Package: PartialNetwork (via r-universe)

October 13, 2024

Encoding UTF-8

Version 1.0.4

Type Package

Title Estimating Peer Effects Using Partial Network Data

Date 2024-06-01

Description Implements IV-estimator and Bayesian estimator for linear-in-means Spatial Autoregressive (SAR) model (see LeSage, 1997 <doi:10.1177/016001769702000107>; Lee, 2004 <doi:10.1111/j.1468-0262.2004.00558.x>; Bramoullé et al., 2009 <doi:10.1016/j.jeconom.2008.12.021>), while assuming that only a partial information about the network structure is available. Examples are when the adjacency matrix is not fully observed or when only consistent estimation of the network formation model is available (see Boucher and Houndetoungan <https://ahoundetoungan.com/files/Papers/PartialNetwork.pdf>).

License GPL-3

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

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

Depends R (>= 3.5.0)

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

LinkingTo Rcpp, RcppArmadillo(>= 0.11.4.4.0), RcppEigen, RcppNumerical, RcppProgress

RoxygenNote 7.3.1

Suggests AER, knitr, rmarkdown, CDatanet, ggplot2, MASS

VignetteBuilder knitr

NeedsCompilation yes

Author Vincent Boucher [aut], Aristide Houndetoungan [cre, aut]

Maintainer Aristide Houndetoungan <ahoundetoungan@gmail.com>

Repository CRAN

Date/Publication 2024-05-15 11:50:13 UTC

Contents

`PartialNetwork-package`

The PartialNetwork package

Description

The **PartialNetwork** package implements instrumental variables (IV) and Bayesian estimators for the linear-in-mean SAR model (e.g. Bramoulle et al., 2009) when the distribution of the network is available, but not the network itself. To make the computations faster **PartialNetwork** uses `C++` through the **Rcpp** package (Eddelbuettel et al., 2011).

Details

Two main functions are provided to estimate the linear-in-mean SAR model using only the distribution of the network. The function `sim.IV` generates valid instruments using the distribution of the network (see Propositions 1 and 2 in Boucher and Houndetoungan (2020)). Once the instruments are constructed, one can estimate the model using standard IV estimators. We recommend the function ivreg from the package **AER** (Kleiber et al., 2020). The function mcmcSAR performs a Bayesian estimation based on an adaptive MCMC (Atchade and Rosenthal, 2005). In that case, the distribution of the network acts as prior distribution for the network.
The package **PartialNetwork** also implements a network formation model based on Aggregate Relational Data (McCormick and Zheng, 2015; Breza et al., 2017). This part of the package relies on the functions rvMF, dvMF and logCpvMF partly implemented in C++, but using code from **movMF** (Hornik and Grun, 2014).

Author(s)

Maintainer: Aristide Houndetoungan <ahoundetoungan@gmail.com>

Authors:

- Vincent Boucher <vincent.boucher@ecn.ulaval.ca>

References

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

Boucher, V., & Houndetoungan, A., 2022, Estimating peer effects using partial network data, *Centre de recherche sur les risques les enjeux economiques et les politiques publiques*, https://ahoundetoungan.com/files/Papers/PartialNetwork.pdf.

Bramoulle, Y., Djebbari, H., & Fortin, B., 2009, Identification of peer effects through social networks, *Journal of econometrics*, 150(1), 41-55, doi:10.1016/j.jeconom.2008.12.021.

Breza, E., Chandrasekhar, A. G., McCormick, T. H., & Pan, M., 2020, Using aggregated relational data to feasibly identify network structure without network data, *American Economic Review*, 110(8), 2454-84, doi:10.1257/aer.20170861

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

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

LeSage, J. P. 1997, Bayesian estimation of spatial autoregressive models, *International regional science review*, 20(1-2), 113-129, doi:10.1177/016001769702000107.

Mardia, K. V., 2014, Statistics of directional data, *Academic press*.

McCormick, T. H., & Zheng, T., 2015, Latent surface models for networks using Aggregated Relational Data, *Journal of the American Statistical Association*, 110(512), 1684-1695, doi:10.1080/01621459.2014.991395.

Wood, A. T., 1994, Simulation of the von Mises Fisher distribution. *Communications in statistics-simulation and computation*, 23(1), 157-164. doi:10.1080/03610919408813161.

See Also

Useful links:

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

dvMF *Density function of the von Mises-Fisher distribution*

Description

Density function for the von Mises-Fisher distribution of dimension p with location parameter equal to mu and intensity parameter eta.

Usage

```
dvMF(z, theta, log.p = FALSE)
```

Arguments

z	is a matrix where each row is a spherical coordinate at which the density will be evaluated.
theta	is a vector of dimension p equal to $\eta\mu$, where η is the concentration parameter, and μ the location parameter.
log.p	is logical; if TRUE, probabilities p are given as log(p).

Value

the densities computed at each point.

Examples

```
# Draw 1000 vectors from vMF with parameter eta = 1 and mu = c(1,0)
z <- rvMF(1000, c(1,0))

# Compute the density at z
dvMF(z, c(1,0))

# Density of c(0, 1, 0, 0) with the parameter eta = 3 and mu = c(0, 1, 0, 0)
dvMF(matrix(c(0, 1, 0, 0), nrow = 1), c(0, 3, 0, 0))
```

fit.dnetwork *Fitting Network Distribution using ARD.*

Description

fit.dnetwork computes the network distribution using the simulations from the posterior distribution of the ARD network formation model. The linking probabilities are also computed for individuals without ARD. The degrees and the gregariousness of the individuals without ARD are computed from the sample with ARD using a k-nearest neighbors method.

Usage

```
fit.dnetwork(
  object,
  X = NULL,
  obsARD = NULL,
  m = NULL,
  burnin = NULL,
  print = TRUE
)
```

Arguments

`object`	`estim.ARD` object returned by `mcmcARD`.
`X`	(required when ARD are available for a sample of individuals) is a matrix of variables describing individuals with ARD and those without ARD. This matrix will be used to compute distance between individuals in the k-nearest neighbors approach. This could be the matrix of traits (see details).
`obsARD`	logical vector of length `nrow(X)` (number of individuals with and without ARD), where the i-th entry equal to `TRUE` if the i-th individual in `X` has ARD and `FALSE` otherwise. If missing, `obsARD = rep(c(TRUE, FALSE), n1, n2)`, where `n1` is the number of individuals with ARD (see details).
`m`	number of neighbors used to compute the gregariousness and the degree for individuals without ARD (default value is `1`).
`burnin`	number of simulations from the posterior distribution used as burn-in. The network distribution will be computed used the simulation from the iteration `burnin + 1`.
`print`	logical; if TRUE, the progression will be printed in the console.

Details

The order of individuals provided through the arguments `traitARD` and `ARD` (when calling the function `mcmcARD`) should fit the order of individuals in `X` and `obsARD`. Especially, the i-th row of `X[obsARD,]` should correspond to the i-th row in `traitARD` or `ARD`.

Value

A list consisting of:

`dnetwork`	posterior mean of the network distribution.
`degree`	posterior mean of the degree.
`nu`	posterior mean of the gregariousness, nu.

Examples

```
set.seed(123)
# GENERATE DATA
# Sample size
```

```
N  <- 500
n  <- 300

# ARD parameters
genzeta <- 1
mu      <- -1.35
sigma   <- 0.37
K       <- 12    # number of traits
P       <- 3     # Sphere dimension

# Generate z (spherical coordinates)
genz    <- rvMF(N,rep(0,P))

# Genetate nu  from a Normal distribution with parameters mu and sigma (The gregariousness)
gennu   <- rnorm(N,mu,sigma)

# compute degrees
gend <- N*exp(gennu)*exp(mu+0.5*sigma^2)*exp(logCpvMF(P,0) - logCpvMF(P,genzeta))

# Link probabilities
Probabilities <- sim.dnetwork(gennu,gend,genzeta,genz)

# Adjacency matrix
G <- sim.network(Probabilities)

# Generate vk, the trait location
genv <- rvMF(K,rep(0,P))

# set fixed some vk  distant
genv[1,] <- c(1,0,0)
genv[2,] <- c(0,1,0)
genv[3,] <- c(0,0,1)

# eta, the intensity parameter
geneta   <-abs(rnorm(K,2,1))

# Build traits matrix
densityatz      <- matrix(0,N,K)
for(k in 1:K){
  densityatz[,k] <- dvMF(genz,genv[k,]*geneta[k])
}

trait       <- matrix(0,N,K)
NK          <- floor(runif(K, 0.8, 0.95)*colSums(densityatz)/apply(densityatz, 2, max))
for (k in 1:K) {
  trait[,k]       <- rbinom(N, 1, NK[k]*densityatz[,k]/sum(densityatz[,k]))
}

# print a percentage of people having a trait
colSums(trait)*100/N

# Build ARD
ARD         <- G %*% trait
```

```
# generate b
genb        <- numeric(K)
for(k in 1:K){
  genb[k]   <- sum(G[,trait[,k]==1])/sum(G)
}

############ ARD Posterior distribution ###################
# EXAMPLE 1: ARD observed for the entire population
# initialization
d0     <- exp(rnorm(N)); b0 <- exp(rnorm(K)); eta0 <- rep(1,K);
zeta0  <- 1; z0 <- matrix(rvMF(N,rep(0,P)),N); v0 <- matrix(rvMF(K,rep(0,P)),K)

# We need to fix some of the vk and bk for identification (see Breza et al. (2020) for details).
vfixcolumn      <- 1:6
bfixcolumn      <- c(3, 5)
b0[bfixcolumn]  <- genb[bfixcolumn]
v0[vfixcolumn,] <- genv[vfixcolumn,]

start  <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0, "zeta" = zeta0)
# MCMC ARD
out    <- mcmcARD(Y = ARD, traitARD = trait, start = start, fixv = vfixcolumn,
                  consb = bfixcolumn, iteration = 5000)

# fit network distribution
dist   <- fit.dnetwork(out)

plot(rowSums(dist$dnetwork), gend)
abline(0, 1, col = "red")

# EXAMPLE 2: ARD observed for a sample of the population
# observed sample
selectARD   <- sort(sample(1:N, n, FALSE))
traitard    <- trait[selectARD,]
ARD         <- ARD[selectARD,]
logicalARD  <- (1:N) %in% selectARD

# initianalization
d0     <- exp(rnorm(n)); b0 <- exp(rnorm(K)); eta0 <- rep(1,K);
zeta0  <- 1; z0 <- matrix(rvMF(n,rep(0,P)),n); v0 <- matrix(rvMF(K,rep(0,P)),K)

# We need to fix some of the vk and bk for identification (see Breza et al. (2020) for details).
vfixcolumn      <- 1:6
bfixcolumn      <- c(3, 5)
b0[bfixcolumn]  <- genb[bfixcolumn]
v0[vfixcolumn,] <- genv[vfixcolumn,]

start  <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0, "zeta" = zeta0)
# MCMC ARD
out    <- mcmcARD(Y = ARD, traitARD = traitard, start = start, fixv = vfixcolumn,
                  consb = bfixcolumn, iteration = 5000)

# fit network distribution
```

```
dist   <- fit.dnetwork(out, X = trait, obsARD = logicalARD, m = 1)

library(ggplot2)
ggplot(data.frame("etimated.degree" = dist$degree,
                  "true.degree"     = gend,
                  "observed"        = ifelse(logicalARD, TRUE, FALSE)),
       aes(x = etimated.degree, y = true.degree, colour = observed)) +
  geom_point()
```

`logCpvMF` *Normalization constant of the von Mises-Fisher distribution*

Description

log of the Normalization Constant for the von Mises-Fisher distribution of dimension `p` with intensity parameter `eta`.

Usage

```
logCpvMF(p, eta)
```

Arguments

`p`	is the dimension of the hypersphere.
`eta`	is the intensity parameter.

Value

the log of normalization constant of the von Mises-Fisher distribution.

Examples

```
logCpvMF(2, 3.1)
```

`mcmcARD` *Estimate network model using ARD*

Description

`mcmcARD` estimates the network model proposed by Breza et al. (2020).

Usage

```
mcmcARD(
  Y,
  traitARD,
  start,
  fixv,
  consb,
  iteration = 2000L,
  sim.d = TRUE,
  sim.zeta = TRUE,
  hyperparms = NULL,
  ctrl.mcmc = list()
)
```

Arguments

`Y`	is a matrix of ARD. The entry (i, k) is the number of i's friends having the trait k.
`traitARD`	is the matrix of traits for individuals with ARD. The entry (i, k) is equal to 1 if i has the trait k and 0 otherwise.
`start`	is a list containing starting values of `z` (matrix of dimension $N \times p$), `v` (matrix of dimension $K \times p$), `d` (vector of dimension N), `b` (vector of dimension K), `eta` (vector of dimension K) and `zeta` (scalar).
`fixv`	is a vector setting which location parameters are fixed for identifiability. These fixed positions are used to rotate the latent surface back to a common orientation at each iteration using a Procrustes transformation (see Section Identification in Details).
`consb`	is a vector of the subset of β_k constrained to the total size (see Section Identification in Details).
`iteration`	is the number of MCMC steps to be performed.
`sim.d`	is logical indicating whether the degree `d` will be updated in the MCMC. If `sim.d = FALSE`, the starting value of `d` in the argument `start` is set fixed along the MCMC.
`sim.zeta`	is logical indicating whether the degree `zeta` will be updated in the MCMC. If `sim.zeta = FALSE`, the starting value of `zeta` in the argument `start` is set fixed along the MCMC.
`hyperparms`	is an 8-dimensional vector of hyperparameters (in this order) μ_d, σ_d, μ_b, σ_b, α_η, β_η, α_ζ and β_ζ (see Section Model in Details).
`ctrl.mcmc`	is a list of MCMC controls (see Section MCMC control in Details).

Details

The linking probability is given by

Model:

$$P_{ij} \propto \exp(\nu_i + \nu_j + \zeta \mathbf{z}_i \mathbf{z}_j).$$

McCormick and Zheng (2015) write the likelihood of the model with respect to the spherical coordinate $\mathbf{z}_i$, the trait locations $\mathbf{v}_k$, the degree d_i, the fraction of ties in the network that are made with members of group k b_k, the trait intensity parameter η_k and ζ. The following prior distributions are defined.

$$\mathbf{z}_i \sim Uniform\ von\ Mises - Fisher$$

$$\mathbf{v}_k \sim Uniform\ von\ Mises - Fisher$$

$$d_i \sim log - \mathcal{N}(\mu_d, \sigma_d)$$

$$b_k \sim log - \mathcal{N}(\mu_b, \sigma_b)$$

$$\eta_k \sim Gamma(\alpha_\eta, \beta_\eta)$$

$$\zeta \sim Gamma(\alpha_\zeta, \beta_\zeta)$$

Identification:

For identification, some $\mathbf{v}_k$ and b_k need to be exogenously fixed around their given starting value (see McCormick and Zheng, 2015 for more details). The parameter `fixv` can be used to set the desired value for $\mathbf{v}_k$ while `fixb` can be used to set the desired values for b_k.

MCMC control:

During the MCMC, the jumping scales are updated following Atchade and Rosenthal (2005) in order to target the acceptance rate of each parameter to the `target` values. This requires to set minimal and maximal jumping scales through the parameter `ctrl.mcmc`. The parameter `ctrl.mcmc` is a list which can contain the following named components.

- `target`: The default value is `rep(0.44, 5)`. The target of every $\mathbf{z}_i$, d_i, b_k, η_k and ζ is 0.44.
- `jumpmin`: The default value is `c(0,1,1e-7,1e-7,1e-7)*1e-5`. The minimal jumping of every $\mathbf{z}_i$ is 0, every d_i is 10^{-5}, and every b_k, η_k and ζ is 10^{-12}.
- `jumpmax`: The default value is `c(100,1,1,1,1)*20`. The maximal jumping scale is 20 except for $\mathbf{z}_i$ which is set to 2000.
- `print`: A logical value which indicates if the MCMC progression should be printed in the console. The default value is `TRUE`.

Value

A list consisting of:

`n`	dimension of the sample with ARD.
`K`	number of traits.
`p`	hypersphere dimension.
`time`	elapsed time in second.
`iteration`	number of MCMC steps performed.
`simulations`	simulations from the posterior distribution.
`hyperparms`	return value of hyperparameters (updated and non updated).
`accept.rate`	list of acceptance rates.
`start`	starting values.
`ctrl.mcmc`	return value of `ctrl.mcmc`.

Examples

```
# Sample size
N       <- 500

# ARD parameters
genzeta <- 1
mu      <- -1.35
sigma   <- 0.37
K       <- 12    # number of traits
P       <- 3     # Sphere dimension


# Generate z (spherical coordinates)
genz    <- rvMF(N,rep(0,P))

# Generate nu  from a Normal distribution with parameters mu and sigma (The gregariousness)
gennu   <- rnorm(N,mu,sigma)

# compute degrees
gend <- N*exp(gennu)*exp(mu+0.5*sigma^2)*exp(logCpvMF(P,0) - logCpvMF(P,genzeta))

# Link probabilities
Probabilities <- sim.dnetwork(gennu,gend,genzeta,genz)

# Adjacency matrix
G <- sim.network(Probabilities)

# Generate vk, the trait location
genv <- rvMF(K,rep(0,P))

# set fixed some vk  distant
genv[1,] <- c(1,0,0)
genv[2,] <- c(0,1,0)
genv[3,] <- c(0,0,1)

# eta, the intensity parameter
geneta   <-abs(rnorm(K,2,1))

# Build traits matrix
densityatz       <- matrix(0,N,K)
for(k in 1:K){
  densityatz[,k] <- dvMF(genz,genv[k,]*geneta[k])
}

trait       <- matrix(0,N,K)
NK          <- floor(runif(K, 0.8, 0.95)*colSums(densityatz)/apply(densityatz, 2, max))
for (k in 1:K) {
  trait[,k]  <- rbinom(N, 1, NK[k]*densityatz[,k]/sum(densityatz[,k]))
}

# print a percentage of people having a trait
colSums(trait)*100/N
```

```
# Build ARD
ARD        <- G %*% trait

# generate b
genb       <- numeric(K)
for(k in 1:K){
  genb[k]   <- sum(G[,trait[,k]==1])/sum(G)
}

############ ARD Posterior distribution ###################
# initialization
d0     <- exp(rnorm(N)); b0 <- exp(rnorm(K)); eta0 <- rep(1,K);
zeta0  <- 05; z0 <- matrix(rvMF(N,rep(0,P)),N); v0 <- matrix(rvMF(K,rep(0,P)),K)

# We need to fix some of the vk and bk for identification (see Breza et al. (2020) for details).
vfixcolumn      <- 1:6
bfixcolumn      <- c(3, 5)
b0[bfixcolumn]  <- genb[bfixcolumn]
v0[vfixcolumn,] <- genv[vfixcolumn,]
start  <- list("z" = z0, "v" = v0, "d" = d0, "b" = b0, "eta" = eta0, "zeta" = zeta0)

# MCMC
out   <- mcmcARD(Y = ARD, traitARD = trait, start = start, fixv = vfixcolumn,
                 consb = bfixcolumn, iteration = 5000)

# plot simulations
# plot d
plot(out$simulations$d[,100], type = "l", col = "blue", ylab = "")
abline(h = gend[100], col = "red")

# plot coordinates of individuals
i <- 123 # individual 123
{
  lapply(1:3, function(x) {
   plot(out$simulations$z[i, x,] , type = "l", ylab = "", col = "blue", ylim = c(-1, 1))
    abline(h = genz[i, x], col = "red")
  })
}

# plot coordinates of traits
k <- 8
{
  lapply(1:3, function(x) {
   plot(out$simulations$v[k, x,] , type = "l", ylab = "", col = "blue", ylim = c(-1, 1))
    abline(h = genv[k, x], col = "red")
  })
}
```

---

mcmcSAR *Bayesian Estimator of SAR model*

Description

`mcmcSAR` implements the Bayesian estimator of the linear-in-mean SAR model when only the linking probabilities are available or can be estimated.

Usage

```
mcmcSAR(
  formula,
  contextual,
  start,
  G0.obs,
  G0 = NULL,
  mlinks = list(),
  hyperparms = list(),
  ctrl.mcmc = list(),
  iteration = 2000L,
  data
)
```

Arguments

`formula` object of class formula: a symbolic description of the model. The `formula` should be as for example `y ~ x1 + x2 | x1 + x2` where `y` is the endogenous vector, the listed variables before the pipe, `x1`, `x2` are the individual exogenous variables and the listed variables after the pipe, `x1`, `x2` are the contextual observable variables. Other formulas may be `y ~ x1 + x2` for the model without contextual effects, `y ~ -1 + x1 + x2 | x1 + x2` for the model without intercept, or `y ~ x1 + x2 | x2 + x3` to allow the contextual variables to be different from the individual variables.

`contextual` (optional) logical; if true, this means that all individual variables will be set as contextual variables. Set `formula` as `y ~ x1 + x2` and `contextual` as `TRUE` is equivalent to set formula as `y ~ x1 + x2 | x1 + x2`.

`start` (optional) vector of starting value of the model parameter as $(\beta' \ \gamma' \ \alpha \ \sigma^2)'$, where β is the individual variables parameter, γ is the contextual variables parameter, α is the peer effect parameter and σ^2 the variance of the error term. If the `start` is missing, a Maximum Likelihood estimator will be used, where the network matrix is that given through the argument `G0` (if provided) or generated from it distribution.

`G0.obs` list of matrices (or simply matrix if the list contains only one matrix) indicating the part of the network data which is observed. If the (i,j)-th element of the m-th matrix is one, then the element at the same position in the network data will be considered as observed and will not be inferred in the MCMC. In contrast, if the (i,j)-th element of the m-th matrix is zero, the element at the same position in the network data will be considered as a starting value of the missing link which will be inferred. `G0.obs` can also take `"none"` when no part of the network data is observed (equivalent to the case where all the entries are zeros) and `"all"` when the network data is fully observed (equivalent to the case where all the entries are ones).

`G0`	list of sub-network matrices (or simply network matrix if there is only one sub-network). `G0` is made up of starting values for the entries with missing network data and observed values for the entries with observed network data. `G0` is optional when `G0.obs = "none"`.
`mlinks`	list specifying the network formation model (see Section Network formation model in Details).
`hyperparms`	(optional) is a list of hyperparameters (see Section Hyperparameters in Details).
`ctrl.mcmc`	list of MCMC controls (see Section MCMC control in Details).
`iteration`	number of MCMC steps to be performed.
`data`	optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If missing, the variables are taken from `environment(formula)`, typically the environment from which `mcmcSAR` is called.

Details

Outcome model:

The model is given by

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{G}\mathbf{X}\gamma + \alpha\mathbf{G}\mathbf{y} + \epsilon.$$

where

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

The parameters to estimate in this model are the matrix $\mathbf{G}$, the vectors β, γ and the scalar α, σ. Prior distributions are assumed on $\mathbf{A}$, the adjacency matrix in which $\mathbf{A}_{ij} = 1$ if i is connected to j and $\mathbf{A}_{ij} = 0$ otherwise, and on β, γ, α and σ^2.

$$\mathbf{A}_{ij} \sim Bernoulli(\mathbf{P}_{ij})$$

$$(\beta' \ \gamma')'|\sigma^2 \sim \mathcal{N}(\mu_\theta, \sigma^2\Sigma_\theta)$$

$$\zeta = \log\left(\frac{\alpha}{1-\alpha}\right) \sim \mathcal{N}(\mu_\zeta, \sigma_\zeta^2)$$

$$\sigma^2 \sim IG(\frac{a}{2}, \frac{b}{2})$$

where $\mathbf{P}$ is the linking probability. The linking probability is an hyperparameters that can be set fixed or updated using a network formation model.

Network formation model:

The linking probability can be set fixed or updated using a network formation model. Information about how $\mathbf{P}$ should be handled in in the MCMC can be set through the argument `mlinks` which should be a list with named elements. Divers specifications of network formation model are possible. The list assigned to `mlist` should include an element named `model`. The expected values of `model` are `"none"` (default value), `"logit"`, `"probit"`, and `"latent space"`.

- `"none"` means that the network distribution $\mathbf{P}$ is set fixed throughout the MCMC,
- `"probit"` or `"logit"` implies that the network distribution $\mathbf{P}$ will be updated using a Probit or Logit model,
- `"latent spate"` means that $\mathbf{P}$ will be updated following Breza et al. (2020).

Fixed network distribution:
To set $\mathbf{P}$ fixed, `mlinks` could contain,

- `dnetwork`, a list, where the m-th elements is the matrix of link probability in the m-th sub-network.
- `model = "none"` (optional as `"none"` is the default value).

Probit and Logit models:
For the Probit and Logit specification as network formation model, the following elements could be declared in `mlinks`.

- `model = "probit"` or `model = "logit"`.
- `mlinks.formula` object of class formula: a symbolic description of the Logit or Probit model. The `formula` should only specify the explanatory variables, as for example `~ x1 + x2`, the variables `x1` and `x2` are the dyadic observable characteristics. Each variable should verify `length(x) == sum(N^2 - N)`, where `N` is a vector of the number of individual in each sub-network. Indeed, `x` will be associated with the entries $(1, 2)$; $(1, 3)$; $(1, 4)$; ...; $(2, 1)$; $(2, 3)$; $(2, 4)$; ... of the linking probability and as so, in all the sub-networks. Functions `mat.to.vec` and `vec.to.mat` can be used to convert a list of dyadic variable as in matrix form to a format that suits `mlinks.formula`.
- `weights` (optional) is a vector of weights of observed entries. This is important to address the selection problem of observed entries. Default is a vector of ones.
- `estimates` (optional when a part of the network is observed) is a list containing `rho`, a vector of the estimates of the Probit or Logit parameters, and `var.rho` the covariance matrix of the estimator. These estimates can be automatically computed when a part of the network data is available. In this case, `rho` and the unobserved part of the network are updated without using the observed part of the network. The latter is assumed non-stochastic in the MCMC. In addition, if `G0.obs = "none"`, `estimates` should also include `N`, a vector of the number of individuals in each sub-network.
- `prior` (optional) is a list containing `rho`, a vector of the prior beliefs on `rho`, and `var.rho` the prior covariance matrix of `rho`. This input is relevant only when the observed part of the network is used to update `rho`, i.e. only when `estimates = NULL` (so, either `estimates` or `prior` should be `NULL`).
  To understand the difference between `estimates` and `prior`, note that `estimates` includes initial estimates of `rho` and `var.rho`, meaning that the observed part of the network is not used in the MCMC to update `rho`. In contrast, `prior` contains the prior beliefs of the user, and therefore, `rho` is updated using this prior and information from the observed part of the network. In addition, if `G0.obs = "none"`, `prior` should also include `N`, a vector of the number of individuals in each sub-network.
- `mlinks.data` optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the dyadic observable characteristics If missing, the variables will be taken from `environment(mlinks.formula)`, typically the environment from which `mcmcARD` is called.

Latent space models:
The following element could be declared in `mlinks`.

- `model = "latent space"`.
- `estimates` a list of objects of class `mcmcARD`, where the m-th element is Breza et al. (2020) estimator as returned by the function `mcmcARD` in the m-th sub-network.
- `mlinks.data` (required only when ARD are partially observed) is a list of matrices, where the m-th element is the variable matrix to use to compute distance between individuals

(could be the list of traits) in the m-th sub-network. The distances will be used to compute gregariousness and coordinates for individuals without ARD by k-nearest neighbors approach.

- `obsARD` (required only when ARD are partially observed) is a list of logical vectors, where the i-th entry of the m-th vector indicates by `TRUE` or `FALSE` if the i-th individual in the m-th sub-network has ARD or not.
- `mARD` (optional, default value is `rep(1, M)`) is a vector indicating the number of neighbors to use in each sub-network.
- `burninARD` (optional) set the burn-in to summarize the posterior distribution in `estimates`.

Hyperparameters:

All the hyperparameters can be defined through the argument `hyperparms` (a list) and should be named as follow.

- `mutheta`, the prior mean of $(\beta' \ \gamma')'|\sigma^2$. The default value assumes that the prior mean is zero.
- `invstheta` as Σ_θ^{-1}. The default value is a diagonal matrix with 0.01 on the diagonal.
- `muzeta`, the prior mean of ζ. The default value is zero.
- `invszeta`, the inverse of the prior variance of ζ with default value equal to 2.
- `a` and `b` which default values equal to 4.2 and 2.2 respectively. This means for example that the prior mean of σ^2 is 1.

Inverses are used for the prior variance through the argument `hyperparms` in order to allow non informative prior. Set the inverse of the prior variance to 0 is equivalent to assume a non informative prior.

MCMC control:

During the MCMC, the jumping scales of α and ρ are updated following Atchade and Rosenthal (2005) in order to target the acceptance rate to the `target` value. This requires to set a minimal and a maximal jumping scales through the parameter `ctrl.mcmc`. The parameter `ctrl.mcmc` is a list which can contain the following named components.

- `target`: the default value is `c("alpha" = 0.44, "rho" = 0.234)`.
- `jumpmin`: the default value is `c("alpha" = 1e-5, "rho" = 1e-5)`.
- `jumpmax`: the default value is `c("alpha" = 10, "rho" = 10)`.
- `print.level`: an integer in {0, 1, 2} that indicates if the MCMC progression should be printed in the console. If 0, the MCMC progression is not be printed. If 1 (default value), the progression is printed and if 2, the simulations from the posterior distribution are printed.
- `block.max`: The maximal number of entries that can be updated simultaneously in $\mathbf{A}$. It might be more efficient to update simultaneously 2 or 3 entries (see Boucher and Houndetoungan, 2022).

If `block.max` > 1, several entries are randomly chosen from the same row and updated simultaneously. The number of entries chosen is randomly chosen between 1 and `block.max`. In addition, the entries are not chosen in order. For example, on the row i, the entries (i, 5) and (i, 9) can be updated simultaneously, then the entries (i, 1), (i, 3), (i, 8), and so on.

Value

A list consisting of:

n.group	number of groups.
N	vector of each group size.
time	elapsed time to run the MCMC in second.
iteration	number of MCMC steps performed.
posterior	matrix (or list of matrices) containing the simulations.
hyperparms	return value of hyperparms.
mlinks	return value of mlinks.
accept.rate	acceptance rates.
prop.net	proportion of observed network data.
method.net	network formation model specification.
start	starting values.
formula	input value of formula and mlinks.formula.
contextual	input value of contextual.
ctrl.mcmc	return value of ctrl.mcmc.

See Also

smmSAR, sim.IV

Examples

```
# We assume that the network is fully observed
# See our vignette for examples where the network is partially observed
# Number of groups
M             <- 50
# size of each group
N             <- rep(30,M)
# individual effects
beta          <- c(2,1,1.5)
# contextual effects
gamma         <- c(5,-3)
# endogenous effects
alpha         <- 0.4
# std-dev errors
se            <- 1
# prior distribution
prior         <- runif(sum(N*(N-1)))
prior         <- vec.to.mat(prior, N, normalise = FALSE)
# covariates
X             <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0            <- sim.network(prior)
# normalise
G0norm        <- norm.network(G0)
# simulate dependent variable use an external package
y             <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                                  theta = c(alpha, beta, gamma, se))
```

```
y             <- y$y
# dataset
dataset       <- as.data.frame(cbind(y, X1 = X[,1], X2 = X[,2]))
out.none1     <- mcmcSAR(formula = y ~ X1 + X2, contextual = TRUE, G0.obs = "all",
                         G0 = G0, data = dataset, iteration = 1e4)
summary(out.none1)
plot(out.none1)
plot(out.none1, plot.type = "dens")
```

---

peer.avg *Computing peer average value*

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

sim.network

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)
```

`plot.mcmcSAR` *Plotting estimation of Bayesian SAR model*

Description

Plotting the simulation from the posterior distribution as well as the density functions of Bayesian SAR model parameter. For more details about the graphical parameter arguments, see par.

Usage

```
## S3 method for class 'mcmcSAR'
plot(x, plot.type = "sim", burnin = NULL, which.parms = "theta", ...)

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

Arguments

`x`	object of class "mcmcSAR", output of the function `mcmcSAR` or object of class "plot.mcmcSAR", output of the function `plot.mcmcSAR`.
`plot.type`	character indicating the type of plot: `"sim"` for plotting the simulation from the posterior distribution or `"dens"` for plotting the posterior density functions.
`burnin`	number of MCMC steps which will be considered as burn-in iterations. If `NULL` (default value), the 50% first MCMC steps performed are used as burn-in iterations.
`which.parms`	character indicating the parameters whose the posterior distribution will be plotted: `"theta"` for the parameters of the outcome model and `"rho"` for the parameters of the network formation model.
`...`	arguments to be passed to methods, such as par.

Value

A list consisting of:

`n.group`	number of groups.
`N`	vector of each group size.
`iteration`	number of MCMC steps performed.
`burnin`	number of MCMC steps which will be considered as burn-in iterations.
`posterior`	summary of the posterior distribution to be plotted.
`hyperparms`	return value of `hyperparms`.

`accept.rate`	acceptance rate of zeta.
`propG0.obs`	proportion of observed network data.
`method.net`	network formation model specification.
`formula`	input value of `formula`.
`ctrl.mcmc`	return value of `ctrl.mcmc`.
`which.parms`	return value of `which.parms`.
`plot.type`	type of the plot.
`...`	arguments passed to methods.

`remove.ids` *Removes IDs with NA in a list of adjacency matrices optimally*

Description

The function optimally removes identifiers with NA in a list of 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)
```

rvMF — *Simulation from the von Mises-Fisher distribution*

Description

Random generation for the von Mises-Fisher distribution of dimension `p` with location parameter `mu` and intensity parameter `eta` (see Wood, 1994; Mardia, 2014).

Usage

```
rvMF(size, theta)
```

Arguments

`size`	is the number of simulations.
`theta`	is the parameter as `eta*mu`.

Value

A matrix whose each row is a random draw from the distribution.

Examples

```
# Draw 10 vectors from vMF with parameters eta = 1 and mu = c(1,0)
rvMF(10,c(1,0))

# Draw 10 vectors from vMF with parameters eta = sqrt(14) and mu proportional to (2,1,3)
rvMF(10,c(2,1,3))

# Draw from the vMF distribution with mean direction proportional to c(1, -1)
# and concentration parameter 3
rvMF(10, 3 * c(1, -1) / sqrt(2))
```

sim.dnetwork — *Simulation of the distribution of the network for Breza et al. (2020)*

Description

Compute the distribution of the network following McCormick and Zheng (2015) and Breza et al. (2020).

Usage

```
sim.dnetwork(nu, d, zeta, z)
```

Arguments

nu	is the vector of gregariousness.
d	is the vector of degrees.
zeta	is a scale parameter that captures the influence of the latent positions on the link probabilities.
z	is a matrix where each row is a spherical coordinate.

Value

a matrix of linking probabilities.

See Also

sim.network

Examples

```
N       <- 500
zeta    <- 1

# Generate the spherical coordinates
z       <- rvMF(N, c(0, 0, 0))

# Genetate the gregariousness
nu      <- rnorm(N, -1.35, 0.37)

# Generate degrees
d       <- runif(N, 0, 45)

dist    <- sim.dnetwork(nu, d, zeta, z)
```

---

sim.IV *Instrument Variables for SAR model*

Description

sim.IV generates Instrument Variables (IV) for linear-in-mean SAR models using only the distribution of the network. See Propositions 1 and 2 of Boucher and Houndetoungan (2020).

Usage

```
sim.IV(
  dnetwork,
  X,
  y = NULL,
  replication = 1L,
  power = 1L,
  exp.network = FALSE
)
```

Arguments

`dnetwork`	network matrix of list of sub-network matrices, where the (i, j)-th position is the probability that i be connected to j.
`X`	matrix of the individual observable characteristics.
`y`	(optional) the endogenous variable as a vector.
`replication`	(optional, default = 1) is the number of repetitions (see details).
`power`	(optional, default = 1) is the number of powers of the interaction matrix used to generate the instruments (see details).
`exp.network`	(optional, default = FALSE) indicates if simulated network should be exported.

Details

Bramoulle et al. (2009) show that one can use GX, G^2X, ..., G^PX as instruments for Gy, where P is the maximal power desired. `sim.IV` generate approximation of those instruments, based on Propositions 1 and 2 in Boucher and Houndetoungan (2020) (see also below). The argument `power` is the maximal power desired.
When Gy and the instruments GX, G^2X, ..., G^PX are not observed, Boucher and Houndetoungan (2022) show that we can use one drawn from the distribution of the network in order to approximate Gy, but that the same draw should not be used to approximate the instruments. Thus, each component in the function's output gives `G1y` and `G1X` computed with the same network and `G2X` computed with another network, which can be used in order to approximate the instruments. This process can be replicated several times and the argument `replication` can be used to set the number of replications desired.

Value

list of `replication` components. Each component is a list containing `G1y` (if the argument `y` was provided), `G1` (if `exp.network = TRUE`), `G2` (if `exp.network = TRUE`) , `G1X`, and `G2X` where `G1` and `G2` are independent draws of network from the distribution (see details).

`G1y`	is an approximation of Gy.
`G1X`	is an approximation of G^pX with the same network draw as that used in `G1y`. `G1X` is an array of dimension $N \times K \times power$, where K is the number of column in `X`. For any $p \in \{1, 2, ..., power\}$, the approximation of G^pX is given by `G1X[,,p]`.
`G2X`	is an approximation of G^pX with a different different network. `G2X` is an array of dimension $N \times K \times power$. For any $p \in \{1, 2, ..., power\}$, the approximation of G^pX is given by `G2X[,,p]`.

See Also

`mcmcSAR`

Examples

```
library(AER)
# Number of groups
M            <- 30
# size of each group
N            <- rep(50,M)
# individual effects
beta         <- c(2,1,1.5)
# endogenous effects
alpha        <- 0.4
# std-dev errors
se           <- 2
# prior distribution
prior        <- runif(sum(N*(N-1)))
prior        <- vec.to.mat(prior, N, normalise = FALSE)
# covariates
X            <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# true network
G0           <- sim.network(prior)
# normalise
G0norm       <- norm.network(G0)
# simulate dependent variable use an external package
y            <- CDatanet::simsar(~ X, contextual = FALSE, Glist = G0norm,
                                 theta = c(alpha, beta, se))
y            <- y$y
# generate instruments
instr        <- sim.IV(prior, X, y, replication = 1, power = 1)

GY1c1        <- instr[[1]]$G1y       # proxy for Gy (draw 1)
GXc1         <- instr[[1]]$G1X[,,1]  # proxy for GX (draw 1)
GXc2         <- instr[[1]]$G2X[,,1]  # proxy for GX (draw 2)
# build dataset
# keep only instrument constructed using a different draw than the one used to proxy Gy
dataset           <- as.data.frame(cbind(y, X, GY1c1, GXc1, GXc2))
colnames(dataset) <- c("y","X1","X2","G1y", "G1X1", "G1X2", "G2X1", "G2X2")

# Same draws
out.iv1           <- ivreg(y ~ X1 + X2 + G1y | X1 + X2 + G1X1 + G1X2, data = dataset)
summary(out.iv1)

# Different draws
out.iv2           <- ivreg(y ~ X1 + X2 + G1y | X1 + X2 + G2X1 + G2X2, data = dataset)
summary(out.iv2)
```

`sim.network` *Simulating network data*

Description

Simulating network data

Usage

```
sim.network(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.

See Also

sim.dnetwork

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         <- sim.network(dnetwork)
```

---

smmSAR — *Simulated Method of Moments (SMM) Estimator of SAR model*

Description

smmSAR implements the Simulated Method of Moments (SMM) estimator of the linear-in-mean SAR model when only the linking probabilities are available or can be estimated.

Usage

```
smmSAR(
  formula,
  contextual = FALSE,
  fixed.effects = FALSE,
  dnetwork,
  W = "identity",
  smm.ctr = list(R = 30L, iv.power = 2L, opt.tol = 1e-04, smoother = FALSE, print =
    FALSE),
```

```
  cond.var = TRUE,
  data
)
```

Arguments

- `formula` — object of class formula: a symbolic description of the model. The `formula` should be as for example `y ~ x1 + x2 | gy | gx1 + gx2` where `y` is the endogenous vector, the listed variables before the pipe, `x1`, `x2` are the individual exogenous variables, `gy` is the average of `y` among friends, and `gx1`, `gx2` are the contextual observed variables. If `gy` is observed and `gx1`, `gx2` are not, the formula should be `y ~ x1 + x2 | gy`. If `gy` is not observed and `gx1`, `gx2` are, the formula should be `y ~ x1 + x2 || gx1 + gx2`. If `gy`, `gx1`, and `gx2` are not observed, the the formula should simply be `y ~ x1 + x2`.
- `contextual` — logical; if true, this means that all individual variables will be set as contextual variables. In contrast `mcmcSAR`, `formula` as `y ~ x1 + x2` and `contextual` as `TRUE` is not equivalent to set formula as `y ~ x1 + x2 || gx1 + gx2`. `formula = y ~ x1 + x2` means that `gy`, `gx1`, and `gx2` are not observed and `contextual = TRUE` means that the estimated model includes contextual effects.
- `fixed.effects` — logical; if true, group heterogeneity is included as fixed effects.
- `dnetwork` — a list, where the m-th elements is the matrix of link probability in the m-th sub-network.
- `W` — is the weighted-matrix in the objective function of the SMM.
- `smm.ctr` — is the list of some control parameters (see details).
- `cond.var` — logical; if true the estimator variance conditional on `dnetwork` will be computed.
- `data` — optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If missing, the variables are taken from `environment(formula)`, typically the environment from which `smmSAR` is called.

Details

The parameter `smm.ctr` is the list of some control parameters such as:

- `R` numbers of draws R (in the package, we assume S = 1 and T = 1);
- `iv.power` number of powers of the network matrix `G` to be used to construct instruments;
- `opt.tol` optimization tolerance that will be used in optimize;
- `smoother` (logical) which indicates if draws should be performed using the smoother simulator;
- `h` bandwith of the smoother (required if `smoother = TRUE`);
- `print` (logical) indicates if the optimization process should be printed step by step.

Value

A list consisting of:

`n.group`	number of groups.
`N`	vector of each group size.
`time`	elapsed time to run the SMM in second.
`estimates`	vector of estimated parameters.
`formula`	input value of `formula`.
`contextual`	input value of `contextual`.
`fixed.effects`	input value of `fixed.effects`.
`smm.ctr`	input value of `smm.ctr`.
`details`	other details of the model.

Examples

```
# Number of groups
M        <- 100
# size of each group
N        <- rep(30,M)
# covariates
X        <- cbind(rnorm(sum(N),0,5),rpois(sum(N),7))
# network formation model parameter
rho      <- c(-0.8, 0.2, -0.1)
# individual effects
beta     <- c(2, 1, 1.5, 5, -3)
# endogenous effects
alpha    <- 0.4
# std-dev errors
se       <- 1
# network
tmp      <- c(0, cumsum(N))
X1l      <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],1])
X2l      <- lapply(1:M, function(x) X[c(tmp[x] + 1):tmp[x+1],2])
dist.net <- function(x, y) abs(x - y)
X1.mat   <- lapply(1:M, function(m) {
  matrix(kronecker(X1l[[m]], X1l[[m]], FUN = dist.net), N[m])})
X2.mat   <- lapply(1:M, function(m) {
  matrix(kronecker(X2l[[m]], X2l[[m]], FUN = dist.net), N[m])})
Xnet     <- as.matrix(cbind("Const" = 1,
                            "dX1"   = mat.to.vec(X1.mat),
                            "dX2"   = mat.to.vec(X2.mat)))
ynet     <- Xnet %*% rho
ynet     <- c(1*((ynet + rlogis(length(ynet))) > 0))
G0       <- vec.to.mat(ynet, N, normalise = FALSE)
# normalise
G0norm   <- norm.network(G0)
# Matrix GX
GX       <- peer.avg(G0norm, X)
# simulate dependent variable use an external package
```

```
y        <- CDatanet::simsar(~ X, contextual = TRUE, Glist = G0norm,
                             theta = c(alpha, beta, se))
Gy       <- y$Gy
y        <- y$y
# build dataset
dataset           <- as.data.frame(cbind(y, X, Gy, GX))
colnames(dataset) <- c("y","X1","X2", "Gy", "GX1", "GX2")
nNet      <- nrow(Xnet) # network formation model sample size
Aobs      <- sample(1:nNet, round(0.3*nNet)) # We observed 30%
# We can estimate rho using the gml function from the stats package
logestim  <- glm(ynet[Aobs] ~ -1 + Xnet[Aobs,], family = binomial(link = "logit"))
slogestim <- summary(logestim)
rho.est   <- logestim$coefficients
rho.var   <- slogestim$cov.unscaled # we also need the covariance of the estimator

d.logit     <- lapply(1:M, function(x) {
  out       <- 1/(1 + exp(-rho.est[1] - rho.est[2]*X1.mat[[x]] -
                            rho.est[3]*X2.mat[[x]]))
  diag(out) <- 0
  out})
smm.logit   <- smmSAR(y ~ X1 + X2, dnetwork = d.logit, contextual = TRUE,
                      smm.ctr  = list(R = 100L, print = TRUE), data = dataset)
summary(smm.logit, dnetwork = d.logit, data = dataset)
```

---

summary.mcmcSAR *Summarizing Bayesian SAR Model*

Description

Summary and print methods for the class mcmcSAR.

Usage

```
## S3 method for class 'mcmcSAR'
summary(object, alpha = 0.95, plot.type = NULL, burnin = NULL, ...)

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

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

Arguments

object	an object of class "mcmcSAR", output of the function mcmcSAR.
alpha	(optional, default = 0.95), the significance level of parameter.

plot.type	(optional) a character that indicate if the simulations from the posterior distribution should be printed (if plot.type = "sim") or if the posterior distribution densities should be plotted (plot.type = "dens"). The plots can also done using the method plot.
burnin	is the number of MCMC steps which will be considered as burn-in iterations. If NULL (default value), the 50% first MCMC steps performed are used as burn-in iterations.
...	further arguments passed to or from other methods.
x	an object of class "summary.mcmcSAR" or "mcmcSAR, output of the functions summary.mcmcSAR and print.summary.mcmcSAR.

Details

The function is smart and allows all the possible arguments with the functions summary, plot, par... such as col, lty, mfrow... summary.mcmcSAR, print.summary.mcmcSAR and print.mcmcSAR can be called by summary or print.

Value

A list consisting of:

n.group	number of groups.
N	vector of each group size.
iteration	number of MCMC steps performed.
burnin	number of MCMC steps which will be considered as burn-in iterations.
posterior	matrix (or list of matrices) containing the simulations.
hyperparms	return value of hyperparms.
accept.rate	acceptance rate of zeta.
prop.net	proportion of observed network data.
method.net	network formation model specification.
formula	input value of formula.
alpha	significance level of parameter.
ctrl.mcmc	return value of ctrl.mcmc.
...	arguments passed to methods.

summary.smmSAR *Summarizing SMM Estimation of SAR model*

Description

Summary and print methods for the class smmSAR.

Usage

```
## S3 method for class 'smmSAR'
summary(object, .fun, .args, sim = 30, ncores = 1, dnetwork, data, ...)

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

## S3 method for class 'smmSAR'
print(x, dnetwork, .fun, .args, sim = NULL, ncores = 1, data, ...)
```

Arguments

object	an object of class "smmSAR", output of the function smmSAR.
.fun, .args	are used to simulate from the distribution of dnetwork. .fun is the simulator function where .args is a list of its arguments. Typically do.call(.fun, .args) is supposed to simulate one dnetwork from the distribution.
sim	the number of simulations of dnetwork.
ncores	the number of cores to be used for the simulation. Use a lot of cores for fast simulations.
dnetwork	a list, where the m-th elements is the matrix of link probability in the m-th subnetwork.
data	optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If missing, the variables are taken from environment(formula), typically the environment from which smmSAR is called.
...	further arguments passed to or from other methods.
x	an object of class "summary.smmSAR" or "smmSAR", output of the functions summary.smmSAR or smmSAR.

Value

A list consisting of:

n.group	number of groups.
N	vector of each group size.
estimates	vector of estimated parameters.

`formula`	input value of `formula`.
`contextual`	input value of `contextual`.
`fixed.effects`	input value of `fixed.effects`.
`smm.ctr`	input value of `smm.ctr`.
`details`	other details of the model.

`vec.to.mat` *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

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

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

norm.network(W)
```

Arguments

`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.
`W`	matrix or list of matrices to convert.
`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

sim.network, sim.dnetwork, 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)
```

Index