Package ‘fastnet’

December 1, 2020

Type Package
Title Large-Scale Social Network Analysis
Version 1.0.0
Description We present an implementation of the algorithms required to simulate large-scale social networks and retrieve their most relevant metrics. Details can be found in the accompanying scientific paper on the Journal of Statistical Software, <doi:10.18637/jss.v096.i07>.
License GPL (>= 2)
Imports doParallel (>= 1.0.0), foreach (>= 1.5.0), igraph (>= 1.2.0), tidygraph(>= 1.2.0)
Depends R (>= 4.0.0)
Encoding UTF-8
LazyData true
BugReports https://github.com/networkgroupR/fastnet/issues
RoxygenNote 7.1.1
NeedsCompilation no
Author Nazrul Shaikh [aut, cre], Xu Dong [aut], Luis Castro [aut], Christian Llano [ctb]
Maintainer Nazrul Shaikh <networkgroupr@gmail.com>
Repository CRAN
Date/Publication 2020-12-01 07:40:02 UTC

R topics documented:

  degree.collect ........................................ 3
  degree.dist ........................................... 3
  degree.hist ........................................... 4
  draw.net ............................................. 5
R topics documented:

from.adjacency ..................................................  6
from.edgelist ...................................................  6
from.igraph .....................................................  7
from.statnet ...................................................  8
get.neighbors ...................................................  9
metric.cluster.global .........................................  9
metric.cluster.mean ............................................ 10
metric.cluster.median ......................................... 11
metric.degree.effective ........................................ 12
metric.degree.entropy .......................................... 13
metric.degree.max ............................................. 14
metric.degree.max.efficient ................................. 14
metric.degree.mean ........................................... 15
metric.degree.median ......................................... 16
metric.degree.min ............................................. 17
metric.degree.sd ................................................ 17
metric.distance.apl ........................................... 18
metric.distance.diameter ...................................... 19
metric.distance.effdia ......................................... 21
metric.distance.meanecc ....................................... 22
metric.distance.medianecc .................................... 23
metric.distance.mpl ............................................ 24
metric.eigen.mean ............................................. 25
metric.eigen.median .......................................... 26
metric.eigen.value ............................................ 27
metric.graph.density ........................................... 28
net.barabasi.albert ........................................... 28
net.caveman ..................................................... 29
net.cluster.affiliation ........................................ 30
net.complete .................................................... 32
net.degree.constraint ......................................... 32
net.erdos.renyi.gnm .......................................... 33
net.erdos.renyi.gnp .......................................... 34
net.holme.kim ................................................ 35
net.random.plc ................................................ 36
net.rewired.caveman ......................................... 37
net.ring.lattice ................................................. 38
net.watts.strogatz ............................................ 39
preview.deg ..................................................... 40
preview.net ..................................................... 40
to.edgelist ..................................................... 41
to.igraph ....................................................... 42
to.tidygraph .................................................... 42

Index 44
**degree.collect**

*Degrees of nodes*

Description

Collect the degrees for all nodes in a network.

Usage

```r
degree.collect(net)
```

Arguments

- `net` The input network.

Details

Obtain the degrees for all nodes.

Value

A vector.

Author(s)

Xu Dong, Nazrul Shaikh.

Examples

```r
## Not run:
x.deg <- degree.collect(x)
summary(x.deg)
## End(Not run)
```

---

**degree.dist**

*Plot of the degree distribution of a network*

Description

Plot the degree distribution of a network.

Usage

```r
degree.dist(net, cumulative = TRUE, log = TRUE)
```
degree.hist

Histogram of the degree distribution of a network

Description
Plot the histogram of all degrees of a network.

Usage
degree.hist(g, breaks = 100)
Arguments

- `g`: The input network.
- `breaks`: A single number giving the number of cells for the histogram.

Details

Plot the histogram of all degrees of a network.

Value

A .gif plot.

Author(s)

Xu Dong

Examples

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.05)
degree.hist(x)
## End(Not run)
```

---

**draw.net**  
Plot of a small network

Description

Plot a small network.

Usage

```r
draw.net(net)
```

Arguments

- `net`: The input network.

Details

Plot a small network.

Value

A .gif plot.

Author(s)

Xu Dong, Nazrul Shaikh.
from.edgelist

Examples
## Not run:
x <- net.ring.lattice(12,4)
draw.net(x)
## End(Not run)

from.adjacency

Adjacency Matrix to fastnet

Description
Transform an adjacency matrix to an ego-centric list form used in fastnet.

Usage
from.adjacency(adj.mat)

Arguments
adj.mat The input adjacency matrix

Value
A list containing the nodes of the network and their respective neighbors.

Author(s)
Xu Dong, Christian Llano.

Examples
adj.mat <- matrix(c(0,1,0,0,0,0,0,0,0,0,1,0,0,1,0), nrow = 4, ncol = 4)
g <- from.adjacency(adj.mat)

from.edgelist

Edgelist to fastnet

Description
Transform an edgelist to an ego-centric list form used in fastnet.

Usage
from.edgelist(edgelist)
from.igraph

Arguments

edgelist A 2-column data frame, in which the 1st column represents the start nodes, and the 2nd column represents the destination nodes.

Details

Most network data repositories choose to store the data in an edgelist form. This function helps user to load it in fastnet.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong

Examples

```r
edgelist <- data.frame(from=c(1, 3, 2, 3, 3), to=c(4, 5, 6, 5, 7))
g <- from.edgelist(edgelist)
```

from.igraph

Transform an igraph object to a fastnet object

Description

Transform an igraph object to an ego-centric list form used in fastnet.

Usage

```r
from.igraph(net.igraph)
```

Arguments

net.igraph The input igraph object.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong.
Examples

```r
## Not run:
library("igraph")
net.igraph <- erdos.renyi.game(100, 0.1)
g <- from.igraph(net.igraph)
## End(Not run)
```

from.statnet statnet to fastnet

Description

Import a `statnet` object.

Usage

```r
from.statnet(net.statnet, ncores = detectCores())
```

Arguments

- `net.statnet`: The input `statnet` object.
- `ncores`: The number of cores to be used.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong

Examples

```r
## Not run:
library("ergm")
library("doParallel")
data("flo")
nflo <- network(flo, loops = TRUE)
fflo <- from.statnet(nflo)
## End(Not run)
```
get.neighbors

Neighbors of an agent in a network

Description

Presents all neighbors of a given node.

Usage

get.neighbors(net, NodeID)

Arguments

net
The input network.
NodeID
The ID of the input node.

Details

Neighbors of a node are nodes that directly connect to this node.

Value

A vector.

Author(s)

Xu Dong

Examples

```r
## Not run:
x <- net.ring.lattice(12,4)
get.neighbors(x,2)
## End(Not run)
```

metric.cluster.global

Global Clustering Coefficient

Description

Calculate the global clustering coefficient of a graph.

Usage

metric.cluster.global(g)
Arguments

\( g \)  The input network.

Details

The global clustering coefficient measures the ratio of (closed) triples versus the total number of all possible triples in network \( g \). \texttt{metric.cluster.global()} calculates the global clustering coefficient of \( g \).

Value

A real constant.

Author(s)

Xu Dong, Nazrul Shaikh.

References


Examples

\[
\text{## Not run:} \\
x \leftarrow \text{net.erdos.renyi.gnp(1000, 0.01)} \\
\text{metric.cluster.global}(x) \\
\text{## End(Not run)}
\]

---

\texttt{metric.cluster.mean} \hspace{1cm} \textit{Mean Local Clustering Coefficient}

Description

Calculate the average local clustering coefficient of a graph.

Usage

\texttt{metric.cluster.mean}(g)

Arguments

\( g \)  The input network.

Details

The local clustering coefficient of a node is the ratio of the triangles connected to the node and the triples centered on the node. \texttt{metric.cluster.mean()} calculates the (estimated) average clustering coefficient for all nodes in graph \( g \) with a justified error.
Description

Calculate the median local clustering coefficient of a graph.

Usage

metric.cluster.median(g)

Arguments

g The input network.

Details

The local clustering coefficient of a node is the ratio of the triangles connected to the node and the triples centered on the node. `metric.cluster.median()` calculates the (estimated) median clustering coefficient for all nodes in graph `g` with a justified error.

Value

A real constant.

Author(s)

Xu Dong, Nazrul Shaikh.
References


Examples

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.1)
mwu.cluster.median(x)
## End(Not run)
```

---

**metric.degree.effective**

*Effective Degree*

Description

Calculate the effective degree of a network.

Usage

```r
metric.degree.effective(g, effective_rate = 0.9)
```

Arguments

- `g`: The input network.
- `effective_rate`: The effective rate (0.9 is set by default).

Details

The effective degree

Value

A real constant.

Author(s)

Xu Dong, Nazrul Shaikh.

Examples

```r
## Not run:
mwu.degree.effective(x)
## End(Not run)
```
**Description**

Calculate the degree entropy of a graph.

**Usage**

```
metric.degree.entropy(g)
```

**Arguments**

- `g`: The input network.

**Details**

Calculates the degree entropy of graph g, i.e.

\[
Entropy(g) = - \sum_{i=1}^{n} i \cdot \log_2(i)
\]

**Value**

A real constant.

**Author(s)**

Xu Dong, Nazrul Shaikh.

**References**


**Examples**

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.degree.entropy(x)
## End(Not run)
```
**metric.degree.max**  
*Maximal Degree*

**Description**  
Calculate the maximal degree of a graph.

**Usage**  
metric.degree.max(g)

**Arguments**

- **g**  
The input network.

**Details**  
The maximal degree.

**Value**  
A real constant.

**Author(s)**  
Xu Dong, Nazrul Shaikh.

**Examples**

```r
## Not run:
metric.degree.max(x)
## End(Not run)
```

---

**metric.degree.max.efficient**  
*Efficient Maximal Degree*

**Description**  
Calculate the efficient maximal degree of a graph.

**Usage**  
metric.degree.max.efficient(g)
**metric.degree.mean**

**Arguments**

- `g` The input network.

**Details**

The efficient maximal degree is the 90

**Value**

A real constant.

**Author(s)**

Xu Dong, Nazrul Shaikh.

**Examples**

```r
## Not run: x <- net.erdos.renyi.gnp(1000, 0.01)
metric.degree.max.efficient(x)
## End(Not run)
```

---

**Description**

Calculate the mean degree of a graph.

**Usage**

```r
metric.degree.mean(g)
```

**Arguments**

- `g` The input network.

**Details**

The mean degree is the average value of the degrees of all nodes in graph `g`.

**Value**

A real constant.

**Author(s)**

Xu Dong, Nazrul Shaikh.
Examples

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.degree.median(x)

## End(Not run)
```

### Description

Calculate the median degree of a graph.

### Usage

```r
metric.degiree.median(g)
```

### Arguments

- `g`: The input network.

### Details

The median degree is the median value of the degrees of all nodes in graph `g`.

### Value

A real constant.

### Author(s)

Xu Dong, Nazrul Shaikh.

### Examples

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.degiree.median(x)
## End(Not run)
```
**metric.degree.min**  
*Minimal Degree*

**Description**
Calculate the minimal degree of a network.

**Usage**

```r
metric.degree.min(g)
```

**Arguments**
- `g`: The input network.

**Details**
The minimal degree.

**Value**
A real constant.

**Author(s)**
Xu Dong, Nazrul Shaikh.

**Examples**

```r
## Not run:
meteric.degree.min(x)
## End(Not run)
```

---

**metric.degree.sd**  
*Standard Deviation of Degree Distribution*

**Description**
Calculate the standard deviation of all degrees of a network.

**Usage**

```r
metric.degree.sd(g)
```

**Arguments**
- `g`: The input network.
**Details**

The standard deviation of all degrees of a network.

**Value**

A real constant.

**Author(s)**

Xu Dong, Nazrul Shaikh.

**Examples**

```r
## Not run:
metric.degree.sd(x)
## End(Not run)
```

---

**Description**

Calculate the average path length of a graph.

**Usage**

```r
metric.distance.apl(
    Network,
    probability = 0.95,
    error = 0.03,
    Cores = detectCores(),
    full.apl = FALSE
)
```

**Arguments**

- **Network**: The input network.
- **probability**: The confidence level probability.
- **error**: The sampling error.
- **Cores**: Number of cores to use in the computations. By default uses `parallel` function `detectCores()`.
- **full.apl**: It will calculate the sampling version by default. If it is set to true, the population APL will be calculated and the rest of the parameters will be ignored.
Details

The average path length (APL) is the average shortest path lengths of all pairs of nodes in graph Network. `metric.distance.apl` calculates the population APL and estimated APL of graph g with a sampling error set by the user.

The calculation uses a parallel load balancing approach, distributing jobs equally among the cores defined by the user.

Value

A real value.

Author(s)

Luis Castro, Nazrul Shaikh.

References


Examples

```r
## Not run:
## Default function
x <- net.erdos.renyi.gnp(1000,0.01)
metric.distance.apl(x)
## Population APL
metric.distance.apl(x, full.apl=TRUE)
## Sampling at 99% level with an error of 10% using 5 cores
metric.distance.apl(Network = x, probability=0.99, error=0.1, Cores=5)
## End(Not run)
```

---

`metric.distance.diameter`


**Diameter**

Description

Calculate the diameter of a graph.
Usage

metric.distance.diameter(
  Network,
  probability = 0.95,
  error = 0.03,
  Cores = detectCores(),
  full = TRUE
)

Arguments

Network The input network.
probability The confidence level probability
error The sampling error
Cores Number of cores to use in the computations. By default uses parallel function
detectCores().
full It will calculate the popular full version by default. If it is set to FALSE, the
estimated diameter will be calculated.

Details

The diameter is the largest shortest path lengths of all pairs of nodes in graph Network.
metric.distance.diameter calculates the (estimated) diameter of graph Network with a justified
error.

Value

A real value.

Author(s)

Luis Castro, Nazrul Shaikh.

References

(December 1959), 269-271.

Castro L, Shaikh N. Estimation of Average Path Lengths of Social Networks via Random Node Pair
Sampling. Department of Industrial Engineering, University of Miami. 2016.

Examples

## Not run:
##Default function
x <- net.erdos.renyi.gnp(1000,0.01)
metric.distance.diameter(x)
##Population APL
metric.distance.diameter(x, full=TRUE)
metric.distance.effdia

## Sampling at 99% level with an error of 10% using 5 cores
metric.distance.diameter(Network = x, probability=0.99, error=0.1, Cores=5)

## End(Not run)

---

**Effective Diameter**

**Description**

Calculate the effective diameter of a graph.

**Usage**

```r
metric.distance.effdia(
    Network, 
    probability = 0.95, 
    error = 0.03, 
    effective_rate = 0.9, 
    Cores = detectCores(), 
    full = TRUE
)
```

**Arguments**

- **Network**: The input network.
- **probability**: The confidence level probability
- **error**: The sampling error
- **effective_rate**: The effective rate (by default it is set to be 0.9)
- **Cores**: Number of cores to use in the computations. By default uses `parallel` function `detectCores()`.
- **full**: It will calculate the popular full version by default. If it is set to FALSE, the estimated diameter will be calculated.

**Details**

The diameter is the largest shortest path lengths of all pairs of nodes in graph `Network`. `metric.distance.diameter` calculates the (estimated) diameter of graph `Network` with a justified error.

**Value**

A real value.
Author(s)

Luis Castro, Nazrul Shaikh.

References


Examples

```r
## Not run:
## Default function
x <- net.erdos.renyi.gnp(1000,0.01)
metric.distance.effdia(x)
## Population APL
metric.distance.effdia(x, full=TRUE)
## Sampling at 99% level with an error of 10% using 5 cores
metric.distance.effdia(Network = x, probability=0.99, error=0.1, Cores=5)

## End(Not run)
```

metric.distance.meanecc

*Mean Eccentricity*

Description

Calculate the mean eccentricity of a graph.

Usage

```r
metric.distance.meanecc(g, p)
```

Arguments

- `g`: The input network.
- `p`: The sampling probability.

Details

The mean eccentricities of all nodes in graph `g`. Calculates the (estimated) mean eccentricity of graph `g` with a justified error.
Value

A real constant.

Author(s)

Xu Dong, Nazrul Shaikh.

References


Examples

```
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.distance.medianecc(x, 0.01)
## End(Not run)
```

Description

Calculate the (estimated) median eccentricity of a graph.

Usage

```
metric.distance.medianecc(g, p)
```

Arguments

- **g**
  - The input network.
- **p**
  - The sampling probability.

Details

Is the median eccentricities of all nodes in graph g. `metric.distance.medianecc` calculates the (estimated) median eccentricity of graph g with a justified error.

Value

A real constant.

Author(s)

Xu Dong, Nazrul Shaikh.
### Not run:
```r
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.distance.medianecc(x, 0.01)
## End(Not run)
```

### Description
Calculate the median path length (MPL) of a network.

### Usage
```r
metric.distance.mpl(
  Network,
  probability = 0.95,
  error = 0.03,
  Cores = detectCores(),
  full = FALSE
)
```

### Arguments
- **Network**: The input network.
- **probability**: The confidence level probability
- **error**: The sampling error
- **Cores**: Number of cores to use in the computations. By default detectCores() from `parallel`.
- **full**: It calculates the sampling version by default. If it is set to true, the population MPL will be calculated and the rest of the parameters will be ignored.

### Details
The median path length (MPL) is the median shortest path lengths of all pairs of nodes in Network. `metric.distance.mpl(g)` calculates the population MPL OR estimated MPL of network g with a sampling error set by the user. The calculation uses a parallel load balancing approach, distributing jobs equally among the cores defined by the user.

### Value
A real integer
**Author(s)**

Luis Castro, Nazrul Shaikh.

**References**


**Examples**

```r
## Not run:
## Default function
x <- net.erdos.renyi.gnp(1000,0.01)
metric.distance.mpl(x)
## Population MPL
metric.distance.mpl(x, full=TRUE)
## Sampling at 99% level with an error of 10% using 5 cores
metric.distance.mpl(Network = x, probability=0.99, error=0.1, Cores=5)
## End(Not run)
```

### Description

Calculate the mean eigenvalue centrality of a graph.

### Usage

```r
metric.eigen.mean(g)
```

### Arguments

- `g`: The input network.

### Details

`metric.eigen.mean` calculates the mean eigenvalue centrality score of graph `g`.

### Value

A real constant.
Author(s)
Xu Dong, Nazrul Shaikh.

References

Examples
```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.eigen.mean(x)
## End(Not run)
```

---

**metric.eigen.median**  
*Median Eigenvalue Centrality*

Description
Calculate the median eigenvalue centrality of a graph.

Usage
```r
metric.eigen.median(g)
```

Arguments
- `g` The input network.

Details
`metric.eigen.median` calculates the median eigenvalue centrality score of graph `g`.

Value
A real constant.

Author(s)
Xu Dong, Nazrul Shaikh.

References
metric.eigen.value

Examples

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.eigen.median(x)
## End(Not run)
```

<table>
<thead>
<tr>
<th>metric.eigen.value</th>
<th>Eigenvalue Score</th>
</tr>
</thead>
</table>

Description

Calculate the eigenvalue centrality score of a graph.

Usage

`metric.eigen.value(g)`

Arguments

- `g`: The input network.

Details

`metric.eigen.value` calculates the eigenvalue centrality score of graph `g`.

Value

A real constant.

Author(s)

Xu Dong, Nazrul Shaikh.

References


Examples

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.eigen.value(x)
## End(Not run)
```
**metric.graph.density**  
*Graph Density*

**Description**

Calculate the density of a graph.

**Usage**

```r
metric.graph.density(g)
```

**Arguments**

- `g`  
The input network.

**Details**

Computes the ratio of the number of edges and the number of possible edges.

**Value**

A real constant.

**Author(s)**

Xu Dong, Nazrul Shaikh.

**Examples**

```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
metric.graph.density(x)
## End(Not run)
```

---

**net.barabasi.albert**  
*Barabasi-Albert Scale-free Graph*

**Description**

Simulate a scale-free network using a preferential attachment mechanism (Barabasi and Albert, 1999)

**Usage**

```r
net.barabasi.albert(n, m, ncores = detectCores(), d = FALSE)
```
net.caveman

Arguments

\texttt{m} \hspace{1cm} \text{Number of cliques (or caves) in the network.}
\texttt{k} \hspace{1cm} \text{Number of nodes per clique.}
\texttt{ncores} \hspace{1cm} \text{Number of cores, by default detectCores() from parallel.}

Details

Simulate a (connected) caveman network of \texttt{m} cliques of size \texttt{k}.

Examples

```
## Not run:
x <- net.caveman(1000, 20) # using default ncores
## End(Not run)
```

Description

Simulate a (connected) caveman network of \texttt{m} cliques of size \texttt{k}.

Value

A list containing the nodes of the network and their respective neighbors.

References


Author(s)

Luis Castro, Xu Dong, Nazrul Shaikh.
Details

The (connected) caveman network is formed by connecting a set of isolated \( k \) - cliques (or "caves"), neighbor by neighbor and head to toe, using one edge that removed from each clique such that all \( m \) cliques form a single circle (Watts 1999). The total number of nodes, i.e. \( n \), in this network is given by \( k + m \).

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong, Nazrul Shaikh.

References


Examples

```r
# Not run:
x <- net.caveman(50, 20) # using ncores by default
# End(Not run)
```

```
net.cluster.affiliation

Generate a cluster-affiliation graph

Description

Generate a cluster-affiliation graph.

Usage

```r
net.cluster.affiliation(
  DEG,
  community_affiliation_alpha,  
  community_affiliation_lambda,  
  community_affiliation_min,  
  community_size_alpha,  
  community_size_lambda,  
  community_size_min
)
```
net.cluster.affiliation

Arguments

DEG Degree sequence.
community_affiliation_alpha First scaling parameter of the membership distribution.
community_affiliation_lambda Second scaling parameter of the membership distribution.
community_affiliation_min Minimal membership.
community_size_alpha First scaling parameter of the cluster-size distribution.
community_size_lambda Second scaling parameter of the cluster-size distribution.
community_size_min Minimal size of a cluster.

Details

The generated network has multiple (overlapping) densely-connected clusters.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong, Nazrul Shaikh

References


Examples

```r
# Not run:
DEG <- sample(seq(5,15),100, replace=TRUE)
x <- net.cluster.affiliation(DEG,
  community_affiliation_alpha=1.5,
  community_affiliation_lambda=10,
  community_affiliation_min=1,
  community_size_alpha=2.5,
  community_size_lambda=40,
  community_size_min=3)
# End(Not run)
```
net.complete

Complete Network

Description
Simulate a complete (or full) network.

Usage
net.complete(n, ncores = detectCores())

Arguments
- n: Number of nodes of the network.
- ncores: Number of cores, by default detectCores() from parallel.

Details
The n nodes in the network are fully connected.
Note that the input n should not exceed 10000, for the sake of memory overflow.

Value
A list containing the nodes of the network and their respective neighbors.

Author(s)
Xu Dong, Nazrul Shaikh.

Examples
## Not run:
x <- net.complete(1000) #using ncores by default
## End(Not run)

net.degree.constraint

Generate a degree-constraint graph

Description
Generate a degree-constraint graph.

Usage
net.degree.constraint(DEG, c.alpha, c.min)
net.erdos.renyi.gnm

Arguments

- **DEG**: Degree sequence.
- **c.alpha**: Scaling parameter of the community-size distribution.
- **c.min**: Minimal size of a community.

Details

The generated network has a pre-defined degree sequence with multiple (overlapping) communities.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong, Nazrul Shaikh

References


Examples

```r
## Not run:
DEG <- sample(seq(5,15),100, replace=TRUE)
x <- net.degree.constraint(DEG, c.alpha=2, c.min=3)
## End(Not run)
```

net.erdos.renyi.gnm

Directed / Undirected Erdos-Renyi $G(n,m)$ network using a fix edge size.

Description

Simulate a random network with $n$ nodes and $m$ edges, according to Erdos and Renyi (1959).

Usage

```r
net.erdos.renyi.gnm(n, m, ncores = detectCores(), d = TRUE)
```

Arguments

- **n**: Number of nodes of the network.
- **m**: Number of edges of the network.
- **ncores**: Number of cores, by default `detectCores()` from parallel.
- **d**: A logical value determining whether is a network directed (default) or indiected.
Details

In this (simplest) random network, \(m\) edges are formed at random among \(n\) nodes. When \(d = \text{TRUE}\) is a directed network.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong, Nazrul Shaikh.

References


Examples

```r
## Not run:
x <- net.erdos.renyi.gnm(1000, 100)
## End(Not run)
```

```
net.erdos.renyi.gnp

Directed / Undirected Erdos-Renyi \(G(n, p)\) network

Description

Simulate a random network with \(n\) nodes and a link connecting probability of \(p\), according to Edos and Renyi (1959).

Usage

```r
net.erdos.renyi.gnp(n, p, ncores = detectCores(), d = TRUE)
```

Arguments

- \(n\) Number of nodes of the network.
- \(p\) Connecting probability.
- \(ncores\) Number of cores, by default `detectCores()` from `parallel`.
- \(d\) A logical value determining whether is a network directed (default) or undirected.

Details

In this (simplest) random network, each edge is formed at random with a constant probability. When \(d = \text{TRUE}\) is a directed network.
Value
A list containing the nodes of the network and their respective neighbors.

Author(s)
Luis Castro, Xu Dong, Nazrul Shaikh.

References

Examples
```r
## Not run:
x <- net.erdos.renyi.gnp(1000, 0.01)
## End(Not run)
```

Description
Simulate a scale-free network with relatively high clustering, comparing to B-A networks (Holme and Kim, 1999).

Usage
```
net.holme.kim(n, m, pt)
```

Arguments
```
n   Number of nodes of the network.
m   Number of nodes to which a new node connects at each iteration.
pt  Triad formation probability after each preferential attachment mechanism.
```

Details
The Holme-Kim network model is a simple extension of B-A model. It adds an additional step, called "Triad formation", with the probability $pt$ that compensates the low clustering in B-A networks.

Value
A list containing the nodes of the network and their respective neighbors.

Author(s)
Xu Dong, Nazrul Shaikh
References


Examples

```r
## Not run:
x <- net.holme.kim(1000, 20, 0.1)
## End(Not run)
```

---

net.random.plc  Random Network with a Power-law Degree Distribution that Has An Exponential Cutoff

Description

Simulate a random network with a power-law degree distribution that has an exponential cutoff, according to Newman et al. (2001).

Usage

```r
net.random.plc(n, cutoff, exponent)
```

Arguments

- `n`: The number of the nodes in the network.
- `cutoff`: Exponential cutoff of the degree distribution of the network.
- `exponent`: Exponent of the degree distribution of the network.

Details

The generated random network has a power-law degree distribution with an exponential degree cutoff.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong, Nazrul Shaikh

References

Examples

```r
## Not run:
x <- net.random.plc(1000, 10, 2)
## End(Not run)
```

### Description

Simulate a rewired caveman network of \( m \) cliques of size \( k \), and with a link rewiring probability \( p \).

### Usage

```r
net.rewired.caveman(nc, m, p, seed = 99)
```

### Arguments

- `nc`: Number of cliques (or caves) in the network.
- `m`: Number of nodes per clique.
- `p`: Link rewiring probability.
- `seed`: A random seed.

### Details

The rewired caveman network is built on the corresponding regular caveman network with \( m \) cliques of size \( k \). Then the links in this caveman network are rewired with probability \( p \).

### Value

A list containing the nodes of the network and their respective neighbors.

### Author(s)

Xu Dong, Nazrul Shaikh

### References


### Examples

```r
## Not run:
x <- net.rewired.caveman(50, 20, 0.0005)
## End(Not run)
```
Description

Simulate a network with a $k$-regular ring lattice structure.

Usage

net.ring.lattice(n, k)

Arguments

n  Number of nodes in the network.

k  Number of edges per node.

Details

The $n$ nodes are placed on a circle and each node is connected to the nearest $k$ neighbors.

Value

A list containing the nodes of the network and their respective neighbors.

Author(s)

Xu Dong, Nazrul Shaikh

References


Examples

```r
## Not run:
x <- net.ring.lattice(1000, 10)
## End(Not run)
```
**Description**

Simulate a small-world network according to the model of Watts and Strogatz (1998).

**Usage**

```r
net.watts.strogatz(n, k, re)
```

**Arguments**

- `n`: The number of the nodes in the network (or lattice).
- `k`: Number of edges per node.
- `re`: Rewiring probability.

**Details**

The formation of Watts-Strogatz network starts with a ring lattice with \( n \) nodes and \( k \) edges per node, then each edge is rewired at random with probability \( re \).

**Value**

A list containing the nodes of the network and their respective neighbors.

**Author(s)**

Xu Dong, Nazrul Shaikh

**References**


**Examples**

```r
## Not run:
x <- net.watts.strogatz(1000, 10, 0.05)
## End(Not run)
```
preview.deg

Description
Present the first 10 degrees of a network.

Usage
preview.deg(g)

Arguments
g 

The input network.

Details
Present the first 10 degrees of a network.

Value
A vector.

Author(s)
Xu Dong, Nazrul Shaikh.

Examples

## Not run:
x <- net.ring.lattice(12,4)
preview.deg(x)
## End(Not run)

preview.net

Description
Present the first 10 ego-centric lists of a network.

Usage
preview.net(net)
to.edgelist

Arguments
  net The input network.

Details
  the connection condition of the first 10 nodes in a network.

Value
  A list.

Author(s)
  Xu Dong, Nazrul Shaikh.

Examples
  ```
  ## Not run:
  x <- net.ring.lattice(12,4)
  preview.net(x)
  ## End(Not run)
  ```

Description
  Coerce a fastnet object to edgelist.

Usage
  ```r
to.edgelist(network, ncores)
  ```

Arguments
  ```
  network A fastnet object.
  ncores The number of cores to be used.
  ```

Value
  A 2-column list with each row representing an edge, from source to destination

Author(s)
  Xu Dong
## Not run:
g <- net.erdos.renyi.gnp(100, 0.1)
el <- to.edgelist(g)
## End(Not run)

---

### Description

Coerce a fastnet object to an igraph object

### Usage

```r
to.igraph(g)
```

### Arguments

- `g`: A fastnet object

### Value

An igraph object

### Author(s)

Xu Dong

---

### Description

Coerce a fastnet object to a tidygraph object.

### Usage

```r
to.tidygraph(g)
```

### Arguments

- `g`: A fastnet object.

### Value

A tidygraph object
Author(s)

Xu Dong
Index

degree.collect, 3
degree.dist, 3
degree.hist, 4
draw.net, 5
from.adjacency, 6
from.edgelist, 6
from.igraph, 7
from.statnet, 8
get.neighbors, 9
metric.cluster.global, 9
metric.cluster.mean, 10
metric.cluster.median, 11
metric.degree.effective, 12
metric.degree.entropy, 13
metric.degree.max, 14
metric.degree.max.efficient, 14
metric.degree.mean, 15
metric.degree.median, 16
metric.degree.min, 17
metric.degree.sd, 17
metric.distance.apl, 18
metric.distance.diameter, 19
metric.distance.effdia, 21
metric.distance.mean ecc, 22
metric.distance.median ecc, 23
metric.distance.mpl, 24
metric.eigen.mean, 25
metric.eigen.median, 26
metric.eigen.value, 27
metric.graph.density, 28
net.barabasi.albert, 28
net.caveman, 29
net.cluster.affiliation, 30
net.complete, 32
net.degree.constraint, 32
net.erdos.renyi.gnm, 33
net.erdos.renyi.gnp, 34
net.holme.kim, 35
net.random.plc, 36
net.rewired.caveman, 37
net.ring.lattice, 38
net.watts.strogatz, 39
preview.deg, 40
preview.net, 40
to.edgelist, 41
to.igraph, 42
to.tidygraph, 42