Package: CDatanet (via r-universe)

October 12, 2024

Type Package

Title Econometrics of Network Data

Version 2.2.0

Date 2024-06-01

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, 2024; <doi:10.2139/ssrn.3721250>). 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, ddpcr, Matrix, matrixcalc, foreach, doRNG, doParallel, parallel

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

RoxygenNote 7.3.1

Suggests ggplot2, MASS, knitr, rmarkdown

NeedsCompilation yes

Author Aristide Houndetoungan [cre, aut]

Maintainer Aristide Houndetoungan <ahoundetoungan@gmail.com>

Repository CRAN

Date/Publication 2024-05-14 15:40:02 UTC

Contents

`CDatanet-package` *The CDatanet package*

Description

The **CDatanet** package simulates and estimates peer effect models and network formation models. The class of peer effect models includes 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 pair-wise regressions with degree heterogeneity (Graham, 2017; Yan et al., 2019) and exponential random graph models (Mele, 2017). To make the computations faster **CDatanet** uses C++ through the **Rcpp** package (Eddelbuettel et al., 2011).

Author(s)

Maintainer: Aristide Houndetoungan <ahoundetoungan@gmail.com>

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. (2024). Count Data Models with Social Interactions under Rational Expectations. 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.

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. 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	adjacency matrix. For networks consisting of multiple subnets, Glist can be a list of subnets with the m-th element being an $n_s \times n_s$-adjacency matrix, where n_s is the number of nodes in the m-th subnet. For heterogeneous peer effects (length(unique(group)) = h > 1), the m-th element must be a list of h^2 $n_s \times n_s$-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 h^2 network specifications. The order in which the networks in are specified are important and must match sort(unique(group)) (see examples).
group	the vector indicating the individual groups. The default assumes a common group. For 2 groups; that is, 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.
Rmax	an integer indicating the theoretical upper bound of y. (see the model specification in details).
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	is either fastlbfgs (L-BFGS optimization method of the package **RcppNumerical**), 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.
npl.ctr	a list of controls for the NPL method (see details).
opt.ctr	a list of arguments to be passed in optim_lbfgs of the package **RcppNumerical**, nlm or optim (the solver set in optimizer), such as maxit, eps_f, eps_g, control, method, etc.
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 cdnet is called.

Details

Model:

The count variable y_i take 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-point 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 starts with a guess of θ and $E(y)$ and constructs iteratively a sequence of θ and $E(y)$. The solution converges when the ℓ_1-distance between two consecutive θ and $E(y)$ is less than a tolerance.
The argument `npl.ctr` must include

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.

S the number of simulations performed use to compute integral in the covariance by important sampling.

Value

A list consisting of:

`info`	a list of general information about the model.
`estimate`	the NPL estimator.
`Ey`	$E(y)$, the expectation of y.
`GEy`	the average of $E(y)$ friends.
`cov`	a list including (if `cov == TRUE`) `parms` the covariance matrix and another list `var.comp`, which includes `Sigma`, as Σ, and `Omega`, as Ω, matrices used for compute the covariance matrix.
`details`	step-by-step output as returned by the optimizer.

References

Houndetoungan, E. A. (2024). Count Data Models with Social Interactions under Rational Expectations. 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`	is the `matrix` or `data.frame` of the explanatory variables of the network formation model. This corresponds to the `X` matrix in `homophily.fe` or in `homophily.re`.
`nvec`	is a vector of the number of individuals in the networks.
`to`	indicates the direction of the conversion. For a matrix of explanatory variable `X` (`n*(n-1)` rows), one can 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 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,
  opt.ctr = list(maxit = 10000, eps_f = 1e-09, eps_g = 1e-09),
  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. 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`, an K-dimensional vector of the explanatory variables parameter, `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 value for `c(beta, mu, nu)`.
`opt.ctr`	(optional) is a list of `maxit`, `eps_f`, and `eps_g`, which are control parameters used by the solver `optim_lbfgs`, of the package **RcppNumerical**.
`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 (`fe.way = 2`) as

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

where F is the cumulative of the standard logistic distribution. Unobserved degree heterogeneity is captured by μ_i and ν_j. The latter 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 then necessary and is not implemented in this version. However the estimator 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`	list of model information, such as the type of fixed effects, whether the model is symmetric, number of observations, etc.
`estimate`	maximizer of the log-likelihood.
`loglike`	maximized log-likelihood.
`optim`	returned value of the optimization solver, which contains details of the optimization. The solver used is `optim_lbfgs` of the package **RcppNumerical**.
`init`	returned list of starting value.
`loglike(init)`	log-likelihood at the starting value.

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 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 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 fixed effect model. The expected value is 1 or 2 (see details).
init	(optional) list of starting values containing beta, an 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 (fe.way = 2) as

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

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 (fe.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")
```

`norm.network` *Creating objects for network models*

Description

`vec.to.mat` creates a list of square matrices from a given vector. The elements of the generated matrices are taken from the vector and placed column-wise (ie. the first column is filled up before

filling the second column) and from the first matrix of the list to the last matrix of the list. The diagonal of the generated matrices are zeros. `mat.to.vec` creates a vector from a given list of square matrices . The elements of the generated vector are taken from column-wise and from the first matrix of the list to the last matrix of the list, while dropping the diagonal entry. `norm.network` 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`	matrix or list of matrices to convert.
`u`	numeric vector to convert.
`N`	vector of sub-network sizes such that `length(u) == sum(N*(N - 1))`.
`normalise`	Boolean takes `TRUE` if the returned matrices should be row-normalized and `FALSE` otherwise.
`byrow`	Boolean takes `TRUE` is entries in the matrices should be taken by row and `FALSE` if they should be taken by column.
`ceiled`	Boolean takes `TRUE` if the given matrices should be ceiled before conversion and `FALSE` otherwise.

Value

a vector of size `sum(N*(N - 1))` or list of `length(N)` square matrices. The sizes of the matrices are `N[1], N[2], ...`

See Also

`simnetwork`, `peer.avg`.

Examples

```
# Generate a list of adjacency matrices
## sub-network size
N <- c(250, 370, 120)
## rate of friendship
p <- c(.2, .15, .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 G 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

peer.avg computes peer average value using network data (as a list) and observable characteristics.

Usage

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

Arguments

Glist	the adjacency matrix or list sub-adjacency matrix.
V	vector or matrix of observable characteristics.
export.as.list	(optional) boolean to indicate if the output should be a list of matrices or a single matrix.

Value

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

See Also

simnetwork

Examples

```
# Generate a list of adjacency matrices
## sub-network size
N  <- c(250, 370, 120)
## rate of friendship
p  <- c(.2, .15, .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 IDs with NA from Adjacency Matrices Optimally*

Description

`remove.ids` optimally removes identifiers with NA from adjacency matrices. Many combinations of rows and columns can be deleted removing many rows and column

Usage

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

Arguments

`network`	is a list of adjacency matrices
`ncores`	is the number of cores to be used to run the program in parallel

Value

List of adjacency matrices without missing values and a list of vectors of retained indeces

Examples

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

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

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

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

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:

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

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 (Xu and Lee, 2015).

Usage

```
sart(
  formula,
  Glist,
  starting = NULL,
  Ey0 = NULL,
  optimizer = "fastlbfgs",
  npl.ctr = list(),
  opt.ctr = list(),
  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.
starting	(optional) a starting value for $\theta = (\lambda, \Gamma, \sigma)$ (see the model specification in details).
Ey0	(optional) a starting value for $E(y)$.
optimizer	is either fastlbfgs (L-BFGS optimization method of the package **RcppNumerical**), 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.
npl.ctr	a list of controls for the NPL method (see details of the function cdnet).
opt.ctr	a list of arguments to be passed in optim_lbfgs of the package **RcppNumerical**, nlm or optim (the solver set in optimizer), such as maxit, eps_f, eps_g, control, method, etc.
cov	a Boolean indicating if the covariance must 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 `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 modelswith 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:

`info`	a list of general information on the model.
`estimate`	the Maximum Likelihood (ML) estimator.
`Ey`	$E(y)$, the expectation of y.
`GEy`	the average of $E(y)$ friends.
`cov`	a list including (if `cov == TRUE`) covariance matrices.
`details`	outputs as 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

```
# Groups' size
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)

# 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")

## 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 heterogenous 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 simulate 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,
  tol = 1e-10,
  maxit = 500,
  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.

`group` the vector indicating the individual groups. The default assumes a common group. For 2 groups; that is, `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`.

`Glist` adjacency matrix. For networks consisting of multiple subnets, `Glist` can be a list of subnets with the `m`-th element being an $n_s \times n_s$-adjacency matrix, where n_s is the number of nodes in the `m`-th subnet. For heterogeneous peer effects (`length(unique(group)) = h > 1`), the `m`-th element must be a list of h^2 $n_s \times n_s$-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 h^2 network specifications. The order in which the networks in are specified are important and must match `sort(unique(group))` (see examples).

`parms` a vector defining the true value of $\theta = (\lambda', \Gamma', \delta')'$ (see the model specification in details). Each parameter λ, Γ, or δ can also be given 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 the model specification in details).

`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 the model specification in 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 $E(y)$ 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 `simcdnet` is called.

Details

The count variable y_i take 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-point 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 $E(y)$ friends.
`meff`	a list includinh 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, E. A. (2024). Count Data Models with Social Interactions under Rational Expectations. 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))
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)
```

`simnetwork` *Simulating network data*

Description

`simnetwork` simulates adjacency matrices.

Usage

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

Arguments

dnetwork	is a list of sub-network matrices, where the (i, j)-th position of the m-th matrix is the probability that i be connected to j, with i and j individuals from the m-th network.
normalise	boolean takes TRUE if the returned matrices should be row-normalized and FALSE otherwise.

Value

list of (row-normalized) adjacency matrices.

Examples

```
# Generate a list of adjacency matrices
## sub-network size
N         <- c(250, 370, 120)
## distribution
dnetwork  <- lapply(N, function(x) matrix(runif(x^2), x))
## network
G         <- simnetwork(dnetwork)
```

---

simsar *Simulating data from linear-in-mean models with social interactions*

Description

simsar simulates continuous variables with social interactions (see Lee, 2004 and Lee et al., 2010).

Usage

```
simsar(formula, Glist, theta, 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.

theta	a vector defining the true value of $\theta = (\lambda, \Gamma, \sigma)$ (see the model specification in details).
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 simsar is called.

Details

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

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

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:

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

Value

A list consisting of:

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, tol = 1e-15, maxit = 500, 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.
`theta`	a vector defining the true value of $\theta = (\lambda, \Gamma, \sigma)$ (see the model specification in 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.
`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 modelswith 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 expectation of y.
`Gy`	the average of y among friends.
`GEy`	the average of $E(y)$ friends.
`meff`	a list includinh average and individual marginal effects.
`iteration`	number of iterations performed by 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

```
# Groups' size
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)

# 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")

## 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
```

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 heterogenous 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