Package ‘netUtils’

June 29, 2023

Title A Collection of Tools for Network Analysis
Version 0.8.2
Description Provides a collection of network analytic (convenience) functions which are missing in other standard packages. This includes triad census with attributes <doi:10.1016/j.socnet.2019.04.003>, core-periphery models <doi:10.1016/S0378-8733(99)00019-2>, and several graph generators. Most functions are build upon 'igraph'.

URL https://github.com/schochastics/netUtils/
BugReports https://github.com/schochastics/netUtils/issues
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.2.3
LinkingTo Rcpp, RcppArmadillo
Imports Rcpp, igraph, stats
Suggests covr, GA, testthat (>= 3.0.0)
Config/testthat/edition 3
NeedsCompilation yes
Author David Schoch [aut, cre] (<https://orcid.org/0000-0003-2952-4812>)
Maintainer David Schoch <david@schochastics.net>
Repository CRAN
Date/Publication 2023-06-29 16:40:08 UTC

R topics documented:

as_adj_list1 .................................................. 2
as_adj_weighted ............................................. 3
as_multi_adj ................................................ 4
bipartite_from_data_frame ................................. 4
clique_vertex_mat .......................................... 5
core_periphery ............................................. 6
dyad_census_attr .......................................... 7
as_adj_list1

Description

Create adjacency lists from a graph, either for adjacent edges or for neighboring vertices. This version is faster than the version of igraph but less general.

Usage

as_adj_list1(g)

Arguments

g An igraph object

Details

The function does not have a mode parameter and only returns the adjacency list comparable to as_adj_list(g, mode="all")

Value

A list of numeric vectors.

Author(s)

David Schoch
Examples

```r
library(igraph)
g <- make_ring(10)
as_adj_list!1(g)
```

---

as_adj_weighted weighted dense adjacency matrix

Description

returns the weighted adjacency matrix in dense format

Usage

```r
as_adj_weighted(g, attr = NULL)
```

Arguments

- `g`: An igraph object
- `attr`: Either NULL or a character string giving an edge attribute name. If NULL a traditional adjacency matrix is returned. If not NULL then the values of the given edge attribute are included in the adjacency matrix.

Details

This method is faster than as_adj from igraph if you need the weighted adjacency matrix in dense format

Value

Numeric matrix

Author(s)

David Schoch

Examples

```r
library(igraph)
g <- sample_gnp(10, 0.2)
E(g)$weight <- runif(ecount(g))
as_adj_weighted(g, attr="weight")
```
as_multi_adj  Convert a list of graphs to an adjacency matrices

Description
Convenience function that turns a list of igraph objects into adjacency matrices.

Usage
as_multi_adj(g_lst, attr = NULL, sparse = FALSE)

Arguments
- **g_lst**: A list of igraph object
- **attr**: Either NULL or a character string giving an edge attribute name. If NULL a binary adjacency matrix is returned.
- **sparse**: Logical scalar, whether to create a sparse matrix. The 'Matrix' package must be installed for creating sparse matrices.

Value
List of numeric matrices

Author(s)
David Schoch

bipartite_from_data_frame  two-mode network from a data.frame

Description
Create a two-mode network from a data.frame

Usage
bipartite_from_data_frame(d, type1, type2, attr = NULL, weighted = TRUE)

Arguments
- **d**: data.frame
- **type1**: column name of mode 1
- **type2**: column name of mode 2
- **attr**: named list of edge attributes
- **weighted**: should a weighted graph be created if multiple edges occur
clique_vertex_mat

Value

two mode network as igraph object

Author(s)
David Schoch

Examples

```r
library(igraph)
edges <- data.frame(mode1=1:5,mode2=letters[1:5])
bipartite_from_data_frame(edges,"mode1","mode2")
```

---

clique_vertex_mat  Clique Vertex Matrix

Description

Creates the clique vertex matrix with entries (i,j) equal to one if node j is in clique i

Usage

```r
clique_vertex_mat(g)
```

Arguments

- `g`  
  An igraph object

Value

Numeric matrix

Author(s)
David Schoch

Examples

```r
library(igraph)
g <- sample_gnp(10,0.2)
clique_vertex_mat(g)
```
core_periphery

Description

Fits a discrete core-periphery model to a given network

Usage

core_periphery(graph, method = "rk1_dc", iter = 500, ...)

Arguments

graph igraph object
method algorithm to use (see details)
iter number of iterations if method=GA
... other parameters for GA

Details

The function fits the data to an optimal pattern matrix with a genetic algorithm (method="GA") or a rank 1 approximation, either with degree centrality (method="rk1_dc") or eigenvector centrality (method="rk1_ec"). The rank 1 approximation is computationally far cheaper but also more experimental. Best is to compare the results from both models.

Value

list with numeric vector with entries (k1,k2,...ki...) where ki assigns vertex i to either the core (ki=1) or periphery (ki=0), and the maximal correlation with an optimal pattern matrix

Author(s)

David Schoch

References


Examples

set.seed(121)
#split graphs have a perfect core-periphery structure
sg <- split_graph(n = 20, p = 0.3,core = 0.5)
core_periphery(sg)
dyad_census_attr

dyad census with node attributes

Description

dyad census with node attributes

Usage

dyad_census_attr(g, vattr)

Arguments

g igraph object. should be a directed graph.
vattr name of vertex attribute to be used.

Details

The node attribute should be integers from 1 to max(attr)

Value

dyad census as a data.frame.

Author(s)

David Schoch

Examples

library(igraph)
g <- sample_gnp(10,0.4,directed = TRUE)
V(g)$attr <- c(rep(1,5),rep(2,5))
dyad_census_attr(g,"attr")

fast_cliques

Find Cliques, maximal or not, fast

Description

Enumerates all ( maximal) cliques using MACE. Can be faster than igraph in some circumstances

Usage

fast_cliques(g, what = "M", min = NULL, max = NULL, outfile = NA)
Arguments

- `g`: An igraph object
- `what`: either "M" for maximal cliques or "C" for all cliques
- `min`: Numeric constant, lower limit on the size of the cliques to find. NULL means no limit, i.e., it is the same as 0
- `max`: Numeric constant, upper limit on the size of the cliques to find. NULL means no limit
- `outfile`: character. If not NA, cliques are written to file

Details

C Code downloaded from http://research.nii.ac.jp/~uno/codes.htm. Download the code and run `make` and then point an environment variable called `MACE_PATH` to the binary. See http://research.nii.ac.jp/~uno/code/mace.html for more details. MACE is faster than igraph for dense graphs.

Value

a list containing numeric vectors of vertex ids. Each list element is a clique. If `outfile`!=NA, the output is written to the specified file

Author(s)

David Schoch

References


---

**graph_cartesian**

Cartesian product of two graphs

Description

Compute the Cartesian product of two graphs

Usage

`graph_cartesian(g, h)`

Arguments

- `g`: An igraph object
- `h`: An igraph object
Details


Value

Cartesian product as igraph object

Author(s)

David Schoch

Examples

library(igraph)
g <- make_ring(4)
h <- make_full_graph(2)
graph_cartesian(g, h)

Description

This function computes the correlation between networks. Implemented methods expect the graph to be an adjacency matrix, an igraph, or a network object.

Usage

graph_cor(object1, object2)

## Default S3 method:
graph_cor(object1, object2)

## S3 method for class 'igraph'
graph_cor(object1, object2, ...)

## S3 method for class 'matrix'
graph_cor(object1, object2)

## S3 method for class 'array'
graph_cor(object1, object2)

Arguments

object1 igraph object or adjacency matrix
object2 igraph object or adjacency matrix over the same vertex set as object1
... additional arguments
Value

correlation between graphs

---

**graph_direct**

*Direct product of two graphs*

Description

Compute the direct product of two graphs

Usage

`graph_direct(g, h)`

Arguments

- `g`: An igraph object
- `h`: An igraph object

Details


Value

Direct product as igraph object

Author(s)

David Schoch

Examples

```r
library(igraph)
g <- make_ring(4)
h <- make_full_graph(2)
graph_direct(g,h)
```
Description

Create a list of igraph objects from an edgelist according to a type attribute

Usage

```r
graph_from_multi_edgelist(
  d,
  from = NULL,
  to = NULL,
  type = NULL,
  weight = NULL,
  directed = FALSE
)
```

Arguments

da data frame.
from column name of sender. If NULL, defaults to first column.
to column of receiver. If NULL, defaults to second column.
type type attribute to split the edgelist. If NULL, defaults to third column.
weight optional column name of edge weights. Ignored if NULL.
directed logical scalar, whether or not to create a directed graph.

Value

list of igraph objects.

Author(s)

David Schoch

Examples

```r
library(igraph)
d <- data.frame(from=rep(c(1,2,3),3),to=rep(c(2,3,1),3),
  type=rep(c("a","b","c"),each=3),weight=1:9)
graph_from_multi_edgelist(d,"from","to","type","weight")
```
**graph_kpartite**

**k partite graphs**

**Description**
Create a random k-partite graph.

**Usage**

graph_kpartite(n = 10, grp = c(5, 5))

**Arguments**

- **n**
  number of nodes
- **grp**
  vector of partition sizes

**Value**
igraph object

**Author(s)**
David Schoch

**Examples**

```
#3-partite graph with equal sized groups
graph_kpartite(n = 15, grp = c(5,5,5))
```

---

**graph_to_sage**

**convert igraph object to sage format**

**Description**
convert igraph object to sage format to be read in SAGE

**Usage**

```r
graph_to_sage(g)
```

**Arguments**

- **g**
  igraph object

**Value**

sage string
helpers

Author(s)
David Schoch

Description
small functions to deal with typical network problems

Usage
biggest_component(g)
delete_isolates(g)

Arguments
\( g \)  
igraph object

Value
igraph object

Author(s)
David Schoch

reciprocity_cor  Reciprocity correlation coefficient

Description
Reciprocity correlation coefficient

Usage
reciprocity_cor(g)

Arguments
\( g \)  
igraph object. should be a directed graph
Details

The usual definition of reciprocity has some defects. It cannot tell the relative difference of reciprocity compared with purely random network with the same number of vertices and edges. The useful information from reciprocity is not the value itself, but whether mutual links occur more or less often than expected by chance.

To overcome this issue, reciprocity can be defined as the correlation coefficient between the entries of the adjacency matrix of a directed graph:

\[ \frac{\sum_{i \neq j} (a_{ij} - a')(a_{ji} - a')}{\sum_{i \neq j} (a_{ij} - a')^2} \]

where \( a' \) is the density of \( g \).

This definition gives an absolute quantity which directly allows one to distinguish between reciprocal (>0) and antireciprocal (< 0) networks, with mutual links occurring more and less often than random respectively.

Value

Reciprocity as a correlation

Author(s)

David Schoch

References


Examples

library(igraph)
g <- sample_gnp(20,p = 0.3,directed = TRUE)
reciprocity(g)
reciprocity_cor(g)

---

sample_coreseq  Generate random graphs with a given coreness sequence

Description

Similar to sample_degseq just with coreness

Usage

sample_coreseq(cores)
sample_lfr

Arguments

- cores
  coreness sequence

Details

The code is an adaption of the python code from https://github.com/ktvank/Random-Graphs-with-Prescribed-K-Core-Sequences/

Value

igraph object of graph with the same coreness sequence as the input

Author(s)

David Schoch

References


Examples

library(igraph)
g1 <- make_graph("Zachary")
kcores1 <- coreness(g1)
g2 <- sample_coreseq(kcores1)
kcores2 <- coreness(g2)

# the sorted arrays are the same
all(sort(kcores1)==sort(kcores2))
Usage

```r
sample_lfr(
  n,
  tau1,
  tau2,
  mu,
  average_degree = NULL,
  max_degree = NULL,
  min_community = NULL,
  max_community = NULL,
  on = 0,
  om = 0
)
```

Arguments

- **n**: Number of nodes in the created graph.
- **tau1**: Power law exponent for the degree distribution of the created graph. This value must be strictly greater than one.
- **tau2**: Power law exponent for the community size distribution in the created graph. This value must be strictly greater than one.
- **mu**: Fraction of inter-community edges incident to each node. This value must be in the interval 0 to 1.
- **average_degree**: Desired average degree of nodes in the created graph. This value must be in the interval 0 to n. Exactly one of this and `min_degree` must be specified, otherwise an error is raised.
- **max_degree**: Maximum degree of nodes in the created graph. If not specified, this is set to `n-1`.
- **min_degree**: Minimum size of communities in the graph. If not specified, this is set to `min_degree`.
- **max_degree**: Maximum size of communities in the graph. If not specified, this is set to `n`, the total number of nodes in the graph.
- **on**: number of overlapping nodes
- **om**: number of memberships of the overlapping nodes

Details

code adapted from https://github.com/synwalk/synwalk-analysis/tree/master/lfr_generator

Value

an igraph object

References

**Examples**

```r
# Simple Girven-Newman benchmark graphs
g <- sample_lfr(n = 128, average_degree = 16,
               max_degree = 16, mu = 0.1,
               min_community = 32, max_community = 32)
```

**Description**

A graph of \( n \) nodes is grown by attaching new nodes each with \( m \) edges that are preferentially attached to existing nodes with high degree, depending on the homophily parameters.

**Usage**

```r
sample_pa_homophilic(
  n,
  m,
  minority_fraction,
  h_ab,
  h_ba = NULL,
  directed = FALSE
)
```

**Arguments**

- `n` number of nodes
- `m` number of edges a new node is connected to
- `minority_fraction` fraction of nodes that belong to the minority group
- `h_ab` probability to connect a node from group a with group b
- `h_ba` probability to connect a node from group b with group a. If NULL, \( h_{ab} \) is used.
- `directed` should a directed network be created

**Details**

The code is an adaption of the python code from https://github.com/gesiscss/HomophilicNtwMinorities/

**Value**

igraph object

**Author(s)**

David Schoch #maximally heterophilic network `sample_pa_homophilic(n = 50, m = 2, minority_fraction = 0.2, h_ab = 1)` #maximally homophilic network `sample_pa_homophilic(n = 50, m = 2, minority_fraction = 0.2, h_ab = 0)`
References


Description

Create a random split graph with a perfect core-periphery structure.

Usage

```r
split_graph(n, p, core)
```

Arguments

- `n`: number of nodes
- `p`: probability of peripheral nodes to connect to the core nodes
- `core`: fraction of nodes in the core

Value

igraph object

Author(s)

David Schoch

Examples

```r
# split graph with 20 nodes and a core size of 10
split_graph(n = 20, p = 0.4, 0.5)
```
**str.igraph**  
*Print graphs to terminal*

**Description**  
Prints an igraph object to terminal (different than the standard igraph method)

**Usage**  
```r
## S3 method for class 'igraph'
str(object, ...)
```

**Arguments**  
- `object`: An igraph object
- `...`: additional arguments to print (ignored)

**Value**  
str does not return anything. The obvious side effect is output to the terminal.

**Author(s)**  
David Schoch

---

**structural_equivalence**  
*Maximal Structural Equivalence*

**Description**  
Calculates structural equivalence for an undirected graph

**Usage**  
```r
structural_equivalence(g)
```

**Arguments**  
- `g`: An igraph object

**Details**  
Two nodes u and v are structurally equivalent if they have exactly the same neighbors. The equivalence classes produced with this function are either cliques or empty graphs.
**triad_census_attr**

**Value**

vector of equivalence classes

**Author(s)**

David Schoch

---

**triad_census_attr**  *triad census with node attributes*

**Description**

triad census with node attributes

**Usage**

```r
triad_census_attr(g, vattr)
```

**Arguments**

- `g`  
  igraph object. should be a directed graph
- `vattr`  
  name of vertex attribute to be used

**Details**

The node attribute should be integers from 1 to max(attr). The output is a named vector where the names are of the form Txxx-abc, where xxx corresponds to the standard triad census notation and "abc" are the attributes of the involved nodes.

The implemented algorithm is comparable to the algorithm in Lienert et al.

**Value**

triad census with node attributes

**Author(s)**

David Schoch

**References**

Examples

library(igraph)
set.seed(112)
g <- sample_gnp(20, p = 0.3, directed = TRUE)
# add a vertex attribute
V(g)$type <- rep(1:2, each = 10)
triad_census_attr(g, "type")
Index

as_adj_list1, 2
as_adj_weighted, 3
as_multi_adj, 4

biggest_component (helpers), 13
bipartite_from_data_frame, 4

clique_vertex_mat, 5
core_periphery, 6
coreness, 14

delete_isolates (helpers), 13
dyad_census_attr, 7

fast_cliques, 7
graph_cartesian, 8
graph_cor, 9
graph_direct, 10
graph_from_multi_edgelist, 11
graph_kpartite, 12
graph_to_sage, 12

helpers, 13

reciprocity_cor, 13

sample_coreseq, 14
sample_degseq, 14
sample_lfr, 15
sample_pa_homophilic, 17
split_graph, 18
str.igraph, 19
structural_equivalence, 19

triad_census_attr, 20