# Package ‘netdep’

July 10, 2018

**Type** Package

**Title** Testing for Network Dependence

**Version** 0.1.0

**Maintainer** Youjin Lee <ylee160@jhu.edu>

**Imports** stats, igraph, igraphdata, MASS, mvrtn

**Suggests** knitr, testthat

**Description** When network dependence is present, that is when social relations can engender dependence in the outcome of interest, treating such observations as independent results in invalid, anti-conservative statistical inference. We propose a test of independence among observations sampled from a single network <arXiv:1710.03296>.

**License** GPL (>= 3) | file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Youjin Lee [aut, cre],
Elizabeth Ogburn [aut]

**Repository** CRAN

**Date/Publication** 2018-07-10 16:00:07 UTC

## R topics documented:

---

`latent.netdep` *Generate latent variable dependent network.*

---

## Description

Generate latent variable dependent network.

## Usage

```
latent.netdep(n.node, rho = 0.3, dep.factor = 1, dep.range = c(-5, 5))
```

## Arguments

| | |
|---|---|
| `n.node` | The number of nodes in network. |
| `rho` | correlation coefficient between continuous observations and latent factor . |
| `dep.factor` | multiplicative factor applied to.<br>• If `dep.factor` < 0 Then $A_ij$ Bern (logistic ( `dep.factor`*\| X_i - X_j \|))<br>• If `dep.factor` $\geq$ 0Then $A_ij$ Bern (logistic ( `dep.factor` / \| X_i - X_j \|)) |
| `dep.range` | a vector specifying lower bound and upper bound for `dep.factor`*\| X_i - X_j \|. Defaults to `c(-5, 5)`. |

## Value

an undirected and binary `igraph` object `G` having both $Y$ and $U$ as nodal attributes.

| | |
|---|---|
| `V(G)$outcome` | one-dimensional continuous observations. |
| `V(G)$latent` | one-dimensional continuous latent variable dependent on `V(G)$Y` through `rho`. |

## Examples

```
library(netdep)
library(MASS)
library(mvrtn)
library(igraph)
G = latent.netdep(n.node = 100, rho = 0.5, dep.factor = 1)
```

## make.permute.moran — *Permutation Test of Moran's I*

### Description

Permutation Test of Moran's I

### Usage

```
make.permute.moran(A, Y, np = 100)
```

### Arguments

| | |
|---|---|
| A | `[n x n]` adjacency matrix or general relational weight matrix of which $A_i j$ indicates relationship from $i$ to $j$. |
| Y | a vector of $n$ continuous or binary, one-dimensional observations. |
| np | the number of permutation samples. |

### Value

| | |
|---|---|
| moran | a standardized Moran's $I$ statistic. |
| pval.z | p-value of a standardized Moran's $I$ statistic assuming asymptotic normality. |
| pval.permute | p-value of a standardized Moran's $I$ statistic using `np` independent permutation samples. |

### Author(s)

Youjin Lee

### Examples

```
library(netdep)
library(igraph)
library(igraphdata)
data(karate)
A = as.matrix(get.adjacency(karate, attr= "weight", sparse = TRUE)) # weighted adjacency matrix
Y = V(karate)$Faction
result = make.permute.moran(A, Y, np = 100)
```

make.permute.Phi — *Permutation Test of* $\Phi$

## Description

This function prints out the network dependence test results for categorical observations.

## Usage

```
make.permute.Phi(A, Y, np)
```

## Arguments

| | |
|---|---|
| A | `[n x n]` adjacency matrix or general relational weight matrix of which $A_ij$ indicates relationship from $i$ to $j$. |
| Y | a vector of $n$ continuous or binary, one-dimensional observations. |
| np | the number of permutation samples. |

## Value

| | |
|---|---|
| phi | a standardized $\Phi$ statistic. |
| pval.z | p-value of a standardized $\Phi$ statistic assuming asymptotic normality. |
| pval.permute | p-value of a standardized $\Phi$ statistic using `np` independent permutation samples. |

## Author(s)

Youjin Lee

## Examples

```
library(netdep)
library(igraph)
library(igraphdata)
data(UKfaculty)
A = as.matrix(get.adjacency(UKfaculty, attr= "weight", sparse = TRUE)) # weighted adjacency matrix
Y = V(UKfaculty)$Group
result = make.permute.Phi(A, Y, np = 50)
```

`MoranI` *Moran's I statistic*

## Description

This function calculates Moran's I statistic given adjacency matrix and outcome. The statistic can be used for testing network dependence.

## Usage

```
MoranI(A, Y)
```

## Arguments

| | |
|---|---|
| A | `[n x n]` adjacency matrix or general relational weight matrix of which $A_ij$ indicates relationship from $i$ to $j$. |
| Y | a vector of $n$ continuous or binary, one-dimensional observations. |

## Value

| | |
|---|---|
| moran | a standardized Moran's $I$ statistic. |

## Author(s)

Youjin Lee

`peer.process` *Generate direct transmission process*

## Description

Generate time-evolving outcomes where outcomes at time $t$ of $i$ depends on outcomes of $i$'s adjacent peers at time $t-1$.

## Usage

```
peer.process(A, max.time = 3, mprob = 0.5, epsilon = 0.3)
```

### Arguments

- `A` — `[n x n]` adjacency matrix.
- `max.time` — the maximum discrete time that direct transmission occurs.
- `mprob` — the maximum susceptibility probability, i.e. maximum probability that $i$'s outcome at time $t$ depends on $i$'s peers at time $t-1$.
- `epsilon` — standard deviation of error process. This adds uncertainties in outcomes.

  For t=1,2, ... `max.time` :

$$p\, Unif(0, mprob)$$

$$Y_i^t = Y_i^t = (1-p)Y_i^{t-1} + p\sum jA_{ij}Y_j^{t-1} / \sum_j A_{ij} + N(0,\epsilon)$$

### Value

a list of time-evolving outcomes from `time0` to `time(max.time)`.

### Examples

```
library(netdep)
library(igraph)
library(stats)
G = latent.netdep(n.node = 100, rho = 0.2)
A = as.matrix(get.adjacency(G))
outcomes = peer.process(A, max.time = 3, mprob = 0.3, epsilon = 0.5)
```

---

## phi.moment     *Calculate Φ statistic*

---

### Description

This is an auxiliary function to calculate non-standardized Φ statistic and its first and second moment.

### Usage

```
phi.moment(A, Y)
```

### Arguments

- `A` — `[n x n]` adjacency matrix or general relational weight matrix of which $A_ij$ indicates relationship from $i$ to $j$.
- `Y` — a vector of $n$ nominal or binary, one-dimensional observations.

## Value

| | |
|---|---|
| `rawphi` | Non-standardized $\Phi$ statistic. |
| `m1.rawphi` | the first (permutation) moment of `rawphi`. |
| `m2.rawphi` | the second (permutation) moment of `rawphi`. |

## Author(s)

Youjin Lee

---

`phi.stat` *Standardized $\Phi$ statistic*

---

## Description

A test statistic of $\Phi$ is to test network dependence in categorical observations.

## Usage

```
phi.stat(A, Y)
```

## Arguments

| | |
|---|---|
| `A` | `[n x n]` adjacency matrix or general relational weight matrix of which $A_ij$ indicates relationship from $i$ to $j$. |
| `Y` | a vector of $n$ nominal or binary, one-dimensional observations. |

## Value

| | |
|---|---|
| `phi` | a standardized $\Phi$ statistic. |

## Author(s)

Youjin Lee

snowball.sampling *Snowball sampling*

## Description

Sampling from one graph to its induced subgraph depending on network structure.

## Usage

```
snowball.sampling(G, samn)
```

## Arguments

| | |
|---|---|
| G | an igraph object. |
| samn | is a size of snowball sample that will be samples from G. |

## Value

| | |
|---|---|
| subG | an induced subgraph of G sampled using snowball sampling. |
| ind | a set of index of samples. |

## Examples

```
library(netdep)
library(igraph)
G = latent.netdep(n.node = 200, rho = 0.2, dep.factor = -3, dep.range = c(-10, 0))
subG = snowball.sampling(G, 100)
```

# Index