models with rational expectations, y_i is defined as:

$$\begin{cases} y_i^* = \lambda E(\bar{y}_i) + \mathbf{z}_i'\Gamma + \epsilon_i, \\ y_i = \max(0, y_i^*). \end{cases}$$

Value

A list containing:

`info` General information about the model.

`estimate` The Maximum Likelihood (ML) estimates of the parameters.

`Ey` $E(y)$, the expected values of the endogenous variable.

`GEy` The average of $E(y)$ among peers.

`cov` A list including covariance matrices (if `cov = TRUE`).

`details` Additional outputs returned by the optimizer.

References

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

sar, cdnet, simsart.

Examples

```
# Group sizes
set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 200))
n      <- sum(nvec)

# Parameters
lambda <- 0.4
Gamma  <- c(2, -1.9, 0.8, 1.5, -1.2)
sigma  <- 1.5
theta  <- c(lambda, Gamma, sigma)

# Covariates (X)
X      <- cbind(rnorm(n, 1, 1), rexp(n, 0.4))

# Network creation
G      <- 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
  G[[m]]       <- Gm
}

# Data creation
data   <- data.frame(X, peer.avg(G, cbind(x1 = X[, 1], x2 = X[, 2])))
colnames(data) <- c("x1", "x2", "gx1", "gx2")

## Complete information game
ytmp    <- simsart(formula = ~ x1 + x2 + gx1 + gx2, Glist = G, theta = theta,
                   data = data, cinfo = TRUE)
data$yc <- ytmp$y

## Incomplete information game
ytmp    <- simsart(formula = ~ x1 + x2 + gx1 + gx2, Glist = G, theta = theta,
                   data = data, cinfo = FALSE)
data$yi <- ytmp$y

# Complete information estimation for yc
outc1   <- sart(formula = yc ~ x1 + x2 + gx1 + gx2, optimizer = "nlm",
```

```
                Glist = G, data = data, cinfo = TRUE)
summary(outc1)

# Complete information estimation for yi
outc1   <- sart(formula = yi ~ x1 + x2 + gx1 + gx2, optimizer = "nlm",
                Glist = G, data = data, cinfo = TRUE)
summary(outc1)

# Incomplete information estimation for yc
outi1   <- sart(formula = yc ~ x1 + x2 + gx1 + gx2, optimizer = "nlm",
                Glist = G, data = data, cinfo = FALSE)
summary(outi1)

# Incomplete information estimation for yi
outi1   <- sart(formula = yi ~ x1 + x2 + gx1 + gx2, optimizer = "nlm",
                Glist = G, data = data, cinfo = FALSE)
summary(outi1)
```

`simcdEy` *Counterfactual Analyses with Count Data Models and Social Interactions*

Description

`simcdpar` computes the average expected outcomes for count data models with social interactions and standard errors using the Delta method. This function can be used to examine the effects of changes in the network or in the control variables.

Usage

```
simcdEy(object, Glist, data, group, tol = 1e-10, maxit = 500, S = 1000)
```

Arguments

`object`	an object of class `summary.cdnet`, output of the function `summary.cdnet` or class `cdnet`, output of the function `cdnet`.
`Glist`	adjacency matrix. For networks consisting of multiple subnets, `Glist` can be a list of subnets with the `m`-th element being an `ns*ns` adjacency matrix, where `ns` is the number of nodes in the `m`-th subnet. For heterogeneous peer effects (e.g., boy-boy, boy-girl friendship effects), the `m`-th element can be a list of many `ns*ns` adjacency matrices corresponding to the different network specifications (see Houndetoungan, 2024). For heterogeneous peer effects in the case of a single large network, `Glist` must be a one-item list. This item must be a list of many specifications of large networks.
`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 `summary.cdnet` is called.

group	the vector indicating the individual groups (see function cdnet). If missing, the former group saved in object will be used.
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 $E(y)$ is less than tol.
maxit	the maximal number of iterations in the Fixed Point Iteration Method.
S	number of simulations to be used to compute integral in the covariance by important sampling.

Value

A list consisting of:

Ey	$E(y)$, the expectation of y.
GEy	the average of $E(y)$ friends.
aEy	the sampling mean of $E(y)$.
se.aEy	the standard error of the sampling mean of $E(y)$.

See Also

simcdnet

---

simcdnet *Simulating Count Data Models with Social Interactions Under Rational Expectations*

Description

simcdnet simulates the count data model with social interactions under rational expectations developed by Houndetoungan (2024).

Usage

```
simcdnet(
  formula,
  group,
  Glist,
  parms,
  lambda,
  Gamma,
  delta,
  Rmax,
  Rbar,
  cont.var,
  bin.var,
  tol = 1e-10,
```

```
  maxit = 500,
  data
)
```

Arguments

formula	A class object of class formula: a symbolic description of the model. formula should be specified, for example, as y ~ x1 + x2 + gx1 + gx2, where y is the endogenous vector and x1, x2, gx1, and gx2 are control variables. These control variables can include contextual variables, such as averages among the peers. Peer averages can be computed using the function peer.avg.
group	A vector indicating the individual groups. By default, this assumes a common group. If there are 2 groups (i.e., length(unique(group)) = 2, such as A and B), four types of peer effects are defined: peer effects of A on A, A on B, B on A, and B on B.
Glist	An adjacency matrix or list of adjacency matrices. For networks consisting of multiple subnets (e.g., schools), Glist can be a list of subnet matrices, where the m-th element is an $n_m \times n_m$ adjacency matrix, with n_m representing the number of nodes in the m-th subnet. For heterogeneous peer effects (length(unique(group)) = h > 1), the m-th element should be a list of h^2 $n_m \times n_m$ adjacency matrices corresponding to different network specifications (see Houndetoungan, 2024). For heterogeneous peer effects in a single large network, Glist should be a one-item list, where the item is a list of h^2 network specifications. The order of these networks is important and must match sort(unique(group)) (see examples).
parms	A vector defining the true values of $\theta = (\lambda', \Gamma', \delta')'$ (see model specification in the details section). Each parameter λ, Γ, or δ can also be provided separately to the arguments lambda, Gamma, or delta.
lambda	The true value of the vector λ.
Gamma	The true value of the vector Γ.
delta	The true value of the vector δ.
Rmax	An integer indicating the theoretical upper bound of y (see model specification in detail).
Rbar	An L-vector, where L is the number of groups. For large Rmax, the cost function is assumed to be semi-parametric (i.e., nonparametric from 0 to $\bar{R}$ and quadratic beyond $\bar{R}$). The l-th element of Rbar indicates $\bar{R}$ for the l-th value of sort(unique(group)) (see model specification in detail).
cont.var	A character vector of continuous variable names for which the marginal effects should be computed.
bin.var	A character vector of binary variable names for which the marginal effects should be computed.
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 $E(y)$ is less than tol.
maxit	The maximum 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 `simcdnet` is called.

Details

The count variable y_i takes the value r with probability.

$$P_{ir} = F(\sum_{s=1}^{S} \lambda_s \bar{y}_i^{e,s} + \mathbf{z}_i'\Gamma - a_{h(i),r}) - F(\sum_{s=1}^{S} \lambda_s \bar{y}_i^{e,s} + \mathbf{z}_i'\Gamma - a_{h(i),r+1}).$$

In this equation, $\mathbf{z}_i$ is a vector of control variables; F is the distribution function of the standard normal distribution; $\bar{y}_i^{e,s}$ is the average of $E(y)$ among peers using the `s`-th network definition; $a_{h(i),r}$ is the `r`-th cut-point in the cost group $h(i)$.

The following identification conditions have been introduced: $\sum_{s=1}^{S} \lambda_s > 0$, $a_{h(i),0} = -\infty$, $a_{h(i),1} = 0$, and $a_{h(i),r} = \infty$ for any $r \geq R_{\max} + 1$. The last condition implies that $P_{ir} = 0$ for any $r \geq R_{\max} + 1$. For any $r \geq 1$, the distance between two cut-points is $a_{h(i),r+1} - a_{h(i),r} = \delta_{h(i),r} + \sum_{s=1}^{S} \lambda_s$. As the number of cut-points can be large, a quadratic cost function is considered for $r \geq \bar{R}_{h(i)}$, where $\bar{R} = (\bar{R}_1, ..., \bar{R}_L)$. With the semi-parametric cost function, $a_{h(i),r+1} - a_{h(i),r} = \bar{\delta}_{h(i)} + \sum_{s=1}^{S} \lambda_s$.

The model parameters are: $\lambda = (\lambda_1, ..., \lambda_S)'$, Γ, and $\delta = (\delta_1', ..., \delta_L')'$, where $\delta_l = (\delta_{l,2}, ..., \delta_{l,\bar{R}_l}, \bar{\delta}_l)'$ for $l = 1, ..., L$. The number of single parameters in δ_l depends on $R_{\max}$ and $\bar{R}_l$. The components $\delta_{l,2}, ..., \delta_{l,\bar{R}_l}$ or/and $\bar{\delta}_l$ must be removed in certain cases.
If $R_{\max} = \bar{R}_l \geq 2$, then $\delta_l = (\delta_{l,2}, ..., \delta_{l,\bar{R}_l})'$.
If $R_{\max} = \bar{R}_l = 1$ (binary models), then δ_l must be empty.
If $R_{\max} > \bar{R}_l = 1$, then $\delta_l = \bar{\delta}_l$.

Value

A list consisting of:

`yst`	y^*, the latent variable.
`y`	the observed count variable.
`Ey`	$E(y)$, the expectation of y.
`GEy`	the average of $E(y)$ among peers.
`meff`	a list including average and individual marginal effects.
`Rmax`	infinite sums in the marginal effects are approximated by sums up to Rmax.
`iteration`	number of iterations performed by sub-network in the Fixed Point Iteration Method.

References

Houndetoungan, A. (2024). Count Data Models with Heterogeneous Peer Effects. Available at SSRN 3721250, doi:10.2139/ssrn.3721250.

See Also

cdnet, simsart, simsar.

Examples

```
set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 200)) # Random group sizes
n      <- sum(nvec) # Total number of individuals

# Adjacency matrix for each group
A      <- list()
for (m in 1:M) {
  nm           <- nvec[m] # Size of group m
  Am           <- matrix(0, nm, nm) # Empty adjacency matrix
  max_d        <- 30 # Maximum number of friends
  for (i in 1:nm) {
    tmp        <- sample((1:nm)[-i], sample(0:max_d, 1)) # Sample friends
    Am[i, tmp] <- 1 # Set friendship links
  }
  A[[m]]       <- Am # Add to the list
}
Anorm  <- norm.network(A) # Row-normalization of the adjacency matrices

# Covariates (X)
X      <- cbind(rnorm(n, 1, 3), rexp(n, 0.4)) # Random covariates

# Two groups based on first covariate
group  <- 1 * (X[,1] > 0.95) # Assign to groups based on x1

# Networks: Define peer effects based on group membership
# The networks should capture:
# - Peer effects of `0` on `0`
# - Peer effects of `1` on `0`
# - Peer effects of `0` on `1`
# - Peer effects of `1` on `1`
G        <- list()
cums     <- c(0, cumsum(nvec)) # Cumulative indices for groups
for (m in 1:M) {
  tp     <- group[(cums[m] + 1):(cums[m + 1])] # Group membership for group m
  Am     <- A[[m]] # Adjacency matrix for group m
  # Define networks based on peer effects
  G[[m]] <- norm.network(list(Am * ((1 - tp) %*% t(1 - tp)),
                              Am * ((1 - tp) %*% t(tp)),
                              Am * (tp %*% t(1 - tp)),
                              Am * (tp %*% t(tp))))
}

# Parameters for the model
lambda <- c(0.2, 0.3, -0.15, 0.25)
Gamma  <- c(4.5, 2.2, -0.9, 1.5, -1.2)
delta  <- rep(c(2.6, 1.47, 0.85, 0.7, 0.5), 2) # Repeated values for delta
```

```
# Prepare data for the model
data   <- data.frame(X, peer.avg(Anorm, cbind(x1 = X[,1], x2 = X[,2])))
colnames(data) = c("x1", "x2", "gx1", "gx2") # Set column names

# Simulate outcomes using the `simcdnet` function
ytmp   <- simcdnet(formula = ~ x1 + x2 + gx1 + gx2, Glist = G, Rbar = rep(5, 2),
                   lambda = lambda, Gamma = Gamma, delta = delta, group = group,
                   data = data)
y      <- ytmp$y

# Plot histogram of the simulated outcomes
hist(y, breaks = max(y) + 1)

# Display frequency table of the simulated outcomes
table(y)
```

`simnetwork` *Simulating Network Data*

Description

`simnetwork` generates adjacency matrices based on specified probabilities.

Usage

```
simnetwork(dnetwork, normalise = FALSE)
```

Arguments

`dnetwork`	A list of sub-network matrices, where the (i, j)-th position of the m-th matrix represents the probability that individual `i` is connected to individual `j` in the m-th network.
`normalise`	A boolean indicating whether the returned matrices should be row-normalized (`TRUE`) or not (`FALSE`).

Value

A list of (row-normalized) adjacency matrices.

Examples

```
# Generate a list of adjacency matrices
## Sub-network sizes
N         <- c(250, 370, 120)
## Probability distributions
dnetwork  <- lapply(N, function(x) matrix(runif(x^2), x))
## Generate networks
G         <- simnetwork(dnetwork)
```

simsar *Simulating Data from Linear-in-Mean Models with Social Interactions*

Description

simsar simulates continuous variables under linear-in-mean models with social interactions, following the specifications described in Lee (2004) and Lee et al. (2010). The model incorporates peer interactions, where the value of an individual's outcome depends not only on their own characteristics but also on the average characteristics of their peers in the network.

Usage

```
simsar(formula, Glist, theta, cinfo = TRUE, data)
```

Arguments

formula	A symbolic description of the model, passed as a class object of type formula. The formula must specify the endogenous variable and control variables, for example: y ~ x1 + x2 + gx1 + gx2, where y is the endogenous vector, and x1, x2, gx1, and gx2 are the control variables, which may include contextual variables (peer averages). Peer averages can be computed using the function peer.avg.
Glist	A list of network adjacency matrices representing multiple subnets. The m-th element in the list should be an ns * ns matrix, where ns is the number of nodes in the m-th subnet.
theta	A numeric vector defining the true values of the model parameters $\theta = (\lambda, \Gamma, \sigma)$. These parameters are used to define the model specification in the details section.
cinfo	A Boolean flag indicating whether the information is complete (cinfo = TRUE) or incomplete (cinfo = FALSE). If information is incomplete, the model operates under rational expectations.
data	An optional data frame, list, or environment (or an object coercible by as.data.frame to a data frame) containing the variables in the model. If not provided, the variables are taken from the environment of the function call.

Details

In the complete information model, the outcome y_i for individual i is defined as:

$$y_i = \lambda \bar{y}_i + \mathbf{z}_i' \Gamma + \epsilon_i,$$

where $\bar{y}_i$ represents the average outcome y among individual i's peers, $\mathbf{z}_i$ is a vector of control variables, and $\epsilon_i \sim N(0, \sigma^2)$ is the error term. In the case of incomplete information models with rational expectations, the outcome y_i is defined as:

$$y_i = \lambda E(\bar{y}_i) + \mathbf{z}_i' \Gamma + \epsilon_i,$$

where $E(\bar{y}_i)$ is the expected average outcome of i's peers, as perceived by individual i.

Value

A list containing the following elements:

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

References

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.

Lee, L. F., Liu, X., & Lin, X. (2010). Specification and estimation of social interaction models with network structures. The Econometrics Journal, 13(2), 145-176, doi:10.1111/j.1368423X.2010.00310.x

See Also

sar, simsart, simcdnet.

Examples

```
# Groups' size
set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 1000))
n      <- sum(nvec)

# Parameters
lambda <- 0.4
Gamma  <- c(2, -1.9, 0.8, 1.5, -1.2)
sigma  <- 1.5
theta  <- c(lambda, Gamma, sigma)

# X
X      <- cbind(rnorm(n, 1, 1), rexp(n, 0.4))

# Network
G      <- 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
  G[[m]]       <- Gm
}

# data
```

```
data   <- data.frame(X, peer.avg(G, cbind(x1 = X[,1], x2 =  X[,2])))
colnames(data) <- c("x1", "x2", "gx1", "gx2")

ytmp    <- simsar(formula = ~ x1 + x2 + gx1 + gx2, Glist = G,
                  theta = theta, data = data)
y       <- ytmp$y
```

---

simsart *Simulating Data from Tobit Models with Social Interactions*

Description

simsart simulates censored data with social interactions (see Xu and Lee, 2015).

Usage

```
simsart(
  formula,
  Glist,
  theta,
  cont.var,
  bin.var,
  tol = 1e-15,
  maxit = 500,
  cinfo = TRUE,
  data
)
```

Arguments

formula	a class object formula: a symbolic description of the model. formula must be, for example, y ~ x1 + x2 + gx1 + gx2, where y is the endogenous vector, and x1, x2, gx1, and gx2 are control variables. These can include contextual variables, i.e., averages among the peers. Peer averages can be computed using the function peer.avg.
Glist	The network matrix. For networks consisting of multiple subnets, Glist can be a list of subnets with the m-th element being an ns*ns adjacency matrix, where ns is the number of nodes in the m-th subnet.
theta	a vector defining the true value of $\theta = (\lambda, \Gamma, \sigma)$ (see the model specification in the details).
cont.var	A character vector of continuous variable names for which the marginal effects should be computed.
bin.var	A character vector of binary variable names for which the marginal effects should be computed.

`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 maximum number of iterations in the fixed-point iteration method.
`cinfo`	a Boolean indicating whether information is complete (`cinfo = TRUE`) or incomplete (`cinfo = FALSE`). In the case of incomplete information, the model is defined under rational expectations.
`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 `simsart` is called.

Details

For a complete information model, the outcome y_i is defined as:

$$\begin{cases} y_i^* = \lambda \bar{y}_i + \mathbf{z}_i'\Gamma + \epsilon_i, \\ y_i = \max(0, y_i^*), \end{cases}$$

where $\bar{y}_i$ is the average of y among peers, $\mathbf{z}_i$ is a vector of control variables, and $\epsilon_i \sim N(0, \sigma^2)$.

In the case of incomplete information models with rational expectations, y_i is defined as:

$$\begin{cases} y_i^* = \lambda E(\bar{y}_i) + \mathbf{z}_i'\Gamma + \epsilon_i, \\ y_i = \max(0, y_i^*). \end{cases}$$

Value

A list consisting of:

yst y^*, the latent variable.

y The observed censored variable.

Ey $E(y)$, the expected value of y.

Gy The average of y among peers.

GEy The average of $E(y)$ among peers.

meff A list including average and individual marginal effects.

iteration The number of iterations performed per sub-network in the fixed-point iteration method.

References

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

`sart`, `simsar`, `simcdnet`.

Examples

```
# Define group sizes
set.seed(123)
M      <- 5 # Number of sub-groups
nvec   <- round(runif(M, 100, 200)) # Number of nodes per sub-group
n      <- sum(nvec) # Total number of nodes

# Define parameters
lambda <- 0.4
Gamma  <- c(2, -1.9, 0.8, 1.5, -1.2)
sigma  <- 1.5
theta  <- c(lambda, Gamma, sigma)

# Generate covariates (X)
X      <- cbind(rnorm(n, 1, 1), rexp(n, 0.4))

# Construct network adjacency matrices
G      <- list()
for (m in 1:M) {
  nm           <- nvec[m] # Nodes in sub-group m
  Gm           <- matrix(0, nm, nm) # Initialize adjacency matrix
  max_d        <- 30 # Maximum degree
  for (i in 1:nm) {
    tmp        <- sample((1:nm)[-i], sample(0:max_d, 1)) # Random connections
    Gm[i, tmp] <- 1
  }
  rs           <- rowSums(Gm) # Normalize rows
  rs[rs == 0]  <- 1
  Gm           <- Gm / rs
  G[[m]]       <- Gm
}

# Prepare data
data   <- data.frame(X, peer.avg(G, cbind(x1 = X[, 1], x2 = X[, 2])))
colnames(data) <- c("x1", "x2", "gx1", "gx2") # Add column names

# Complete information game simulation
ytmp    <- simsart(formula = ~ x1 + x2 + gx1 + gx2,
                   Glist = G, theta = theta,
                   data = data, cinfo = TRUE)
data$yc <- ytmp$y # Add simulated outcome to the dataset

# Incomplete information game simulation
ytmp    <- simsart(formula = ~ x1 + x2 + gx1 + gx2,
                   Glist = G, theta = theta,
                   data = data, cinfo = FALSE)
data$yi <- ytmp$y # Add simulated outcome to the dataset
```

summary.cdnet *Summary for the Estimation of Count Data Models with Social Interactions under Rational Expectations*

Description

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

Usage

```
## S3 method for class 'cdnet'
summary(object, Glist, data, S = 1000L, ...)

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

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

Arguments

object	an object of class cdnet, output of the function cdnet.
Glist	adjacency matrix. For networks consisting of multiple subnets, Glist can be a list of subnets with the m-th element being an ns*ns adjacency matrix, where ns is the number of nodes in the m-th subnet. For heterogeneous peer effects (e.g., boy-boy, boy-girl friendship effects), the m-th element can be a list of many ns*ns adjacency matrices corresponding to the different network specifications (see Houndetoungan, 2024). For heterogeneous peer effects in the case of a single large network, Glist must be a one-item list. This item must be a list of many specifications of large networks.
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 summary.cdnet is called.
S	number of simulations to be used to compute integral in the covariance by important sampling.
...	further arguments passed to or from other methods.
x	an object of class summary.cdnet, output of the function summary.cdnet or class cdnet, output of the function cdnet.

Value

A list of the same objects in object.

summary.sar — *Summary for the Estimation of Linear-in-mean Models with Social Interactions*

Description

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

Usage

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

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

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

Arguments

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

Value

A list of the same objects in `object`.

summary.sart — *Summary for the Estimation of Tobit Models with Social Interactions*

Description

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

Usage

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

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

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

Arguments

object	an object of class sart, output of the function sart.
Glist	adjacency matrix or list sub-adjacency matrix. This is not necessary if the covariance method was computed in cdnet.
data	dataframe containing the explanatory variables. This is not necessary if the covariance method was computed in cdnet.
...	further arguments passed to or from other methods.
x	an object of class summary.sart, output of the function summary.sart or class sart, output of the function sart.

Value

A list of the same objects in object.

Index