Package: AsyPeer (via r-universe)

October 30, 2026

Type Package

Title Estimating Asymmetric Peer Effects

Version 0.0.1

Date 2026-08-21

Description Simulating and estimating asymmetric peer-effect models (Houndetoungan and Lambotte, 2026 <doi:10.48550/arXiv.2608.09219>). The model nests the widely used linear-in-means model (Manski, 1993 <doi:10.2307/2298123>; Bramoulle et al., 2009 <doi:10.1016/j.jeconom.2008.12.021>) and allows agents to be influenced differently by friends who exert more or less effort than themselves.

SystemRequirements Requires the OpenMP library for parallel computing. If the OpenMP library is not available, the code is executed sequentially, and a warning is printed.

License GPL-3

Language en-US

Encoding UTF-8

BugReports https://github.com/MathieuLambotte/AsyPeer/issues

URL https://github.com/MathieuLambotte/AsyPeer

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.0), ranger, glmnet, xgboost, formula.tools, doParallel, parallel, foreach, doRNG

LinkingTo Rcpp, RcppEigen, RcppProgress

Suggests PartialNetwork

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation yes

Author Aristide Houndetoungan [aut] (ORCID: <https://orcid.org/0009-0003-2733-9362>), Mathieu Lambotte [cre, aut] (ORCID: <https://orcid.org/0000-0002-7806-7286>)

Maintainer Mathieu Lambotte <mathieu.lambotte@univ-rennes.fr>

Repository https://cran.r-universe.dev

Date/Publication 2026-08-31 13:50:02 UTC

RemoteUrl https://github.com/cran/AsyPeer

RemoteRef HEAD

RemoteSha d286833af873d958b3e42b6db67a5386897e7b83

Contents

AsyPeer-package 2
asypeer.estim 4
asypeer.sim 6
cesconfpeer 9
gen.instrument 13
peer.asyavg 16
plot.spillover 18
spillover 19
summary.asypeer.estim 21
summary.cesconfpeer 22

Index **24**

`AsyPeer-package` *The AsyPeer Package*

Description

The **AsyPeer** package simulates and estimates an asymmetric peer effect model (Houndetoungan and Lambotte, 2026 doi:xxx). This model nests the widely used linear-in-means framework (Manski, 1993 doi:10.2307/2298123; Bramoulle et al., 2009 doi:10.1016/j.jeconom.2008.12.021) and allows agents to be influenced differently by friends who exert more or less effort than themselves. The **AsyPeer** package also estimates the CES-based peer effect model with conformity (Boucher et al., 2024 doi:10.3982/ECTA21048).

Details

The package includes the `asypeer.sim` function, which simulates data from a peer effect model with asymmetric conformity effects (Houndetoungan and Lambotte, 2026 doi:xxx). Specifically, agents may be influenced differently by friends who exert more or less effort than themselves.

The package also provides the `asypeer.estim` function, which estimates the model using the Generalized Method of Moments (GMM). A key challenge in the estimation is that the dependent variable appears on both the left- and right-hand sides of the model specification. Compared to the standard linear-in-means setting, this introduces a second endogenous variable: the average difference between peers' outcomes and one's own outcome among peers who have higher outcomes. As

a result, the dependent variable is defined implicitly.

The package also includes the `gen.instrument` function, used to generate instruments for the two endogenous variables. Estimation using `asypeer.estim` returns an object of class `asypeer.estim`, to which the `summary` and `print` methods can be applied. These methods additionally report important diagnostic tests for the GMM estimator, including the J (Sargan) test and the Kleibergen-Paap Wald test (Kleibergen and Paap, 2006 doi:10.1016/j.jeconom.2005.02.011).
The **AsyPeer** package also estimates the CES-based peer effect model with conformity (Boucher et al., 2024 `doi:10.3982/ECTA21048`), using code adapted from the **QuantilePeer** package (Houndetoungan, 2025 `doi:10.32614/CRAN.package.QuantilePeer`).

To improve computational efficiency, **AsyPeer** leverages `C++` via the **Rcpp** package (Eddelbuettel et al., 2011).

Author(s)

Maintainer: Mathieu Lambotte `<mathieu.lambotte@univ-rennes.fr>` (ORCID)

Authors:

- Aristide Houndetoungan `<ahoundetoungan@ecn.ulaval.ca>` (ORCID)

References

Boucher, V., Rendall, M., Ushchev, P., & Zenou, Y. (2024). Toward a general theory of peer effects. Econometrica, 92(2), 543-565, doi:10.3982/ECTA21048.

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.

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.

Houndetoungan A (2025). QuantilePeer: Quantile Peer Effect Models. doi:10.32614/CRAN.package.QuantilePeer, R package version 0.0.1.

Houndetoungan and Lambotte (2026). Asymmetries in Peer Effects for Continuous Outcomes.

Kleibergen, F., & Paap, R. (2006). Generalized reduced rank tests using the singular value decomposition. *Journal of Econometrics*, 133(1), 97-126, doi:10.1016/j.jeconom.2005.02.011.

Manski, C. F. (1993). Identification of endogenous social effects: The reflection problem. *The Review of Economic Studies*, 60(3), 531-542. doi:10.2307/2298123.

See Also

Useful links:

- `https://github.com/MathieuLambotte/AsyPeer`
- Report bugs at `https://github.com/MathieuLambotte/AsyPeer/issues`

asypeer.estim *Estimate the Asymmetric Peer Effects Model*

Description

asypeer.estim estimates the asymmetric peer effects model. The instruments are generated automatically following the procedure described in the paper and implemented in gen.instruments. The user can also supply her own instruments.

Usage

```
asypeer.estim(
  formula,
  excluded.instruments,
  Glist,
  data,
  common.gamma,
  spillover = FALSE,
  asymmetry = TRUE,
  weight = "IV",
  HAC = "group-iid",
  nboot = 500,
  fixed.effects = FALSE,
  tol = 1e-10,
  nthread = 1,
  drop = NULL,
  tol.optim = .Machine$double.eps^0.25,
  gen.inst.arg = list(),
  ...
)
```

Arguments

formula	An object of class formula: a symbolic description of the model. It should be specified as y ~ x1 + x2 + . . . , where y is the outcome variable and x1, x2, ... are control variables, which may include contextual variables such as peer averages.
excluded.instruments	A formula specifying the excluded instruments. It should be written as ~ z1 + z2 + . . . , where z1, z2, ... are the excluded instruments for the two endogenous variables in the asymmetric model: the average peers' outcomes and the average difference in outcomes between an agent and her higher-performing peers. If omitted, default instruments are generated using gen.instrument.
Glist	The adjacency matrix or list of adjacency matrices. For networks consisting of multiple subnets (e.g., schools), Glist must be a list, where the s-th element is an $n_s x n_s$ adjacency matrix and n_s is the number of nodes in subnet s.

`data`	An optional data frame, list, or environment (or an object that can be coerced to a data frame via as.data.frame) containing the variables in the model. If a variable is not found in `data`, it is retrieved from `environment(formula)`, typically the environment from which `asypeer.estim` is called.
`common.gamma`	A character vector indicating the control variables assumed to have the same marginal effect on marginal utility for isolated and non-isolated agents. If omitted, all control variables are included. A value of `NULL` or `character(0)` indicates that none are included.
`spillover`	A logical value indicating whether the model includes an additional spillover component alongside the asymmetric conformity effect.
`asymmetry`	A logical value indicating whether preferences for conformity are asymmetric.
`weight`	A character string specifying the weighting matrix used in GMM estimation. Available options are `"identity"` for GMM with the identity matrix, `"IV"` for the standard instrumental-variable GMM estimator, and `"optimal"` for GMM with the optimal weighting matrix.
`HAC`	A character string specifying the correlation structure of the idiosyncratic errors used for covariance estimation. Options are `"iid"` for independent errors; `"group iid"` for independence within groups of isolated and non-isolated agents; `"hetero"` for heteroskedastic but non-autocorrelated errors; `"cluster"` for heteroskedastic errors with within-subnetwork correlation; and `"cboot"` for the pair-bootstrap version of `"cluster"`.
`nboot`	The number of bootstrap replications when `HAC = "cluster"`.
`fixed.effects`	A logical value indicating whether subnetwork fixed effects are included in the model.
`tol`	A numeric tolerance used in QR factorization to detect collinear columns in the matrices of explanatory variables and instruments, ensuring a full-rank matrix (see qr).
`nthread`	The number of CPU cores (threads) used for parallel computation.
`drop`	A logical vector of the same length as the sample, indicating which observations should be dropped. This can be used, for example, to remove false isolates or to estimate the model only for non-isolated agents. These observations cannot be removed from the network structure because they may remain connected to other agents.
`tol.optim`	A numeric tolerance used in `optimize`. The objective function is concentrated in a function of β_l or β alone, which is optimized using optimize.
`gen.inst.arg`	A list of arguments for the `gen.instrument` function, used when instruments are to be generated. These arguments exclude `formula`, `Glist`, `data`, `asymmetry`, `fold.nthread`, `drop`, and `tol`, which are taken directly from the `asypeer.estim` function.
`...`	Further arguments passed to or from other methods.

Value

A list containing:

model.info	A list with information about the model, such as the number of subnets, number of observations, and other key details.
gmm	A list of GMM estimation results, including parameter estimates, the covariance matrix, and related statistics.
data	A list including the original data used to estimate the model, as well as the endogenous variables and their instruments.

Examples

```
if (requireNamespace("PartialNetwork", quietly = TRUE)) {
library(PartialNetwork)
ngr  <- 50  # Number of subnets
nvec <- rep(30, ngr)  # Size of subnets
n    <- sum(nvec)

### Simulating Data
## Network matrix
G   <- lapply(1:ngr, function(z) {
 Gz <- matrix(rbinom(nvec[z]^2, 1, 0.3), nvec[z], nvec[z])
 diag(Gz) <- 0
 # Adding isolated nodes (important for the structural model)
 niso <- sample(0:nvec[z], 1, prob = ((nvec[z] + 1):1)^5 / sum(((nvec[z] + 1):1)^5))
 if (niso > 0) {
   Gz[sample(1:nvec[z], niso), ] <- 0
 }
 Gz
})

Gnorm   <- norm.network(G)
X       <- cbind(rnorm(n, 0, 2), rpois(n, 2))
GX      <- peer.avg(Gnorm, X)
delta   <- 0.25
beta    <- c(0.3, 0.6)
gamma   <- c(4, 1, -0.7, 0, -0.5)
eps     <- rnorm(n, 0, 0.5)

## Generating `y`
y <- asypeer.sim(formula = ~ X + GX, Glist = Gnorm, delta = delta, beta = beta,
                 gamma = gamma, epsilon = eps)
y <- y$y

### Estimating the asymmetric peer effects model
est <- asypeer.estim(formula=y ~ X + GX, Glist = Gnorm, spillover = TRUE)
summary(est, diagnostic = TRUE)
}
```

Description

`asypeer.sim` simulates asymmetric peer effect models with continuous responses.

Usage

```
asypeer.sim(
  formula,
  Glist,
  parms,
  beta,
  delta,
  gamma,
  epsilon,
  init,
  tol = 1e-10,
  maxit = 500,
  nthread = 1,
  print = FALSE,
  data
)
```

Arguments

`formula`	A formula object (formula): a symbolic description of the model. `formula` should be specified as, for example, `~ x1 + x2`, where `x1` and `x2` are control variables, which can include contextual variables such as averages or quantiles among peers.
`Glist`	The adjacency matrix. For networks consisting of multiple subnets (e.g., schools), `Glist` must be a list of subnets, with the `s`-th element being an $n_s \times n_s$ adjacency matrix, where n_s is the number of nodes in the `s`-th subnet.
`parms`	A vector defining the true values of $(\boldsymbol{\beta}', \delta, \boldsymbol{\gamma}')'$, where $\boldsymbol{\beta} = (\beta^l, \beta^h)'$ captures asymmetric conformity effects, δ measures spillover effects, and $\boldsymbol{\gamma}$ is the parameter vector associated with the exogenous characteristics in agent types (i.e., the control variables `x1`, `x2`, ect.). The parameters δ, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ can also be specified separately using the arguments `delta`, `beta`, and `gamma`.
`beta`	The true value of the asymmetric conformity parameters.
`delta`	The true value of the spillover effect parameter.
`gamma`	The true value of the vector $\boldsymbol{\gamma}$.
`epsilon`	A vector of idiosyncratic error terms. If not specified, it will be simulated from a standard normal distribution.
`init`	An optional initial guess for the equilibrium.
`tol`	The tolerance value used in the Fixed Point Iteration Method to compute the outcome `y`. The process stops if the ℓ_1-distance between two consecutive values of `y` is less than `tol`.
`maxit`	The maximum number of iterations for the Fixed Point Iteration Method.

nthread	Number of CPU cores (threads) used to run parts of the simulation in parallel.
print	A logical value indicating whether the ℓ_1 distance at each iteration of the best-response dynamics should be printed.
data	An optional data frame, list, or environment containing the model variables. If a variable is not found in data, it is retrieved from environment(formula), typically the environment from which asypeer.sim is called.

Value

A list containing:

y	The simulated variable.
epsilon	The idiosyncratic error.
init	The initial guess.
iteration	The number of iterations before convergence.

Examples

```
if (requireNamespace("PartialNetwork", quietly = TRUE)) {
library(PartialNetwork)
ngr  <- 50  # Number of subnets
nvec <- rep(30, ngr)  # Size of subnets
n    <- sum(nvec)

### Simulating Data
## Network matrix
G <- lapply(1:ngr, function(z) {
 Gz       <- matrix(rbinom(nvec[z]^2, 1, 0.3), nvec[z], nvec[z])
 diag(Gz) <- 0
 # Adding isolated nodes (important for the structural model)
 niso <- sample(0:nvec[z], 1, prob = ((nvec[z] + 1):1)^5 / sum(((nvec[z] + 1):1)^5))
 if (niso > 0) {
   Gz[sample(1:nvec[z], niso), ] <- 0
 }
 Gz
})

Gnorm   <- norm.network(G)
X       <- cbind(rnorm(n, 0, 2), rpois(n, 2))
GX      <- peer.avg(Gnorm, X)
delta   <- 0.25
beta    <- c(0.3, 0.6)
gamma   <- c(4, 1, -0.7, 0, -0.5)
eps     <- rnorm(n, 0, 0.5)

#' ## Generating `y`
y <- asypeer.sim(formula = ~ X + GX, Glist = Gnorm, delta = delta, beta = beta,
                 gamma = gamma, epsilon = eps)}
```

cesconfpeer *Estimation of CES-Based Peer Effects Models With Conformity*

Description

cesconfpeer estimates the CES-based peer effects model introduced by Boucher et al. (2024) with conformity only. See Details.

Usage

```
cesconfpeer(
  formula,
  instrument,
  Glist,
  fixed.effects = FALSE,
  set.rho = NULL,
  interval = c(-100, 100),
  tol = 1e-08,
  drop = NULL,
  HAC = "group-iid",
  nthread = 1,
  data,
  n.rho0 = 10,
  n.beta0 = 5,
  weight.rho = NULL,
  weight.optim = TRUE,
  arg.optim = list(),
  ...
)

plotcespeer(
  formula,
  instrument,
  Glist,
  fixed.effects = FALSE,
  grid.rho = seq(-50, 50, 0.1),
  tol = 1e-08,
  drop = NULL,
  nthread = 1,
  data,
  weight.rho = NULL,
  arg.optim = list(),
  ...
)
```

Arguments

`formula`	An object of class formula: a symbolic description of the model. `formula` should be specified as `y ~ x1 + x2`, where `y` is the outcome and `x1` and `x2` are control variables, which can include contextual variables such as averages or quantiles among peers.
`instrument`	An object of class formula indicating the excluded instrument. It should be specified as `~ z`, where `z` is the excluded instrument for the outcome. Following Boucher et al. (2024), it can be an OLS exogenous prediction of `y`. This prediction is used to compute instruments for the CES function of peer outcomes.
`Glist`	The adjacency matrix. For networks consisting of multiple subnets (e.g., schools), `Glist` must be a list of subnets, with the `m`-th element being an $n_m \times n_m$ adjacency matrix, where n_m is the number of nodes in the `m`-th subnet.
`fixed.effects`	A logical value or string specifying whether the model includes subnet fixed effects.
`set.rho`	A fixed value for the CES substitution parameter to estimate a constrained model.
`interval`	A numerical vector of two elements indicating support for ρ.
`tol`	A tolerance value used in the QR factorization to identify columns of explanatory variable and instrument matrices that ensure a full-rank matrix (see the qr function). The same tolerance is also used in the to minimize the concentrated GMM objective function (see optimise).
`drop`	A dummy vector of the same length as the sample, indicating whether an observation should be dropped. This can be used, for example, to remove false isolates or to estimate the model only on non-isolated agents. These observations cannot be directly removed from the network by the user because they may still be friends with other agents.
`HAC`	A character string specifying the correlation structure of the idiosyncratic errors for covariance computation. Options are `"iid"` for independent errors; `"group iid"` for independence within the groups of isolated and non-isolated players; `"hetero"` for heteroskedastic but non-autocorrelated errors; and `"cluster"` for heteroskedastic errors with potential within-subnetwork correlation.
`nthread`	Number of CPU cores (threads) used to run parts of the estimation in parallel.
`data`	An optional data frame, list, or environment (or an object that can be coerced 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 `cesconfpeer` is called.
`n.rho0`, `n.beta0`	Integer values indicating the number of starting points to test for ρ and β, respectively. Because the objective function may exhibit local optima, testing multiple starting values is useful, especially for ρ. On a reasonable parameter set, testing ten starting values often suffices.
`weight.rho`	The ρ value used to compute the weighting matrix for the first GMM estimation or to plot the objective function. The weight is computed as proportional to $(Z'Z)^{-1}$, where Z is the matrix of instruments. If `weight.rho` is omitted, the identity matrix is used.
`weight.optim`	A Boolean indicating whether the GMM with the optimal weighting matrix should be computed.

`arg.optim` A list of additional arguments passed to optim, such as `method` and `control`, for models with a flexible ρ, or to optimize, such as `tol`, for models with a fixed (or grid) value of ρ. For models with a flexible ρ, the objective function is concentrated in a function of ρ and β, which is optimized using optim. For models with a fixed ρ, the objective function is concentrated in a function of β alone, which is optimized using optimize.

`...` Further arguments passed to or from other methods.

`grid.rho` A finite grid of values for the CES substitution parameter ρ (see Details). This grid is used to plot the objective function after maximizing over the other parameters. It is helpful for determining a reasonable interval for ρ.

Details

Let $\mathcal{N}$ denote a set of n agents indexed by $i \in [1, n]$. Agents are connected through a network represented by an adjacency matrix $\mathbf{G} = [g_{ij}]$ of dimension $n \times n$, where $g_{ij} = 1$ if agent j is a friend of agent i, and $g_{ij} = 0$ otherwise. In weighted networks, g_{ij} can be a nonnegative value (not necessarily binary) that measures the intensity of the outgoing link from i to j. The model accommodates such networks. Note that the network may consist of multiple independent subnets (e.g., schools). The `Glist` argument is the list of subnets. For a single subnet, `Glist` should be a list containing one matrix.

The specification of the CES-based peer effects model differs for isolated and non-isolated individuals. For an **isolated** agent i, the specification is similar to a standard linear-in-means model without social interactions:

$$y_i = \mathbf{x}_i'\gamma + \varepsilon_i,$$

where ε_i is an idiosyncratic error term and γ captures the effect of $\mathbf{x}_i$ on y_i.

If agent i is **non-isolated**, the specification is:

$$y_i = \frac{\beta}{1+\beta}\left(\sum_{j=1}^{n} g_{ij} y_j^{\rho}\right)^{1/\rho} + (1-\lambda_2)\mathbf{x}_i'\frac{\gamma}{1+\beta} + \varepsilon_i,$$

where γ captures conformity effects—how agent i tends to conform to friends through the social norm $\left(\sum_{j=1}^{n} g_{ij} y_j^{\rho}\right)^{1/\rho}$.

The parameter ρ determines the form of the social norm:

- When $\rho > 1$, individuals are more sensitive to peers with high outcomes.
- When $\rho < 1$, individuals are more sensitive to peers with low outcomes.
- When $\rho = 1$, peer effects are uniform across peer outcome values.

Value

A list containing:

`model.info`	A list with information about the model, including the number of subnets, the number of observations, and other key details.
`gmm`	A list of GMM estimation results, including parameter estimates, the covariance matrix, and related statistics.
`first.gmm`	Initial GMM estimation results using the identity matrix as the weighting matrix.

Examples

```
if (requireNamespace("PartialNetwork", quietly = TRUE)) {
library(PartialNetwork)
set.seed(123)
ngr  <- 30  # Number of subnets
nvec <- rep(30, ngr)  # Size of subnets
n    <- sum(nvec)

### Simulating Data
## Network matrix
G   <- lapply(1:ngr, function(z) {
  Gz <- matrix(rbinom(nvec[z]^2, 1, 0.3), nvec[z], nvec[z])
  diag(Gz) <- 0
  # Adding isolated nodes
  niso <- sample(0:nvec[z], 1, prob = ((nvec[z] + 1):1)^5 / sum(((nvec[z] + 1):1)^5))
  if (niso > 0) {
    Gz[sample(1:nvec[z], niso), ] <- 0
  }
  Gz
})

Gnorm   <- norm.network(G)
X       <- cbind(rnorm(n, 0, 2), rpois(n, 2))
delta   <- 0
beta    <- c(1, 1)
gamma   <- c(6, 1, 0.7)
eps     <- rnorm(n, 0, 0.5)

## Generating `y`
y <- asypeer.sim(formula = ~ X, Glist = Gnorm, delta = delta, beta = beta,
                 gamma = gamma, epsilon = eps)
y <- y$y

### Estimating the asymmetric peer effects model
z   <- predict(lm(y ~ X))
plotcespeer(formula = y ~ X, instrument = ~ z, Glist = Gnorm,
            grid.rho = seq(-20, 20, 1))
est <- cesconfpeer(formula = y ~ X, instrument = ~ z, Glist = Gnorm,
                   n.rho0 = 2, n.beta0 = 1)
summary(est)
}
```

gen.instrument *Generate Instruments for the Asymmetric Peer Effects Model*

Description

gen.instrument generates instruments for the endogenous variables in asymmetric peer effects models.

Usage

```
gen.instrument(
  formula,
  Glist,
  data,
  asymmetry = TRUE,
  estimator = "ols",
  power = c(1, 1),
  full = FALSE,
  nfold = 2,
  checkrank = TRUE,
  tol = 1e-10,
  fold.nthread = 1,
  drop = NULL,
  ...
)

gen.instruments(
  formula,
  Glist,
  data,
  asymmetry = TRUE,
  estimator = c("ols", "logit"),
  power = c(1, 1),
  full = FALSE,
  nfold = 2,
  checkrank = TRUE,
  tol = 1e-10,
  fold.nthread = 1,
  drop = NULL,
  ...
)

gen.insts(
  formula,
  Glist,
  data,
  asymmetry = TRUE,
```

```
  estimator = c("ols", "logit"),
  power = c(1, 1),
  full = FALSE,
  nfold = 2,
  checkrank = TRUE,
  tol = 1e-10,
  fold.nthread = 1,
  drop = NULL,
  ...
)

gen.inst(
  formula,
  Glist,
  data,
  asymmetry = TRUE,
  estimator = c("ols", "logit"),
  power = c(1, 1),
  full = FALSE,
  nfold = 2,
  checkrank = TRUE,
  tol = 1e-10,
  fold.nthread = 1,
  drop = NULL,
  ...
)
```

Arguments

`formula`	An object of class formula: a symbolic description of the model. The formula should be specified as `y ~ x1 + x2`, where `y` is the outcome variable and `x1`, `x2`, ... are control variables, which may include contextual variables such as peer averages.
`Glist`	The adjacency matrix or a list of adjacency matrices. For networks composed of multiple subnets (e.g., schools), `Glist` must be a list, where the `s`-th element is an $n_s \times n_s$ adjacency matrix, and n_s is the number of nodes in subnet `s`.
`data`	An optional data frame, list, or environment (or an object that can be coerced to a data frame via as.data.frame) containing the variables in the model. If a variable is not found in `data`, it is searched for in `environment(formula)`, typically the environment from which `gen.instrument` is called.
`asymmetry`	A logical value indicating whether preferences for conformity are asymmetric.
`estimator`	A character vector of length 2 specifying the estimators used for the intensive margin $E(y_j - y_i \mid y_j - y_i > 0)$ (or $E(\bar{y}_i)$) and the extensive margin $P(y_j > y_i)$. Supported parametric options include `"ols"` and `"lasso"` for linear models, and `"glm"` for generalized linear models (only for the extensive margin). Nonparametric options include `"rf"` (random forest) and `"xgboost"` (gradient boosting). For example, `estimator = c("xgboost", "glm")` uses gradient boosting for the intensive margin and a generalized linear model for the extensive margin.

`power` A numeric vector of length 2 indicating the maximum walk lengths (typically k in $G^k X$) used to construct or estimate instruments for $\bar{y}_i$ and $\check{y}_i$. The two entries allow different values of k for each endogenous variable.

`full` A logical value. If `TRUE`, predictions for both $\bar{y}_i$ and $\check{y}_i$ are used as instruments. If `FALSE`, only the prediction for $\check{y}_i$ is used as an instrument, while $\bar{y}_i$ is instrumented using the usual instruments, $G^k X$.

`nfold` A strictly positive integer specifying the number of folds used for cross-fitting in the estimation of the probability model.

`checkrank` A logical value indicating whether linearly dependent columns in the matrix of generated instruments should be dropped.

`tol` A numeric tolerance used in QR factorization to detect linearly dependent columns in the matrices of explanatory variables and instruments, ensuring a full-rank matrix (see qr).

`fold.nthread` Number of CPU cores (threads) used to run parts of the estimation in parallel. This refers to the number of threads used for estimations across different folds. Each estimation method can define its own number of threads, which can be passed through the `...` argument. For example, for a random forest estimation (`estimation = "rf"`), the `num.threads` argument can be specified as described in the ranger function documentation.

`drop` A logical vector of the same length as the sample, indicating which observations should be dropped. This can be used, for example, to remove false isolates or to estimate the model only on non-isolated agents. These observations cannot be physically removed from the network structure because they may still be connected to other agents.

`...` Further arguments passed to or from other methods. These correspond to arguments of the functions used for each estimation method. For `"ols"`, see the arguments of the lm function; for `"logit"`, see the arguments of the glm function; for `"lasso"`, see the arguments of the cv.glmnet function; for `"xgboost"`, see the arguments of the xgb.params and xgb.train functions; and for `"rf"`, see the arguments of the ranger function.

Value

A list containing:

`model.info` A list with information about the model, such as the estimator, the number of folds, and other key details.

`instruments` A matrix of generated instruments.

Examples

```
if (requireNamespace("PartialNetwork", quietly = TRUE)) {
library(PartialNetwork)
ngr  <- 50  # Number of subnets
nvec <- rep(30, ngr)  # Size of subnets
n    <- sum(nvec)
```

```
### Simulating Data
## Network matrix
G <- lapply(1:ngr, function(z) {
  Gz       <- matrix(rbinom(nvec[z]^2, 1, 0.3), nvec[z], nvec[z])
  diag(Gz) <- 0
  # Adding isolated nodes (important for the structural model)
  niso <- sample(0:nvec[z], 1, prob = ((nvec[z] + 1):1)^5 / sum(((nvec[z] + 1):1)^5))
  if (niso > 0) {
    Gz[sample(1:nvec[z], niso), ] <- 0
  }
  Gz
})
Gnorm   <- norm.network(G)
X       <- cbind(rnorm(n, 0, 2), rpois(n, 2))
GX      <- peer.avg(Gnorm, X)
delta   <- 0.25
beta    <- c(0.3, 0.6)
gamma   <- c(4, 1, -0.7, 0, -0.5)
eps     <- rnorm(n, 0, 0.5)

## Generating `y`
y <- asypeer.sim(formula = ~ X + GX, Glist = Gnorm, delta = delta,
                 beta = beta, gamma = gamma, epsilon = eps)
y <- y$y

### Generating instruments
ins <- gen.instrument(formula = y ~ X, Glist = Gnorm,
                      estimator = c("ols", "logit"))}
```

`peer.asyavg` *Computing peer (asymmetric) average values*

Description

`peer.asyavg` computes the average values of a vector or matrix `X` among friends, differentiating between friends with a higher or lower outcome than the agent.

Usage

```
peer.asyavg(formula, Glist, data, nthread = 1)
```

Arguments

`formula`	An object of class formula, which should be specified as `y ~ X`, where `y` is the dependent variable used to characterized which friends have a higher or lower outcome and `X` is the vector or matrix of variables whose average values among peers should be computed.

`Glist`	The adjacency matrix. For networks consisting of multiple subnets (e.g., schools), `Glist` must be a list of subnets, with the `m`-th element being an $n_m \times n_m$ adjacency matrix, where n_m is the number of nodes in the `m`-th subnet.
`data`	An optional data frame, list, or environment (or an object that can be coerced to a data frame via as.data.frame) containing the variables in the model. If a variable is not found in `data`, it is taken from `environment(formula)`, typically the environment from which `peer.asyavg` is called.
`nthread`	A strictly positive integer specifying the number of threads used in computationally intensive steps.

Value

A list containing:

`degree`	The row sum of the network matrix. As `Glist` is row-normalized, it is either `0` or `1`.
`peer.avg`	The average of `X` (see the `formula` argument) among all peers.
`hdegree`	The share of higher-performing friends.
`hpeer.avg`	The sum of `X` weighted by g_{ij} among higher-performing friends.
`ldegree`	The share of lower-performing friends.
`lpeer.avg`	The sum of `X` weighted by g_{ij} among lower-performing friends.

Examples

```
if (requireNamespace("PartialNetwork", quietly = TRUE)) {
library(PartialNetwork)
ngr  <- 50  # Number of subnets
nvec <- rep(30, ngr)  # Size of subnets
n    <- sum(nvec)

### Simulating Data
## Network matrix
G <- lapply(1:ngr, function(z) {
  Gz       <- matrix(rbinom(nvec[z]^2, 1, 0.3), nvec[z], nvec[z])
  diag(Gz) <- 0
  # Adding isolated nodes (important for the structural model)
  niso <- sample(0:nvec[z], 1, prob = ((nvec[z] + 1):1)^5 / sum(((nvec[z] + 1):1)^5))
  if (niso > 0) {
    Gz[sample(1:nvec[z], niso), ] <- 0
  }
  Gz
})
Gnorm   <- norm.network(G)
X       <- cbind(rnorm(n, 0, 2), rpois(n, 2))
GX      <- peer.avg(Gnorm, X)
colnames(X)<-c("X1","X2")
delta   <- 0.25
beta    <- c(0.3, 0.6)
gamma   <- c(4, 1, -0.7, 0, -0.5)
```

```
eps     <- rnorm(n, 0, 0.5)

## Generating `y`
y <- asypeer.sim(formula = ~ X + GX, Glist = Gnorm, delta = delta, beta = beta,
                 gamma = gamma, epsilon = eps, nthread = 5)
y <- y$y

### Computing averages X among peers
peeravg <- peer.asyavg(formula = y ~ X, Glist = Gnorm, nthread = 1)
}
```

`plot.spillover` *Plot the Effects of a Targeted Intervention*

Description

`plot.spillover` generates plots illustrating spillovers under different intervention budgets and targeting methods.

Usage

```
## S3 method for class 'spillover'
plot(x, metric = c("spillover", "outcome"), range, ...)
```

Arguments

`x`	An object of class `spillover`, as returned by spillover.
`metric`	A character string specifying which metrics should be plotted. Available options are `"spillover"`, `"gain"`, `"outcome"`, or any combination of these.
`range`	A vector of integer indices indicating the ranks to be considered.
`...`	Further arguments passed to or from other methods.

Value

This function is called for its side effect of producing a plot. It does not return a value.

spillover *Find the Optimal Targeting Order in an Intervention*

Description

spillover identifies the optimal targeting order in an intervention aimed at maximizing utilitarian welfare and computes the associated individual spillovers. The targeting order and spillovers can be generated using different targeting methods: the symmetric specification or the asymmetric specification. In the latter case, the optimal order is computed through forward or backward optimization.

Usage

```
spillover(
  asymodel,
  symodel,
  Glist,
  targ.net,
  data,
  treatment = 1,
  nthread = 1,
  print = TRUE,
  tol = 1e-09
)
```

Arguments

asymodel	An object of class asypeer.estim or summary.asypeer.estim, estimated using the asymmetric specification (asymmetry = TRUE).
symodel	An object of class asypeer.estim or summary.asypeer.estim, estimated using the symmetric specification (asymmetry = FALSE).
Glist	An adjacency matrix or a list of adjacency matrices. For networks consisting of multiple subnets (e.g., schools), Glist must be a list, where the s-th element is an $n_s \times n_s$ adjacency matrix and n_s is the number of nodes in subnet s.
targ.net	A scalar indicating the index of the network in Glist for which the targeting order and spillovers are to be computed.
data	An optional data frame, list, environment, or any object that can be coerced to a data frame via as.data.frame, containing the variables used in the model. If a variable is not found in data, it is retrieved from the environment of asymodel.
treatment	A scalar indicating the treatment value.
nthread	The number of CPU cores (threads) used for parallel computation.
print	A logical value indicating whether a progress bar should be displayed.
tol	A numeric tolerance used to assess convergence to a post-intervention Nash equilibrium.

Value

A list containing:

`targeted`	A matrix or list containing the indices of the targeted individuals, ordered by their influence.
`budget`	A vector indicating the intervention budget (the absolute value of the total treatment) for each number of targeted individuals.
`total.treat`	A vector indicating the total treatment applied for each number of targeted individuals.
`diff.y`	A matrix containing the difference between baseline welfare and post-intervention welfare for each number of targeted individuals.
`spillover`	A matrix containing the spillover associated with each targeted individual.

Examples

```
if (requireNamespace("PartialNetwork", quietly = TRUE)) {
  library(PartialNetwork)
  ngr  <- 50  # Number of subnets
  nvec <- rep(30, ngr)  # Size of subnets
  n    <- sum(nvec)

  ### Simulating Data
  ## Network matrix
  G   <- lapply(1:ngr, function(z) {
    Gz <- matrix(rbinom(nvec[z]^2, 1, 0.3), nvec[z], nvec[z])
    diag(Gz) <- 0
    # Adding isolated nodes (important for the structural model)
    niso <- sample(0:nvec[z], 1, prob = ((nvec[z] + 1):1)^5 / sum(((nvec[z] + 1):1)^5))
    if (niso > 0) {
      Gz[sample(1:nvec[z], niso), ] <- 0
    }
    Gz
  })

  Gnorm   <- norm.network(G)
  X       <- cbind(rnorm(n, 0, 2), rpois(n, 2))
  GX      <- peer.avg(Gnorm, X)
  delta   <- 0.25
  beta    <- c(0.3, 1.6)
  gamma   <- c(4, 1, -0.7, 0, -0.5)
  eps     <- rnorm(n, 0, 0.5)

  ## Generating `y`
  y <- asypeer.sim(formula = ~ X + GX, Glist = Gnorm, delta = delta, beta = beta,
                   gamma = gamma, epsilon = eps)
  y <- y$y

  ### Estimating a symmetric peer effects model
  mod1 <- asypeer.estim(formula = y ~ X + GX, Glist = Gnorm, spillover = TRUE,
                        asymmetry = FALSE)
```

```
  summary(mod1)

  ### Estimating an asymmetric peer effects model
  mod2 <- asypeer.estim(formula = y ~ X + GX, Glist = Gnorm, spillover = TRUE)
  summary(mod2)

  ## treatment
  treat <- spillover(asymodel = mod2, symodel = mod1, Glist = Gnorm, targ.net = 1,
                     treatment = 1)

  ## Plot
  plot(treat)
}
```

---

summary.asypeer.estim *Summary and Print Methods for the Asymmetric Peer Effects Model*

Description

Summary and print methods for objects of class asypeer.estim.

Usage

```
## S3 method for class 'asypeer.estim'
summary(
  object,
  diagnostic = FALSE,
  diagnostics = FALSE,
  SW = FALSE,
  KP = diagnostics || diagnostic,
  nthread = 1L,
  ...
)

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

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

Arguments

object	An object of class asypeer.estim as returned by the function asypeer.estim.
diagnostics, diagnostic	A logical value indicating whether diagnostic tests for the IV GMM should be performed. These include an F-test of the first-stage regression for weak instruments, a Wu-Hausman test for endogeneity, and a Hansen's J-test for overidentifying restrictions (the latter only when the number of instruments exceeds the number of regressors).

SW	Logical value indicating whether the Sanderson–Windmeijer conditional F-statistic is computed instead of the classical first-stage F-statistic.
KP	A logical value indicating whether a Kleibergen-Paap Wald test should be performed in addition to the standard F test of the first-stage regression for weak instruments. should be performed instead of the standard F test.
nthread	A strictly positive integer specifying the number of threads used in computationally intensive steps of the estimation procedure.
...	Further arguments passed to or from other methods.
x	An object of class summary.asypeer.estim or asypeer.estim as returned by the function summary.asypeer.estim or asypeer.estim, respectively.

Value

A list containing:

model.info	A list containing information about the model, such as the number of subnets, number of observations, and other key details.
coefficients	A summary of coefficient estimates, standard errors, and p-values.
diagnostics	A summary of the diagnostic tests for the instrumental-variable regression, if requested.
gmm	A list of GMM estimation results, including parameter estimates, the covariance matrix, and related statistics.

---

summary.cesconfpeer *Summary for the Estimation of CES-based Peer Effects Models*

Description

Summary and print methods for the class cesconfpeer.

Usage

```
## S3 method for class 'cesconfpeer'
summary(object, fullparameters = TRUE, ...)

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

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

Arguments

object	An object of class cesconfpeer.
fullparameters	A logical value indicating whether all parameters should be summarized (may be useful for the structural model).
...	Further arguments passed to or from other methods.
x	An object of class summary.cesconfpeer or cesconfpeer.

Value

A list containing:

`model.info`	A list with information about the model, such as the number of subnets, number of observations, and other key details.
`coefficients`	A summary of the estimates, standard errors, and p-values.
`gmm`	A list of GMM estimation results, including parameter estimates, the covariance matrix, and related statistics.

Index

as.data.frame, *5*, *10*, *14*, *17*, *19*
AsyPeer *(*AsyPeer-package*)*, 2
AsyPeer-package, 2
asypeer.estim, *2, 3*, 4, *19*, *21, 22*
asypeer.sim, *2*, 6

cesconfpeer, 9, *22*
cv.glmnet, *15*

formula, *4*, *7*, *10*, *14*, *16*

gen.inst *(*gen.instrument*)*, 13
gen.instrument, *3–5*, 13
gen.instruments, *4*
gen.instruments *(*gen.instrument*)*, 13
gen.insts *(*gen.instrument*)*, 13
glm, *15*

lm, *15*

optim, *11*
optimise, *10*
optimize, *5*, *11*

peer.asyavg, 16
plot.spillover, 18
plotcespeer *(*cesconfpeer*)*, 9
print, *3*
print.asypeer.estim
        *(*summary.asypeer.estim*)*, 21
print.cesconfpeer
        *(*summary.cesconfpeer*)*, 22
print.summary.asypeer.estim
        *(*summary.asypeer.estim*)*, 21
print.summary.cesconfpeer
        *(*summary.cesconfpeer*)*, 22

qr, *5*, *10*, *15*

ranger, *15*

spillover, *18*, 19
summary, *3*
summary.asypeer.estim, *19*, 21, *22*
summary.cesconfpeer, 22, *22*

xgb.params, *15*
xgb.train, *15*