Package 'migraph'

December 20, 2022

Title Tools for Multimodal Network Analysis

Version 0.13.2

Date 2022-12-20

Description A set of tools for analysing multimodal networks.
All functions operate with matrices, edge lists,
and 'igraph', 'network', and 'tidygraph' objects,
and on one-mode, two-mode (bipartite), and sometimes three-mode networks.
It includes functions for measuring
centrality, centralization, cohesion, closure, and constraint,
as well as for network block-modelling, regression, and diffusion models.
The package is released as a complement to
'Multimodal Political Networks' (2021, ISBN:9781108985000),
and includes various datasets used in the book in addition to other network data.

URL https://github.com/snlab-ch/migraph

BugReports https://github.com/snlab-ch/migraph/issues

Depends R (>= 3.6.0)

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

LazyData true

RoxygenNote 7.2.2

Imports BiocManager, dplyr, generics, ggforce, ggplot2, ggraph,
igraph, methods, network, future, furrr, patchwork, pillar,
purrr, rlang, sna, tidygraph, tidyr

Suggests concaveman, covr, ggdendro, knitr, minMSE, oaqc, readxl,
Rgraphviz, RSiena, rmarkdown, roxygen2, rsconnect, testthat,
xml2

VignetteBuilder knitr

NeedsCompilation no

Author James Hollway [cre, aut, ctb] (IHEID,
<https://orcid.org/0000-0002-8361-9647>),
Jael Tan [ctb] (IHEID, <https://orcid.org/0000-0002-6234-9764>),
Bernhard Bieri [ctb] (<https://orcid.org/0000-0001-5943-9059>),
Henrique Sposito [ctb] (IHEID, <https://orcid.org/0000-0003-3420-6085>)

Maintainer James Hollway <james.hollway@graduateinstitute.ch>

Repository CRAN

Date/Publication 2022-12-20 16:20:02 UTC

R topics documented:

---

add *Adding and copying attributes from one graph to another*

Description

These functions allow users to add attributes to a graph from another graph or from a specified vector supplied by the user.

Usage

```
add_node_attribute(object, attr_name, vector)

add_tie_attribute(object, attr_name, vector)

copy_node_attributes(object, object2)

join_ties(object, object2, attr_name)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package
`attr_name`	Name of the new attribute in the resulting object.
`vector`	A vector of values for the new attribute.
`object2`	A second object to copy nodes or edges from.

Functions

- `add_node_attribute()`: Insert specified values from a vector into the graph as node attributes
- `add_tie_attribute()`: Insert specified values from a vector into the graph as tie attributes
- `copy_node_attributes()`: Copies node attributes from a given graph into specified graph
- `join_ties()`: Copies ties from another graph to specified graph and adds a tie attribute identifying the ties that were newly added

See Also

Other manipulations: `as()`, `grab`, `reformat`, `split()`, `transform()`

Examples

```
add_node_attribute(mpn_elite_mex, "wealth", 1:35)
add_node_attribute(mpn_elite_usa_advice, "wealth", 1:14)
add_tie_attribute(ison_adolescents, "weight", c(1,2,1,1,1,3,2,2,3,1))
autographr(mpn_elite_mex)
both <- join_ties(mpn_elite_mex, generate_random(mpn_elite_mex), "random")
autographr(both)
random <- to_uniplex(both, "random")
autographr(random)
autographr(to_uniplex(both, "orig"))
```

---

as *Coercion between migraph-compatible object classes*

Description

The `as_` functions in `{migraph}` coerce objects between several common classes of social network objects. These include:

- edgelists, as data frames or tibbles
- adjacency (one-mode/unipartite) and incidence (two-mode/bipartite) matrices
- `{igraph}` `graph` objects
- `{tidygraph}` `tbl_graph` objects
- `{network}` `network` objects

An effort is made for all of these coercion routines to be as lossless as possible, though some object classes are better at retaining certain kinds of information than others. Note also that there are some reserved column names in one or more object classes, which could otherwise lead to some unexpected results.

Usage

```
as_edgelist(object, twomode = FALSE)

as_matrix(object, twomode = NULL)

as_igraph(object, twomode = FALSE)

as_tidygraph(object, twomode = FALSE)

as_network(object, twomode = FALSE)

as_siena(object, twomode = FALSE)

as_graphAM(object, twomode = NULL)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
`twomode`	Logical option used to override heuristics for distinguishing incidence (two-mode/bipartite) from adjacency (one-mode/unipartite) networks. By default FALSE.

Details

Edgelists are expected to be held in data.frame or tibble class objects. The first two columns of such an object are expected to be the senders and receivers of a tie, respectively, and are typically named "from" and "to" (even in the case of an undirected network). These columns can contain integers to identify nodes or character strings/factors if the network is labelled. If the sets of senders and receivers overlap, a one-mode network is inferred. If the sets contain no overlap, a two-mode network is inferred. If a third, numeric column is present, a weighted network will be created.

Matrices can be either adjacency (one-mode) or incidence (two-mode) matrices. Incidence matrices are typically inferred from unequal dimensions, but since in rare cases a matrix with equal dimensions may still be an incidence matrix, an additional argument `twomode` can be specified to override this heuristic.

This information is usually already embedded in {igraph}, {tidygraph}, and {network} objects.

Value

The currently implemented coercions or translations are:

to/from	edgelists	matrices	igraph	tidygraph	network	siena	goldfish
edgelists (data frames)	X	X	X	X	X	X	X
matrices	X	X	X	X	X	X	X

igraph	X	X	X	X	X	X	X
tidygraph	X	X	X	X	X	X	X
network	X	X	X	X	X	X	X
graphAM	X	X	X	X	X	X	X

See Also

Other manipulations: add, grab, reformat, split(), transform()

Examples

```
test <- data.frame(from = c("A","B","B","C","C"),
                   to = c("I","G","I","G","H"))
as_edgelist(test)
as_matrix(test)
as_igraph(test)
as_tidygraph(test)
as_network(test)
as_graphAM(test)
```

---

auto_graph *Quickly graph networks with sensible defaults*

Description

The aim of this function is to provide users with a quick and easy graphing function that makes best use of the data, whatever its composition. Users can also tailor the plot according to their preferences regarding node size, colour, and shape. The function also supports visualisation of network measures such as centrality.

Usage

```
autographr(
  object,
  layout = "stress",
  labels = TRUE,
  node_color = NULL,
  node_group = NULL,
  node_shape = NULL,
  node_size = NULL,
  edge_color = NULL,
  ...
)

autographs(netlist, ...)
```

Arguments

object	A migraph-consistent object.
layout	An igraph layout algorithm, currently defaults to 'stress'.
labels	Logical, whether to print node names as labels if present.
node_color	Node variable in quotation marks to be used for coloring the nodes. It is easiest if this is added as a node attribute to the graph before plotting.
node_group	Node variable in quotation marks to be used for drawing convex but also concave hulls around clusters of nodes. These groupings will be labelled with the categories of the variable passed.
node_shape	Character string in quotation marks referring to the name of a node attribute already present in the graph to be used for the shapes of the nodes. Shapes follow the ordering "circle", "square", "triangle", so this aesthetic should be used for a variable with only a few categories.
node_size	Node variable in quotation marks to be used for the size of the nodes. This can be any continuous variable on the nodes of the network. Since this function expects this to be an existing variable, it is recommended to calculate all node-related statistics prior to using this function.
edge_color	Tie variable in quotation marks to be used for coloring the nodes. It is easiest if this is added as an edge or tie attribute to the graph before plotting.
...	Extra arguments to pass on to `autographr()`/`ggraph()`/`ggplot()`.
netlist	A list of migraph-compatible networks.

Functions

- `autographr()`: Graphs a network with sensible defaults
- `autographs()`: Graphs a list of networks with sensible defaults

See Also

Other mapping: `grid_layouts`, `partition_layouts`

Examples

```
ison_adolescents %>%
  mutate(shape = rep(c("circle", "square"), times = 4)) %>%
  mutate(color = rep(c("blue", "red"), times = 4)) %>%
  autographr(node_shape = "shape", node_color = "color")
autographr(ison_karateka, node_size = 8)
ison_adolescents %>%
  mutate(high_degree = node_is_max(node_degree())) %>%
  activate(edges) %>%
  mutate(high_betweenness = tie_is_max(tie_betweenness(ison_adolescents))) %>%
  autographr(node_color = "high_degree", edge_color = "high_betweenness")
autographr(mpn_elite_usa_advice, "concentric")
autographs(to_egos(ison_adolescents))
```

brokerage_census *Censuses of brokerage motifs*

Description

Censuses of brokerage motifs

Usage

```
node_brokerage_census(object, membership, standardized = FALSE)

network_brokerage_census(object, membership, standardized = FALSE)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
membership	A vector of partition membership as integers.
standardized	Whether the score should be standardized into a *z*-score indicating how many standard deviations above or below the average the score lies.

Functions

- node_brokerage_census(): Returns the Gould-Fernandez brokerage roles played by nodes in a network.
- network_brokerage_census(): Returns the Gould-Fernandez brokerage roles in a network.

References

Gould, R.V. and Fernandez, R.M. 1989. "Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks." *Sociological Methodology*, 19: 89-126.

See Also

Other motifs: network_census, node_census

Examples

```
node_brokerage_census(ison_networkers, "Discipline")
network_brokerage_census(ison_networkers, "Discipline")
```

centralisation *Measures of network centralisation*

Description

Measures of network centralisation

Usage

```
network_degree(object, normalized = TRUE, direction = c("all", "out", "in"))

network_closeness(object, normalized = TRUE, direction = c("all", "out", "in"))

network_betweenness(
  object,
  normalized = TRUE,
  direction = c("all", "out", "in")
)

network_eigenvector(object, normalized = TRUE)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
normalized	Logical scalar, whether the centrality scores are normalized. Different denominators are used depending on whether the object is one-mode or two-mode, the type of centrality, and other arguments.
direction	Character string, “out” bases the measure on outgoing ties, “in” on incoming ties, and "all" on either/the sum of the two. For two-mode networks, "all" uses as numerator the sum of differences between the maximum centrality score for the mode against all other centrality scores in the network, whereas "in" uses as numerator the sum of differences between the maximum centrality score for the mode against only the centrality scores of the other nodes in that mode.

Functions

- network_degree(): Calculate the degree centralization for a graph
- network_closeness(): Calculate the closeness centralization for a graph
- network_betweenness(): Calculate the betweenness centralization for a graph
- network_eigenvector(): Calculate the eigenvector centralization for a graph

See Also

Other measures: centrality, closure, cohesion(), diversity, features, holes, tie_centrality

Examples

```
network_degree(ison_southern_women, direction = "in")
network_closeness(ison_southern_women, direction = "in")
network_betweenness(ison_southern_women, direction = "in")
network_eigenvector(mpn_elite_mex)
network_eigenvector(ison_southern_women)
```

---

centrality *Measures of node centrality*

Description

These functions calculate common centrality measures for one- and two-mode networks. All measures attempt to use as much information as they are offered, including whether the networks are directed, weighted, or multimodal. If this would produce unintended results, first transform the salient properties using e.g. to_undirected() functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

```
node_degree(
  object,
  normalized = TRUE,
  alpha = 0,
  direction = c("all", "out", "in")
)

node_closeness(object, normalized = TRUE, direction = "out", cutoff = NULL)

node_betweenness(object, normalized = TRUE, cutoff = NULL)

node_eigenvector(object, normalized = TRUE, scale = FALSE)

node_reach(object, normalized = TRUE, k = 2)

node_power(object, normalized = TRUE, scale = FALSE, exponent = 1)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package

- network, from the {network} package
- tbl_graph, from the {tidygraph} package

normalized	Logical scalar, whether the centrality scores are normalized. Different denominators are used depending on whether the object is one-mode or two-mode, the type of centrality, and other arguments.
alpha	Numeric scalar, the positive tuning parameter introduced in Opsahl et al (2010) for trading off between degree and strength centrality measures. By default, alpha = 0, which ignores tie weights and the measure is solely based upon degree (the number of ties). alpha = 1 ignores the number of ties and provides the sum of the tie weights as strength centrality. Values between 0 and 1 reflect different trade-offs in the relative contributions of degree and strength to the final outcome, with 0.5 as the middle ground. Values above 1 penalise for the number of ties. Of two nodes with the same sum of tie weights, the node with fewer ties will obtain the higher score. This argument is ignored except in the case of a weighted network.
direction	Character string, "out" bases the measure on outgoing ties, "in" on incoming ties, and "all" on either/the sum of the two. For two-mode networks, "all" uses as numerator the sum of differences between the maximum centrality score for the mode against all other centrality scores in the network, whereas "in" uses as numerator the sum of differences between the maximum centrality score for the mode against only the centrality scores of the other nodes in that mode.
cutoff	Maximum path length to use during calculations.
scale	Logical scalar, whether to rescale the vector so the maximum score is 1.
k	Integer of steps out to calculate reach
exponent	Decay rate for the Bonacich power centrality score.

Value

A single centralization score if the object was one-mode, and two centralization scores if the object was two-mode.

Depending on how and what kind of an object is passed to the function, the function will return a tidygraph object where the nodes have been updated

A numeric vector giving the betweenness centrality measure of each node.

A numeric vector giving the eigenvector centrality measure of each node.

A numeric vector giving each node's power centrality measure.

Functions

- node_degree(): Calculates the degree centrality of nodes in an unweighted network, or weighted degree/strength of nodes in a weighted network.
- node_closeness(): Calculate the closeness centrality of nodes in a network
- node_betweenness(): Calculate the betweenness centralities of nodes in a network
- node_eigenvector(): Calculate the eigenvector centrality of nodes in a network
- node_reach(): Calculate nodes' reach centrality or how many nodes they can reach within *k* steps
- node_power(): Calculate the power centrality of nodes in a network

References

Faust, Katherine. 1997. "Centrality in affiliation networks." *Social Networks* 19(2): 157-191. doi:10.1016/S03788733(96)003000.

Borgatti, Stephen P., and Martin G. Everett. 1997. "Network analysis of 2-mode data." *Social Networks* 19(3): 243-270. doi:10.1016/S03788733(96)003012.

Borgatti, Stephen P., and Daniel S. Halgin. 2011. "Analyzing affiliation networks." In *The SAGE Handbook of Social Network Analysis*, edited by John Scott and Peter J. Carrington, 417–33. London, UK: Sage. doi:10.4135/9781446294413.n28.

Opsahl, Tore, Filip Agneessens, and John Skvoretz. 2010. "Node centrality in weighted networks: Generalizing degree and shortest paths." *Social Networks* 32, 245-251. doi:10.1016/j.socnet.2010.03.006

Bonacich, Phillip. 1991. "Simultaneous Group and Individual Centralities." *Social Networks* 13(2):155–68. doi:10.1016/03788733(91)90018O.

Bonacich, Phillip. 1987. "Power and Centrality: A Family of Measures." *The American Journal of Sociology* 92(5): 1170–82. doi:10.1086/228631.

See Also

`to_undirected()` for removing edge directions and `to_unweighted()` for removing weights from a graph.

Other measures: `centralisation`, `closure`, `cohesion()`, `diversity`, `features`, `holes`, `tie_centrality`

Examples

```
node_degree(mpn_elite_mex)
node_degree(ison_southern_women)
node_closeness(mpn_elite_mex)
node_closeness(ison_southern_women)
node_betweenness(mpn_elite_mex)
node_betweenness(ison_southern_women)
node_eigenvector(mpn_elite_mex)
node_eigenvector(ison_southern_women)
node_reach(ison_adolescents)
node_power(ison_southern_women, exponent = 0.5)
```

---

closure *Measures of network closure*

Description

These functions offer methods for summarising the closure in configurations in one-, two-, and three-mode networks.

Usage

```
network_reciprocity(object, method = "default")

node_reciprocity(object)

network_transitivity(object)

node_transitivity(object)

network_equivalency(object)

network_congruency(object, object2)
```

Arguments

object	A one-mode or two-mode matrix, igraph, or tidygraph
method	For reciprocity, either default or ratio. See ?igraph::reciprocity
object2	Optionally, a second (two-mode) matrix, igraph, or tidygraph

Details

For one-mode networks, shallow wrappers of igraph versions exist via network_reciprocity and network_transitivity.

For two-mode networks, network_equivalency calculates the proportion of three-paths in the network that are closed by fourth tie to establish a "shared four-cycle" structure.

For three-mode networks, network_congruency calculates the proportion of three-paths spanning two two-mode networks that are closed by a fourth tie to establish a "congruent four-cycle" structure.

Functions

- network_reciprocity(): Calculate reciprocity in a (usually directed) network
- node_reciprocity(): Calculate nodes' reciprocity
- network_transitivity(): Calculate transitivity in a network
- node_transitivity(): Calculate nodes' transitivity
- network_equivalency(): Calculate equivalence or reinforcement in a (usually two-mode) network
- network_congruency(): Calculate congruency across two two-mode networks

References

Robins, Garry L, and Malcolm Alexander. 2004. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory* 10(1): 69–94. doi:10.1023/B:CMOT.0000032580.12184.c0.

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press. doi:10.1017/9781108985000

See Also

Other measures: centralisation, centrality, cohesion(), diversity, features, holes, tie_centrality

Examples

```
network_reciprocity(ison_southern_women)
node_reciprocity(to_unweighted(ison_networkers))
network_transitivity(ison_adolescents)
node_transitivity(ison_adolescents)
network_equivalency(ison_southern_women)
```

---

cluster *Methods for equivalence clustering*

Description

These functions are used to cluster some census object. They are not intended to be called directly, but are called within node_equivalence() and related functions. They are exported and listed here to provide more detailed documentation.

Usage

```
cluster_hierarchical(census, distance)

cluster_concor(object, census)
```

Arguments

census	A matrix returned by a node_*_census() function.
distance	Character string indicating which distance metric to pass on to stats::dist. By default "euclidean", but other options include "maximum", "manhattan", "canberra", "binary", and "minkowski". Fewer, identifiable letters, e.g. "e" for Euclidean, is sufficient.
object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package

Functions

- cluster_hierarchical(): Returns a hierarchical clustering object created by stats::hclust()
- cluster_concor(): Returns a hierarchical clustering object created from a convergence of correlations procedure (CONCOR)

CONCOR

First a matrix of Pearson correlation coefficients between each pair of nodes profiles in the given census is created. Then, again, we find the correlations of this square, symmetric matrix, and continue to do this iteratively until each entry is either `1` or `-1`. These values are used to split the data into two partitions, with members either holding the values `1` or `-1`. This procedure from census to convergence is then repeated within each block, allowing further partitions to be found. Unlike UCINET, partitions are continued until there are single members in each partition. Then a distance matrix is constructed from records of in which partition phase nodes were separated, and this is given to `stats::hclust()` so that dendrograms etc can be returned.

References

Breiger, Ronald L., Scott A. Boorman, and Phipps Arabie. 1975. "An Algorithm for Clustering Relational Data with Applications to Social Network Analysis and Comparison with Multidimensional Scaling". *Journal of Mathematical Psychology*, 12: 328-83. doi:10.1016/00222496(75)900280.

`cohesion` *Measures of network cohesion or connectedness*

Description

These functions return values or vectors relating to how connected a network is and the number of nodes or edges to remove that would increase fragmentation.

Usage

```
network_density(object)

network_components(object)

network_cohesion(object)

network_adhesion(object)

network_diameter(object)

network_length(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package

Functions

- `network_density()`: summarises the ratio of ties to the number of possible ties.
- `network_components()`: Returns number of (strong) components in the network. To get the 'weak' components of a directed graph, please use `to_undirected()` first.
- `network_cohesion()`: Returns the minimum number of nodes to remove from the network needed to increase the number of components.
- `network_adhesion()`: Returns the minimum number of edges needed to remove from the network to increase the number of components.
- `network_diameter()`: Returns the maximum path length in the network.
- `network_length()`: Returns the average path length in the network.

References

White, Douglas R and Frank Harary. 2001. "The Cohesiveness of Blocks In Social Networks: Node Connectivity and Conditional Density." *Sociological Methodology* 31(1): 305-59.

See Also

Other measures: `centralisation`, `centrality`, `closure`, `diversity`, `features`, `holes`, `tie_centrality`

Examples

```
network_density(mpn_elite_mex)
network_density(mpn_elite_usa_advice)
network_cohesion(ison_marvel_relationships)
network_cohesion(to_giant(ison_marvel_relationships))
network_adhesion(ison_marvel_relationships)
network_adhesion(to_giant(ison_marvel_relationships))
network_diameter(ison_marvel_relationships)
network_diameter(to_giant(ison_marvel_relationships))
network_length(ison_marvel_relationships)
network_length(to_giant(ison_marvel_relationships))
```

`community` *Community graph partitioning algorithms*

Description

Community graph partitioning algorithms

Usage

```
node_kernighanlin(object)

node_walktrap(object, times = 50)

node_edge_betweenness(object)

node_fast_greedy(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package
`times`	Integer indicating number of simulations/walks used. By default, `times=50`.

Functions

- `node_kernighanlin()`: A greedy, iterative, deterministic graph partitioning algorithm that results in a graph with two equally-sized communities
- `node_walktrap()`: A hierarchical, agglomerative algorithm based on random walks.
- `node_edge_betweenness()`: A hierarchical, decomposition algorithm where edges are removed in decreasing order of the number of shortest paths passing through the edge, resulting in a hierarchical representation of group membership.
- `node_fast_greedy()`: A hierarchical, agglomerative algorithm, that tries to optimize modularity in a greedy manner.

Walktrap

The general idea is that random walks on a network are more likely to stay within the same community because few edges lead outside a community. By repeating random walks of 4 steps many times, information about the hierarchical merging of communities is collected.

Edge-betweenness

This is motivated by the idea that edges connecting different groups are more likely to lie on multiple shortest paths when they are the only option to go from one group to another. This method yields good results but is very slow because of the computational complexity of edge-betweenness calculations and the betweenness scores have to be re-calculated after every edge removal. Networks of ~700 nodes and ~3500 ties are around the upper size limit that are feasible with this approach.

Fast-greedy

Initially, each node is assigned a separate community. Communities are then merged iteratively such that each merge yields the largest increase in the current value of modularity, until no further increases to the modularity are possible. The method is fast and recommended as a first approximation because it has no parameters to tune. However, it is known to suffer from a resolution limit.

References

Kernighan, Brian W., and Shen Lin. 1970. "An efficient heuristic procedure for partitioning graphs." *The Bell System Technical Journal* 49(2): 291-307. doi:10.1002/j.15387305.1970.tb01770.x

See Also

Other memberships: components(), core-periphery, equivalence

Examples

```
node_kernighanlin(ison_adolescents)
node_kernighanlin(ison_southern_women)
node_walktrap(ison_adolescents)
node_edge_betweenness(ison_adolescents)
node_fast_greedy(ison_adolescents)
```

---

components *Component partitioning algorithms*

Description

These functions create a vector of nodes' memberships in components or degrees of coreness.

In graph theory, components, sometimes called connected components, are induced subgraphs from partitioning the nodes into disjoint sets. All nodes that are members of the same partition as *i* are reachable from *i*.

For directed networks, strongly connected components consist of subgraphs where there are paths in each direction between member nodes. Weakly connected components consist of subgraphs where there is a path in either direction between member nodes.

Coreness captures the maximal subgraphs in which each vertex has at least degree *k*, where *k* is also the order of the subgraph. As described in `igraph::coreness`, a node's coreness is *k* if it belongs to the *k*-core but not to the (*k*+1)-core.

Usage

```
node_components(object)

node_weak_components(object)

node_strong_components(object)

node_coreness(object)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package

Functions

- `node_components()`: Returns nodes' component membership using edge direction where available.
- `node_weak_components()`: Returns nodes' component membership ignoring edge direction.
- `node_strong_components()`: Returns nodes' component membership based on edge direction.
- `node_coreness()`: Returns k-cores

See Also

Other memberships: `community`, `core-periphery`, `equivalence`

Examples

```
node_components(mpn_bristol)
node_coreness(ison_adolescents)
```

`core-periphery` *Core-periphery clustering algorithms*

Description

This function is used to identify which nodes should belong to the core, and which to the periphery. It seeks to minimize the following quantity:

$$Z(S_1) = \sum_{(i<j)\in S_1} \mathbf{I}_{\{A_{ij}=0\}} + \sum_{(i<j)\notin S_1} \mathbf{I}_{\{A_{ij}=1\}}$$

where nodes $\{i, j, ..., n\}$ are ordered in descending degree, A is the adjacency matrix, and the indicator function is 1 if the predicate is true or 0 otherwise. Note that minimising this quantity maximises density in the core block and minimises density in the periphery block; it ignores ties between these blocks.

Usage

```
node_core(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package

References

Borgatti, Stephen P., & Everett, Martin G. 1999. Models of core /periphery structures. *Social Networks*, 21, 375–395. doi:10.1016/S03788733(99)000192

Lip, Sean Z. W. 2011. "A Fast Algorithm for the Discrete Core/Periphery Bipartitioning Problem." doi:10.48550/arXiv.1102.5511

See Also

Other memberships: `community`, `components()`, `equivalence`

Examples

```
mpn_elite_usa_advice %>% as_tidygraph %>%
  mutate(corep = node_core(mpn_elite_usa_advice)) %>%
  autographr(node_color = "corep")
network_core(mpn_elite_usa_advice)
```

---

create *Make networks with defined structures*

Description

These functions create networks with particular structural properties. They can create either one-mode or two-mode networks. To create a one-mode network, pass the main argument `n` a single integer, indicating the number of nodes in the network. To create a two-mode network, pass `n` a vector of *two* integers, where the first integer indicates the number of nodes in the first mode, and the second integer indicates the number of nodes in the second mode. As an alternative, an existing network can be provided to `n` and the number of modes and nodes will be inferred.

By default, all networks are created as undirected. This can be overruled with the argument `directed = TRUE`. This will return a directed network in which the arcs are out-facing or equivalent. This direction can be swapped using `to_redirected()`. In two-mode networks, this is ignored.

Usage

```
create_empty(n)

create_complete(n, directed = FALSE)

create_ring(n, width = 1, directed = FALSE, ...)

create_star(n, directed = FALSE)

create_tree(n, directed = FALSE, width = 2)

create_lattice(n, directed = FALSE)
```

```
create_components(n, membership = NULL)

create_core(n, membership = NULL)
```

Arguments

n	Given: • A single integer, e.g. n = 10, a one-mode network will be created. • A vector of two integers, e.g. n = c(5,10), a two-mode network will be created. • A migraph-compatible object, a network of the same dimensions will be created.
directed	Logical whether the graph should be directed. By default directed = FALSE. If the opposite direction is desired, use to_redirected().
width	Integer specifying the width or breadth of the ring or branches.
...	Additional arguments passed on to {igraph}.
membership	A vector of partition membership as integers. If left as NULL (the default), nodes in each mode will be assigned to two, equally sized partitions.

Value

By default an igraph object is returned, but this can be coerced into other types of objects using as_edgelist(), as_matrix(), as_tidygraph(), or as_network().

Functions

- create_empty(): Creates an empty graph of the given dimensions.
- create_complete(): Creates a filled graph of the given dimensions, with every possible tie realised.
- create_ring(): Creates a ring or chord graph of the given dimensions that loops around is of a certain width or thickness.
- create_star(): Creates a graph of the given dimensions that has a maximally central node
- create_tree(): Creates a graph of the given dimensions with successive branches.
- create_lattice(): Creates a graph of the given dimensions with ties to all neighbouring nodes
- create_components(): Creates a graph in which the nodes are clustered into separate components.
- create_core(): Creates a graph with a certain proportion of nodes being core nodes, densely tied to each other and peripheral nodes, and the rest peripheral, tied only to the core.

See Also

as

Other makes: generate, read

Examples

```
autographr(create_empty(10)) + autographr(create_complete(10))
autographr(create_empty(c(8,6))) + autographr(create_complete(c(8,6)))
autographr(create_ring(8, width = 2)) +
autographr(create_ring(c(8,6), width = 2))
autographr(create_star(12)) +
autographr(create_star(12, directed = TRUE)) +
autographr(create_star(c(12,1)))
autographr(create_tree(c(7,8), directed = TRUE)) +
autographr(create_tree(15, directed = TRUE), "tree") +
autographr(create_tree(15, directed = TRUE, width = 3), "tree")
autographr(create_lattice(5), layout = "kk") +
autographr(create_lattice(c(5,5)))
autographr(create_components(10, membership = c(1,1,1,2,2,2,3,3,3,3)))
autographr(create_components(c(10, 12)))
autographr(create_core(6)) +
autographr(create_core(6, membership = c(1,1,1,1,2,2))) +
autographr(create_core(c(6,6)))
```

---

diversity *Measures of network diversity*

Description

These functions offer ways to summarise the heterogeneity of an attribute across a network, within groups of a network, or the distribution of ties across this attribute.

Usage

```
network_richness(object, attribute)

node_richness(object, attribute)

network_diversity(object, attribute, clusters = NULL)

node_diversity(object, attribute)

network_homophily(object, attribute)

node_homophily(object, attribute)

network_assortativity(object)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble}

- igraph, from the {igraph} package
- network, from the {network} package
- tbl_graph, from the {tidygraph} package

`attribute` Name of a nodal attribute or membership vector to use as categories for the diversity measure.

`clusters` A nodal cluster membership vector or name of a vertex attribute.

Functions

- `network_richness()`: Calculates the number of unique categories in a network attribute.
- `node_richness()`: Calculates the number of unique categories of an attribute to which each node is connected.
- `network_diversity()`: Calculates the heterogeneity of ties across a network or within clusters by node attributes.
- `node_diversity()`: Calculates the heterogeneity of each node's local neighbourhood.
- `network_homophily()`: Calculates how embedded nodes in the network are within groups of nodes with the same attribute
- `node_homophily()`: Calculates each node's embeddedness within groups of nodes with the same attribute
- `network_assortativity()`: Calculates the degree assortativity in a graph.

network_diversity

Blau's index (1977) uses a formula known also in other disciplines by other names (Gini-Simpson Index, Gini impurity, Gini's diversity index, Gibbs-Martin index, and probability of interspecific encounter (PIE)):

$$1 - \sum_{i=1}^{k} p_i^2$$

, where p_i is the proportion of group members in ith category and k is the number of categories for an attribute of interest. This index can be interpreted as the probability that two members randomly selected from a group would be from different categories. This index finds its minimum value (0) when there is no variety, i.e. when all individuals are classified in the same category. The maximum value depends on the number of categories and whether nodes can be evenly distributed across categories.

network_homophily

Given a partition of a network into a number of mutually exclusive groups then The E-I index is the number of ties between (or *external*) nodes grouped in some mutually exclusive categories minus the number of ties within (or *internal*) these groups divided by the total number of ties. This value can range from 1 to -1, where 1 indicates ties only between categories/groups and -1 ties only within categories/groups.

References

Blau, Peter M. (1977). *Inequality and heterogeneity*. New York: Free Press.

Krackhardt, David and Robert N. Stern (1988). Informal networks and organizational crises: an experimental simulation. *Social Psychology Quarterly* 51(2), 123-140.

See Also

Other measures: centralisation, centrality, closure, cohesion(), features, holes, tie_centrality

Examples

```
marvel_friends <- to_unsigned(ison_marvel_relationships, "positive")
network_diversity(marvel_friends, "Gender")
network_diversity(marvel_friends, "Attractive")
network_diversity(marvel_friends, "Gender", "Rich")
node_diversity(marvel_friends, "Gender")
node_diversity(marvel_friends, "Attractive")
network_homophily(marvel_friends, "Gender")
network_homophily(marvel_friends, "Attractive")
node_homophily(marvel_friends, "Gender")
node_homophily(marvel_friends, "Attractive")
network_assortativity(mpn_elite_mex)
```

---

equivalence *Equivalence clustering algorithms*

Description

These functions combine an appropriate _census() function together with methods for calculating the hierarchical clusters provided by a certain distance calculation.

A plot() method exists for investigating the dendrogram of the hierarchical cluster and showing the returned cluster assignment.

Usage

```
node_equivalence(
  object,
  census,
  k = c("silhouette", "elbow", "strict"),
  cluster = c("hierarchical", "concor"),
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  range = 8L
)

node_structural_equivalence(
  object,
  k = c("silhouette", "elbow", "strict"),
```

```
  cluster = c("hierarchical", "concor"),
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  range = 8L
)

node_regular_equivalence(
  object,
  k = c("silhouette", "elbow", "strict"),
  cluster = c("hierarchical", "concor"),
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  range = 8L
)

node_automorphic_equivalence(
  object,
  k = c("silhouette", "elbow", "strict"),
  cluster = c("hierarchical", "concor"),
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
  range = 8L
)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
census	A matrix returned by a node_*_census() function.
k	Typically a character string indicating which method should be used to select the number of clusters to return. By default "silhouette", other options include "elbow" and "strict". "strict" returns classes with members only when strictly equivalent. "silhouette" and "elbow" select classes based on the distance between clusters or between nodes within a cluster. Fewer, identifiable letters, e.g. "e" for elbow, is sufficient. Alternatively, if k is passed an integer, e.g. k = 3, then all selection routines are skipped in favour of this number of clusters.
cluster	Character string indicating whether clusters should be clustered hierarchically ("hierarchical") or through convergence of correlations ("concor"). Fewer, identifiable letters, e.g. "c" for CONCOR, is sufficient.
distance	Character string indicating which distance metric to pass on to stats::dist. By default "euclidean", but other options include "maximum", "manhattan", "canberra", "binary", and "minkowski". Fewer, identifiable letters, e.g. "e" for Euclidean, is sufficient.
range	Integer indicating the maximum number of (k) clusters to evaluate. Ignored when k = "strict" or a discrete number is given for k.

Functions

- `node_equivalence()`: Returns nodes' membership in according to their equivalence with respective to some census/class
- `node_structural_equivalence()`: Returns nodes' membership in structurally equivalent classes
- `node_regular_equivalence()`: Returns nodes' membership in regularly equivalent classes
- `node_automorphic_equivalence()`: Returns nodes' membership in automorphically equivalent classes

Source

https://github.com/aslez/concoR

References

Thorndike, Robert L. 1953. "Who Belongs in the Family?". *Psychometrika*, 18(4): 267–76. doi:10.1007/BF02289263.

Rousseeuw, Peter J. 1987. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis." *Journal of Computational and Applied Mathematics*, 20: 53–65. doi:10.1016/03770427(87)901257.

See Also

Other memberships: `community`, `components()`, `core-periphery`

Examples

```
(nse <- node_structural_equivalence(mpn_elite_usa_advice))
plot(nse)


(nre <- node_regular_equivalence(mpn_elite_usa_advice,
  cluster = "concor"))
plot(nre)


(nae <- node_automorphic_equivalence(mpn_elite_usa_advice,
  k = "elbow"))
plot(nae)
```

`features` *Measures of network topological features*

Description

Measures of network topological features

Usage

```
network_core(object, membership = NULL)

network_factions(object, membership = NULL)

network_modularity(object, membership = NULL, resolution = 1)

network_smallworld(object, method = c("omega", "sigma", "SWI"), times = 100)

network_scalefree(object)

network_balance(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
`membership`	A vector of partition membership.
`resolution`	A proportion indicating the resolution scale. By default 1.
`method`	There are three small-world measures implemented: • "sigma" is the original equation from Watts and Strogatz (1998), $$\frac{\frac{C}{C_r}}{\frac{L}{L_r}}$$, where C and L are the observed clustering coefficient and path length, respectively, and C_r and L_r are the averages obtained from random networks of the same dimensions and density. A $\sigma > 1$ is considered to be small-world, but this measure is highly sensitive to network size. • "omega" (the default) is an update from Telesford et al. (2011), $$\frac{L_r}{L} - \frac{C}{C_l}$$

, where C_l is the clustering coefficient for a lattice graph with the same dimensions. ω ranges between 0 and 1, where 1 is as close to a small-world as possible.

- "SWI" is an alternative proposed by Neal (2017),

$$\frac{L - L_l}{L_r - L_l} \times \frac{C - C_r}{C_l - C_r}$$

, where L_l is the average path length for a lattice graph with the same dimensions. SWI also ranges between 0 and 1 with the same interpretation, but where there may not be a network for which $SWI = 1$.

`times` Integer of number of simulations.

Functions

- `network_core()`: Returns correlation between a given network and a core-periphery model with the same dimensions.
- `network_factions()`: Returns correlation between a given network and a component model with the same dimensions.
- `network_modularity()`: Returns modularity based on nodes' membership in pre-defined clusters.
- `network_smallworld()`: Returns small-world metrics for one- and two-mode networks. Small-world networks can be highly clustered and yet have short path lengths.
- `network_scalefree()`: Returns the exponent of the fitted power-law distribution. Usually an exponent between 2 and 3 indicates a power-law distribution.
- `network_balance()`: Returns the structural balance index on the proportion of balanced triangles, ranging between `0` if all triangles are imbalanced and `1` if all triangles are balanced.

Modularity

Modularity measures the difference between the number of ties within each community from the number of ties expected within each community in a random graph with the same degrees, and ranges between -1 and +1. Modularity scores of +1 mean that ties only appear within communities, while -1 would mean that ties only appear between communities. A score of 0 would mean that ties are half within and half between communities, as one would expect in a random graph.

Modularity faces a difficult problem known as the resolution limit (Fortunato and Barthélemy 2007). This problem appears when optimising modularity, particularly with large networks or depending on the degree of interconnectedness, can miss small clusters that 'hide' inside larger clusters. In the extreme case, this can be where they are only connected to the rest of the network through a single tie.

Source

`{signnet}` by David Schoch

References

Borgatti, Stephen P., and Martin G. Everett. 2000. "Models of Core/Periphery Structures." *Social Networks* 21(4):375–95. doi:10.1016/S03788733(99)000192

Murata, Tsuyoshi. 2010. Modularity for Bipartite Networks. In: Memon, N., Xu, J., Hicks, D., Chen, H. (eds) *Data Mining for Social Network Data. Annals of Information Systems*, Vol 12. Springer, Boston, MA. doi:10.1007/9781441962874_7

Watts, Duncan J., and Steven H. Strogatz. 1998. "Collective Dynamics of 'Small-World' Networks." *Nature* 393(6684):440–42. doi:10.1038/30918.

Telesford QK, Joyce KE, Hayasaka S, Burdette JH, Laurienti PJ. 2011. "The ubiquity of small-world networks". *Brain Connectivity* 1(5): 367–75. doi:10.1089/brain.2011.0038.

Neal Zachary P. 2017. "How small is it? Comparing indices of small worldliness". *Network Science*. 5 (1): 30–44. doi:10.1017/nws.2017.5.

See Also

`network_transitivity()` and `network_equivalency()` for how clustering is calculated

Other measures: `centralisation`, `centrality`, `closure`, `cohesion()`, `diversity`, `holes`, `tie_centrality`

Examples

```
network_core(ison_adolescents)
network_core(ison_southern_women)
network_factions(ison_adolescents)
network_factions(ison_southern_women)
network_modularity(ison_adolescents,
  node_kernighanlin(ison_adolescents))
network_modularity(ison_southern_women,
  node_kernighanlin(ison_southern_women))
network_smallworld(ison_brandes)
network_smallworld(ison_southern_women)
network_scalefree(ison_adolescents)
network_scalefree(generate_scalefree(50, 1.5))
network_scalefree(create_lattice(100))
network_balance(ison_marvel_relationships)
```

---

generate *Make networks with a stochastic element*

Description

These functions are similar to the `create_*` functions, but include some element of randomisation. They are particularly useful for creating a distribution of networks for exploring or testing network properties.

Usage

```
generate_random(n, p = 0.5, directed = FALSE, with_attr = TRUE)

generate_smallworld(n, p = 0.05, width = 2, directed = FALSE)

generate_scalefree(n, p = 1, directed = FALSE)

generate_permutation(object, with_attr = TRUE)
```

Arguments

n	Given: • A single integer, e.g. n = 10, a one-mode network will be created. • A vector of two integers, e.g. n = c(5,10), a two-mode network will be created. • A migraph-compatible object, a network of the same dimensions will be created.
p	Proportion of possible ties in the network that are realised or, if integer greater than 1, the number of ties in the network.
directed	Whether to generate network as directed. By default FALSE.
with_attr	Logical whether any attributes of the object should be retained. By default TRUE.
width	Integer specifying the width or breadth of the ring or branches.
object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package

Value

By default an igraph object is returned, but this can be coerced into other types of objects using as_matrix(), as_tidygraph(), or as_network().

Functions

- generate_random(): Generates a random network with a particular probability.
- generate_smallworld(): Generates a small-world structure following the lattice rewiring model.
- generate_scalefree(): Generates a scale-free structure following the preferential attachment model.
- generate_permutation(): Generates a permutation of the original network using a Fisher-Yates shuffle on both the rows and columns (for a one-mode network) or on each of the rows and columns (for a two-mode network).

References

Erdős, Paul, and Alfréd Rényi. (1959). "On Random Graphs I" *Publicationes Mathematicae*. 6: 290–297.

Watts, Duncan J., and Steven H. Strogatz. 1998. "Collective Dynamics of 'Small-World' Networks." *Nature* 393(6684):440–42. doi:10.1038/30918.

Barabási, Albert-László, and Réka Albert. 1999. "Emergence of Scaling in Random Networks." *Science* 286(5439):509–12. doi:10.1126/science.286.5439.509.

See Also

as

Other makes: `create`, `read`

Examples

```
autographr(generate_random(12, 0.4)) +
autographr(generate_random(c(6, 6), 0.4))
autographr(generate_smallworld(12, 0.025)) +
autographr(generate_smallworld(12, 0.25)) +
autographr(generate_smallworld(c(6,6), 0.025))
autographr(generate_scalefree(12, 0.25)) +
autographr(generate_scalefree(12, 1.25))
autographr(generate_scalefree(c(12,6), 0.25)) /
autographr(generate_scalefree(c(12,6), 1.25))
autographr(mpn_elite_usa_advice) +
autographr(generate_permutation(mpn_elite_usa_advice))
```

---

ggevolution *Plot the evolution of a network*

Description

This function offers a method to plot a network at two or more timepoints for quick and easy comparison. The function is currently limited to two networks and only the layout given by the first or last network, but further extensions expected.

Usage

```
ggevolution(..., layout = "kk", based_on = c("first", "last", "both"))
```

Arguments

`...`	two or more networks
`layout`	an igraph layout. Default is Kamada-Kawai ("kk")
`based_on`	whether the layout of the joint plots should be based on the "first" or the "last" network.

Examples

```
mpn_elite_mex <- mpn_elite_mex %>% to_subgraph(in_mpn == 1)
mpn_elite_mex2 <- generate_permutation(mpn_elite_mex)
ggevolution(mpn_elite_mex, mpn_elite_mex2)
ggevolution(mpn_elite_mex, mpn_elite_mex2, based_on = "last")
ggevolution(mpn_elite_mex, mpn_elite_mex2, based_on = "both")
```

---

gglineage *Plot lineage graph*

Description

Lineage implies a direct descent from an ancestor; ancestry or pedigree. That is, how observation derives and is connected to previous observations. The function plots a lineage graph of citations, amendments, and more, for example.

Usage

```
gglineage(object, labels = TRUE)
```

Arguments

object	A migraph-consistent network/graph.
labels	Whether to plot node labels or not. Default: TRUE.

Examples

```
cites <- dplyr::tibble(qID1 = c("BNLHPB_2016P:BNLHPB_1970A",
"PARIS_2015A","INOOTO_2015A", "RUS-USA[UUF]_2015A",
"RUS-USA[UUF]_2015A", "RUS-USA[UUF]_2015A", "RUS-USA[UUF]_2015A",
"INECHA_2015O", "ST04DC_2014P", "ST04DC_2014P"),
qID2 = c("BNLHPB_1977P:BNLHPB_1970A", "UNFCCC_1992A", "INOOTO_2005A",
"RUS-USA[MFR]_1988A", "PS07UF_2009A", "UNCLOS_1982A", "UNCLOS_1982A",
"ERECHA_1991O", "AI07EM_1998A", "CNEWNH_1979A"))
gglineage(cites)
```

---

grab *Grab various node or edge attributes from a network*

Description

These functions operate to help extract certain attributes from given network data. They are also useful as helpers within other functions.
`network_*()` functions always relate to the overall graph or network, usually returning a scalar. `node_*()` and `tie_*()` always return vectors the same length as the number of nodes or edges in the network, respectively.

Usage

```
node_names(object)

node_mode(object)

node_attribute(object, attribute)

tie_attribute(object, attribute)

tie_weights(object)

tie_signs(object)

network_nodes(object)

network_ties(object)

network_dims(object)

network_node_attributes(object)

network_tie_attributes(object)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
attribute	Character string naming an attribute in the object.

Functions

- node_names(): Extracts the names of the nodes in a network.
- node_mode(): Extracts the mode of the nodes in a network.
- node_attribute(): Extracts an attribute's values for the nodes in a network.
- tie_attribute(): Extracts an attribute's values for the edges in a network.
- tie_weights(): Extracts the weights of the edges in a network.
- tie_signs(): Extracts the signs of the edges in a network.
- network_nodes(): Returns the total number of nodes (of any mode) in a network.
- network_ties(): Returns the number of edges in a network.
- network_dims(): Returns the dimensions of a network in a vector as long as the number of modes in the network.

- network_node_attributes(): Returns a vector of nodal attributes in a network
- network_tie_attributes(): Returns a vector of edge attributes in a network

See Also

Other manipulations: add, as(), reformat, split(), transform()

Examples

```
node_names(mpn_elite_usa_advice)
node_mode(mpn_elite_usa_advice)
node_attribute(mpn_elite_mex, "full_name")
tie_attribute(ison_algebra, "task_tie")
tie_weights(to_mode1(ison_southern_women))
tie_signs(ison_marvel_relationships)
network_nodes(ison_southern_women)
network_ties(ison_southern_women)
network_dims(ison_southern_women)
network_dims(to_mode1(ison_southern_women))
network_node_attributes(mpn_elite_mex)
network_tie_attributes(mpn_elite_mex)
```

---

grid_layouts *Layouts for snapping layouts to a grid*

Description

The function uses approximate pattern matching to redistribute coarse layouts on square grid points, while preserving the topological relationships among the nodes (see Inoue et al. 2012).

Usage

```
layout_tbl_graph_frgrid(object, circular = FALSE, times = 1000)

layout_tbl_graph_kkgrid(object, circular = FALSE, times = 1000)

layout_tbl_graph_gogrid(object, circular = FALSE, times = 1000)

layout_tbl_graph_stressgrid(object, circular = FALSE, times = 1000)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package

- tbl_graph, from the {tidygraph} package

`circular` Should the layout be transformed into a radial representation. Only possible for some layouts. Defaults to FALSE

`times` Maximum number of iterations, where appropriate

References

Inoue, Kentaro, Shinichi Shimozono, Hideaki Yoshida, and Hiroyuki Kurata. 2012. "Application of Approximate Pattern Matching in Two Dimensional Spaces to Grid Layout for Biochemical Network Maps" edited by J. Bourdon. *PLoS ONE* 7(6):e37739. doi:10.1371/journal.pone.0037739.

See Also

Other mapping: `auto_graph`, `partition_layouts`

`holes` *Measures of structural holes*

Description

These function provide different measures of the degree to which nodes fill structural holes, as outlined in Burt (1992). Burt's theory holds that while those nodes embedded in dense clusters of close connections are likely exposed to the same or similar ideas and information, those who fill structural holes between two otherwise disconnected groups can gain some comparative advantage from that position.

Usage

```
node_bridges(object)

node_redundancy(object)

node_effsize(object)

node_efficiency(object)

node_constraint(object)

node_hierarchy(object)
```

Arguments

`object` An object of a migraph-consistent class:

- matrix (adjacency or incidence) from {base} R
- edgelist, a data frame from {base} R or tibble from {tibble}
- igraph, from the {igraph} package
- network, from the {network} package
- tbl_graph, from the {tidygraph} package

Details

A number of different ways of measuring these structural holes are available. Note that we use Borgatti's reformulation for unweighted networks in `node_redundancy()` and `node_effsize()`. Redundancy is thus $\frac{2t}{n}$, where t is the sum of ties and n the sum of nodes in each node's neighbourhood, and effective size is calculated as $n - \frac{2t}{n}$. Node efficiency is the node's effective size divided by its degree.

Functions

- `node_bridges()`: Returns the sum of bridges to which each node is adjacent.
- `node_redundancy()`: Returns a measure of the redundancy of each nodes' contacts.
- `node_effsize()`: Returns nodes' effective size
- `node_efficiency()`: Returns nodes' efficiency
- `node_constraint()`: Returns nodes' constraint scores for one-mode networks according to Burt (1992) and for two-mode networks according to Hollway et al (2020).
- `node_hierarchy()`: Returns nodes' exposure to hierarchy, where only one or two contacts are the source of closure

References

Burt, Ronald S. 1992. *Structural Holes: The Social Structure of Competition*. Cambridge, MA: Harvard University Press.

Borgatti, Steven. 1997. "Structural Holes: Unpacking Burt's Redundancy Measures" *Connections* 20(1):35-38.

Hollway, James, Jean-Frédéric Morin, and Joost Pauwelyn. 2020. "Structural conditions for novelty: the introduction of new environmental clauses to the trade regime complex." *International Environmental Agreements: Politics, Law and Economics* 20 (1): 61–83. doi:10.1007/s10784019-094645.

See Also

Other measures: `centralisation`, `centrality`, `closure`, `cohesion()`, `diversity`, `features`, `tie_centrality`

Examples

```
node_bridges(ison_adolescents)
node_bridges(ison_southern_women)
node_redundancy(ison_adolescents)
node_redundancy(ison_southern_women)
node_effsize(ison_adolescents)
node_effsize(ison_southern_women)
node_efficiency(ison_adolescents)
node_efficiency(ison_southern_women)
node_constraint(ison_southern_women)
node_hierarchy(ison_adolescents)
node_hierarchy(ison_southern_women)
```

`is` *Marking networks based on their properties*

Description

These functions implement logical tests for various network properties.

Usage

```
is_migraph(object)

is_graph(object)

is_edgelist(object)

is_twomode(object)

is_weighted(object)

is_directed(object)

is_labelled(object)

is_signed(object)

is_connected(object)

is_complex(object)

is_multiplex(object)

is_uniplex(object)

is_acyclic(object)

is_aperiodic(object, max_path_length = 4)

is_perfect_matching(object, mark = "type")

is_eulerian(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}`

- igraph, from the {igraph} package
- network, from the {network} package
- tbl_graph, from the {tidygraph} package

`max_path_length`
: Maximum path length considered. If negative, paths of all lengths are considered. By default 4, to avoid potentially very long computation times.

`mark`
: A logical vector marking two types or modes. By default "type".

Value

TRUE if the condition is met, or FALSE otherwise.

Functions

- `is_migraph()`: Tests whether network is migraph-compatible
- `is_graph()`: Tests whether network contains graph-level information
- `is_edgelist()`: Tests whether data frame is an edgelist
- `is_twomode()`: Tests whether network is a two-mode network
- `is_weighted()`: Tests whether network is weighted
- `is_directed()`: Tests whether network is directed
- `is_labelled()`: Tests whether network includes names for the nodes
- `is_signed()`: Tests whether network is signed positive/negative
- `is_connected()`: Tests whether network is weakly connected if the network is undirected or strongly connected if directed. To test weak connection on a directed network, please see `to_undirected()`.
- `is_complex()`: Tests whether network contains any loops
- `is_multiplex()`: Tests whether network is multiplex, either from multiple rows with the same sender and receiver, or multiple columns to the edgelist.
- `is_uniplex()`: Tests whether network is simple (both uniplex and simplex)
- `is_acyclic()`: Tests whether network is a directed acyclic graph
- `is_aperiodic()`: Tests whether network is aperiodic
- `is_perfect_matching()`: Tests whether there is a matching for a network that covers every node in the network
- `is_eulerian()`: Tests whether there is a Eulerian path for a network where that path passes through every tie exactly once @importFrom igraph has_eulerian_path

Source

https://stackoverflow.com/questions/55091438/r-igraph-find-all-cycles

See Also

Other marks: `mark_nodes`, `mark_ties`

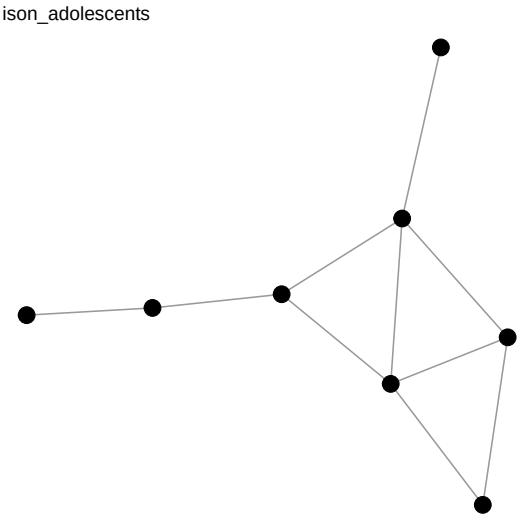
Examples

```
is_twomode(ison_southern_women)
is_weighted(ison_southern_women)
is_directed(ison_algebra)
is_labelled(ison_southern_women)
is_signed(ison_southern_women)
is_connected(ison_southern_women)
is_complex(ison_southern_women)
is_uniplex(ison_algebra)
is_acyclic(ison_algebra)
is_aperiodic(ison_algebra)
is_perfect_matching(ison_southern_women)
is_eulerian(ison_brandes)
```

---

ison_adolescents *One-mode subset (8 individuals) of the adolescent society (Coleman 1961).*

Description



Usage

```
data(ison_adolescents)
```

Format

```
#> # A tbl_graph: 8 nodes and 10 edges
#> #
#> # An undirected simple graph with 1 component
#> #
```

```
#> # Node Data: 8 x 1 (active)
#>   name
#>   <chr>
#> 1 Betty
#> 2 Sue
#> 3 Alice
#> 4 Jane
#> 5 Dale
#> 6 Pam
#> # ... with 2 more rows
#> #
#> # Edge Data: 10 x 2
#>    from    to
#>   <int> <int>
#> 1     1     2
#> 2     2     3
#> 3     3     4
#> # ... with 7 more rows
```

References

Coleman, James S. 1961. The Adolescent Society. New York: Free Press.

Feld, Scott. 1991. "Why your friends have more friends than you do" *American Journal of Sociology* 96(6): 1464-1477. doi:10.1086/229693.

`ison_algebra` *Multiplex graph object of friends, social, and task ties (McFarland 2001)*

Description

Multiplex graph object of friends, social, and task ties (McFarland 2001)

Usage

```
data(ison_algebra)
```

Format

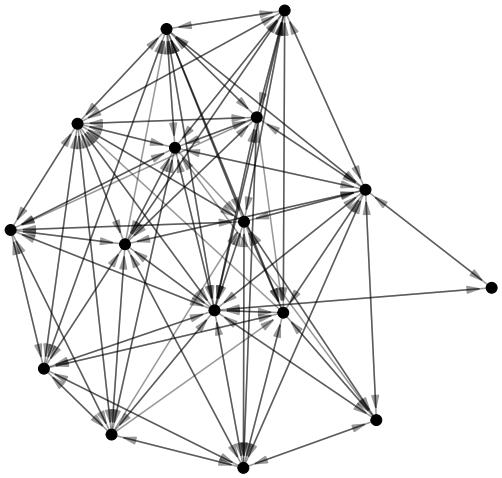
```
#> # A tbl_graph: 16 nodes and 144 edges
#> #
#> # A directed simple graph with 1 component
#> #
#> # Node Data: 16 x 1 (active)
#>   name
#>   <chr>
#> 1 Melinda
```

```
#> 2 Abby
#> 3 Darryl
#> 4 Veronica
#> 5 Rylan
#> 6 Lindsey
#> # ... with 10 more rows
#> #
#> # Edge Data: 144 x 5
#>    from    to friends social tasks
#>   <int> <int>   <dbl>  <dbl> <dbl>
#> 1     1     5       0   1.2    0.3
#> 2     1     8       0   0.15   0
#> 3     1     9       0   2.85   0.3
#> # ... with 141 more rows
```

Details

Multiplex graph object of friends, social, and task ties between 16 anonymous students. M182 was an honors algebra class where researchers collected friendship, social, and task ties between 16 students. The edge attribute `friends` contains friendship ties, where `2` = best friends, `1` = friend, and `0` is not a friend. `social` consists of social interactions per hour, and `tasks` consists of task interactions per hour.

ison_algebra



Source

See also `data(studentnets.M182, package = "NetData")` Larger comprehensive data set publicly available, contact Daniel A. McFarland for details.

References

McFarland, Daniel A. (2001) "Student Resistance." *American Journal of Sociology* 107(3): 612-78. doi:10.1086/338779.

ison_brandes *One-mode and two-mode centrality demonstration networks*

Description

This network should solely be used for demonstration purposes as it does not describe a real network.

Usage

```
data(ison_brandes)

data(ison_brandes2)
```

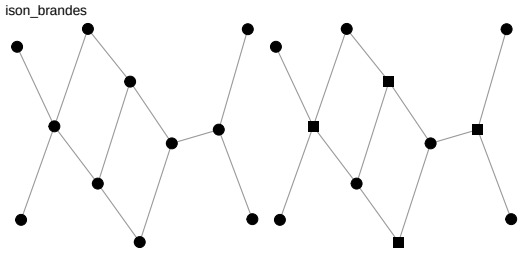
Format

```
#> # A tbl_graph: 11 nodes and 12 edges
#> #
#> # An undirected simple graph with 1 component
#> #
#> # Node Data: 11 x 0 (active)
#> # ... with 5 more rows
#> #
#> # Edge Data: 12 x 2
#>    from    to
#>   <int> <int>
#> 1     1     3
#> 2     2     3
#> 3     3     4
#> # ... with 9 more rows

#> # A tbl_graph: 11 nodes and 12 edges
#> #
#> # A bipartite simple graph with 1 component
#> #
#> # Node Data: 11 x 1 (active)
#>   type
#>   <lgl>
#> 1 FALSE
#> 2 FALSE
#> 3 TRUE
#> 4 FALSE
#> 5 TRUE
#> 6 TRUE
#> # ... with 5 more rows
#> #
#> # Edge Data: 12 x 2
```

```
#>    from    to
#>   <int> <int>
#> 1     1     3
#> 2     2     3
#> 3     3     4
#> # ... with 9 more rows
```

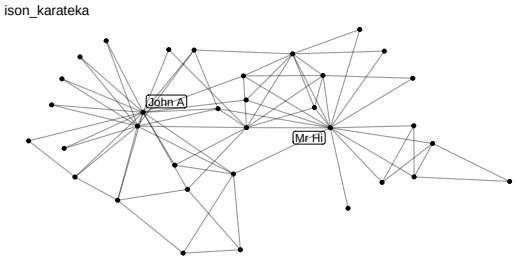
Details



`ison_karateka` *One-mode karateka network (Zachary 1977)*

Description

The network was observed in a university Karate club in 1977. The network describes association patterns among 34 members and maps out allegiance patterns between members and either Mr. Hi, the instructor, or the John A. the club president after an argument about hiking the price for lessons. The allegiance of each node is listed in the obc argument which takes the value 1 if the individual sided with Mr. Hi after the fight and 2 if the individual sided with John A.



Usage

```
data(ison_karateka)
```

Format

```
#> IGRAPH a87bb9f UN-- 34 78 --
#> + attr: name (v/c), obc (v/n)
#> + edges from a87bb9f (vertex names):
```

```
#>  [1] Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi--
#> [10] Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi-- Mr Hi--      --      --
#> [19]      --      --      --      --      --      --      --      --      --
#> [28]      --      --      --      --      --      --      --      --      --
#> [37]      --      --      --      --      --      --      --
#> + ... omitted several edges
```

References

Zachary, Wayne W. 1977. "An Information Flow Model for Conflict and Fission in Small Groups." *Journal of Anthropological Research* 33(4):452–73. doi:10.1086/jar.33.4.3629752.

`ison_lotr` *One-mode interaction network of Lord of the Rings (book) character interactions*

Description

One-mode interaction network of Lord of the Rings (book) character interactions

Usage

```
data(ison_lotr)
```

Format

One-mode tidygraph of 36 Lord of the Rings book characters and 66 interactions

---

`ison_marvel` *Multilevel two-mode affiliation, signed one-mode networks of Marvel comic book characters (Yüksel 2017)*

Description

Multilevel two-mode affiliation, signed one-mode networks of Marvel comic book characters (Yüksel 2017)

Usage

```
data(ison_marvel_teams)

data(ison_marvel_relationships)
```

Format

Two-mode igraph of 53 Marvel comic book characters and 141 team-ups, with 683 team affiliations between them

One-mode igraph of 53 Marvel comic book characters and 558 signed (`1` = friends, `-1` = enemies) undirected ties

Details

This package includes two datasets related to the Marvel *comic book* universe. The first, `ison_marvel_teams`, is a two-mode affiliation network of 53 Marvel comic book characters and their affiliations to 141 different teams. This network includes only information about nodes' names and nodeset, but additional nodal data can be taken from the other Marvel dataset here.

The second network, `ison_marvel_relationships`, is a one-mode signed network of friendships and enmities between the 53 Marvel comic book characters. Friendships are indicated by a positive sign in the edge `sign` attribute, whereas enmities are indicated by a negative sign in this edge attribute. Additional nodal variables have been coded and included by Dr Umut Yüksel:

- **Gender**: binary character, 43 "Male" and 10 "Female"
- **PowerOrigin**: binary character, 2 "Alien", 1 "Cyborg", 5 "God/Eternal", 22 "Human", 1 "Infection", 16 "Mutant", 5 "Radiation", 1 "Robot"
- **Appearances**: integer, in how many comic book issues they appeared in
- **Attractive**: binary integer, 41 1 (yes) and 12 0 (no)
- **Rich**: binary integer, 11 1 (yes) and 42 0 (no)
- **Intellect**: binary integer, 39 1 (yes) and 14 0 (no)
- **Omnilingual**: binary integer, 8 1 (yes) and 45 0 (no)
- **UnarmedCombat**: binary integer, 51 1 (yes) and 2 0 (no)
- **ArmedCombat**: binary integer, 25 1 (yes) and 28 0 (no)

Source

Umut Yüksel, 31 March 2017

`ison_networkers` *One-mode EIES dataset (Freeman and Freeman 1979)*

Description

A directed, simple, named, weighted graph with 32 nodes and 440 edges. Nodes are academics and edges illustrate the communication patterns on an Electronic Information Exchange System among them. Node attributes include the number of citations (`Citations`) and the discipline of the researchers (`Discipline`). Edge weights illustrate the number of emails sent from one academic to another over the studied time period.

Usage

```
data(ison_networkers)
```

Format

```
#> # A tbl_graph: 32 nodes and 440 edges
#> #
#> # A directed simple graph with 1 component
#> #
#> # Node Data: 32 x 3 (active)
#>   name               Discipline   Citations
#>   <chr>              <chr>            <dbl>
#> 1 LIN FREEMAN        Sociology           19
#> 2 DOUG WHITE         Anthropology         3
#> 3 EV ROGERS          Other              170
#> 4 RICHARD ALBA       Sociology           23
#> 5 PHIPPS ARABIE      Other               16
#> 6 CAROL BARNER-BARRY Other                6
#> # ... with 26 more rows
#> #
#> # Edge Data: 440 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1     2    488
#> 2     1     3     28
#> 3     1     4     65
#> # ... with 437 more rows
```

Source

networkdata package

References

Freeman, Sue C. and Linton C. Freeman. 1979. *The networkers network: A study of the impact of a new communications medium on sociometric structure*. Social Science Research Reports No 46. Irvine CA, University of California.

Wasserman Stanley and Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge.

---

ison_projection *Two-mode projection examples (Hollway 2021)*

Description

Two-mode projection examples (Hollway 2021)

Usage

```
data(ison_mm)

data(ison_bm)

data(ison_mb)

data(ison_bb)
```

Format

Directed two-mode {igraph} object with 6 nodes and 6 edges

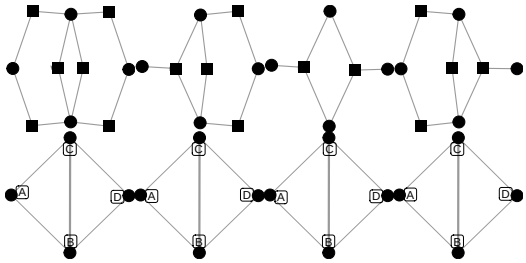
Directed two-mode {igraph} object with 8 nodes and 9 edges

Directed two-mode {igraph} object with 8 nodes and 9 edges

Directed two-mode {igraph} object with 10 nodes and 12 edges

Details

These datasets should only be used for demonstration purposes as they do not describe a real world network. All examples contain named nodes.



---

ison_southern_women *Two-mode southern women (Davis, Gardner and Gardner 1941)*

Description

Two-mode network dataset collected by Davis, Gardner and Gardner (1941) about the attendance pattern of women at informal social events during a 9 month period. Events and women are named.

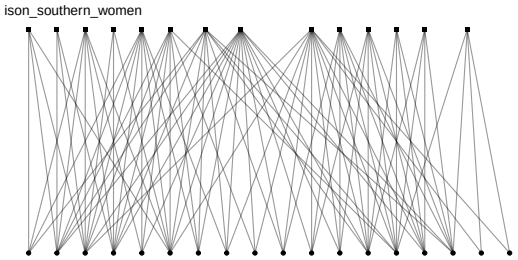
Usage

```
data(ison_southern_women)
```

Format

```
#> IGRAPH f8d9f5f UN-B 32 93 --
#> + attr: type (v/l), name (v/c)
#> + edges from f8d9f5f (vertex names):
#>  [1] EVELYN   --E1 EVELYN   --E2 EVELYN   --E3 EVELYN   --E4 EVELYN   --E5
#>  [6] EVELYN   --E6 EVELYN   --E8 EVELYN   --E9 LAURA    --E1 LAURA    --E2
#> [11] LAURA    --E3 LAURA    --E5 LAURA    --E6 LAURA    --E7 LAURA    --E8
#> [16] THERESA  --E2 THERESA  --E3 THERESA  --E4 THERESA  --E5 THERESA  --E6
#> [21] THERESA  --E7 THERESA  --E8 THERESA  --E9 BRENDA   --E1 BRENDA   --E3
#> [26] BRENDA   --E4 BRENDA   --E5 BRENDA   --E6 BRENDA   --E7 BRENDA   --E8
#> [31] CHARLOTTE--E3 CHARLOTTE--E4 CHARLOTTE--E5 CHARLOTTE--E7 FRANCES  --E3
#> [36] FRANCES  --E5 FRANCES  --E6 FRANCES  --E8 ELEANOR  --E5 ELEANOR  --E6
#> + ... omitted several edges
```

Details



References

Davis, Allison, Burleigh B. Gardner, and Mary R. Gardner. 1941. *Deep South*. Chicago: University of Chicago Press.

`mark_nodes` *Marking nodes based on their properties*

Description

These functions return logical vectors the length of the nodes in a network identifying which hold certain properties.

`node_is_cutpoint()` and `node_is_isolate()` are useful for identifying nodes that are in particular positions in the network. More can be added here.

`node_is_max()` and `node_is_min()` are more generally useful for converting the results from some node measure into a mark-class object. They can be particularly useful for highlighting which node or nodes are key because they minimise or, more often, maximise some measure.

Usage

```
node_is_cutpoint(object)

node_is_isolate(object)

node_is_core(object)

node_is_random(object, size = 1)

node_is_max(node_measure, ranks = 1)

node_is_min(node_measure, ranks = 1)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
size	The number of nodes to select (as TRUE).
node_measure	An object created by a node_ measure.
ranks	The number of ranks of max or min to return. For example, ranks = 3 will return TRUE for nodes with scores equal to any of the top (or, for node_is_min(), bottom) three scores. By default, ranks = 1.

Functions

- node_is_cutpoint(): Returns logical of which nodes cut or act as articulation points in a network, increasing the number of connected components in a graph when removed.
- node_is_isolate(): Returns logical of which nodes are isolates, with neither incoming nor outgoing ties.
- node_is_core(): Returns logical of which nodes are members of the core of the network.
- node_is_random(): Returns a logical vector indicating a random selection of nodes as TRUE.
- node_is_max(): Returns logical of which nodes hold the maximum of some measure
- node_is_min(): Returns logical of which nodes hold the minimum of some measure

See Also

Other marks: is(), mark_ties

Examples

```
node_is_cutpoint(ison_brandes)
node_is_isolate(ison_brandes)
node_is_core(ison_brandes)
node_is_random(ison_brandes, 2)
node_is_max(node_degree(ison_brandes))
node_is_min(node_degree(ison_brandes))
```

---

mark_ties *Marking ties based on their properties*

Description

These functions return logical vectors the length of the ties in a network, identifying which hold some property. They are most useful in highlighting parts of the network that are particularly well- or poorly-connected.

Usage

```
tie_is_multiple(object)

tie_is_loop(object)

tie_is_reciprocated(object)

tie_is_bridge(object)

tie_is_max(tie_measure)

tie_is_min(tie_measure)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package
`tie_measure`	An object created by a `tie_` measure.

Functions

- `tie_is_multiple()`: Returns logical of which ties are multiples
- `tie_is_loop()`: Returns logical of which ties are loops

- `tie_is_reciprocated()`: Returns logical of which ties are mutual/reciprocated
- `tie_is_bridge()`: Returns logical of which ties cut or act as articulation points in a network.
- `tie_is_max()`: Returns logical of which ties hold the maximum of some measure
- `tie_is_min()`: Returns logical of which ties hold the minimum of some measure

See Also

Other marks: `is()`, `mark_nodes`

Examples

```
tie_is_multiple(ison_marvel_relationships)
tie_is_loop(ison_marvel_relationships)
tie_is_reciprocated(ison_algebra)
tie_is_bridge(ison_brandes)
tie_is_max(tie_betweenness(ison_brandes))
tie_is_min(tie_betweenness(ison_brandes))
```

---

mpn_bristol *Multimodal (3) Bristol protest events, 1990-2002 (Diani and Bison 2004)*

Description

A multimodal network with three levels representing ties between individuals, civic organisations in Bristol, and major protest and civic events that occurred between 1990 and 2000. The data contains individuals' affiliations to civic organizations in Bristol, the participation of these individuals in major protest and civic events between 1990-2002, and the involvement of the civic organizations in these events.

Usage

```
data(mpn_bristol)
```

Format

```
#> # A tbl_graph: 264 nodes and 1496 edges
#> #
#> # A bipartite simple graph with 3 components
#> #
#> # Node Data: 264 x 3 (active)
#>   name  type    lvl
#>   <chr> <lgl> <dbl>
#> 1 101   FALSE     1
#> 2 102   FALSE     1
#> 3 103   FALSE     1
#> 4 104   FALSE     1
```

```
#> 5 105   FALSE     1
#> 6 106   FALSE     1
#> # ... with 258 more rows
#> #
#> # Edge Data: 1,496 x 2
#>    from    to
#>   <int> <int>
#> 1    36   151
#> 2    40   151
#> 3    73   151
#> # ... with 1,493 more rows
```

Details

Although represented as a two-mode network, it contains three levels:

1. 150 Individuals, anonymised with numeric ID
2. 97 Bristol civic organizations
3. 17 Major protest and civic events in Bristol, 1990-2002

The network represents ties between level 1 (individuals) and level 2 (organisations), level 1 (individuals) and level 3 (events), as well as level 2 (organisations) and level 3 (events). The network is simple, undirected, and named. For a complete list of civic organisations and protest/civic events included in the data, please see Appendix 6.1 in *Multimodal Political Networks* (Knoke et al., 2021).

Source

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

References

Diani, Mario, and Ivano Bison. 2004. "Organizations, Coalitions, and Movements." *Theory and Society* 33(3–4):281–309. doi:10.1023/B:RYSO.0000038610.00045.07.

`mpn_cow` *One-mode interstate trade relations and two-mode state membership in IGOs (COW)*

Description

One-mode interstate trade relations and two-mode state membership in IGOs (COW)

Usage

```
data(mpn_cow_trade)

data(mpn_cow_igo)
```

Format

```
#> # A tbl_graph: 116 nodes and 11489 edges
#> #
#> # A directed simple graph with 1 component
#> #
#> # Node Data: 116 x 1 (active)
#>   name
#>   <chr>
#> 1 United States of America
#> 2 Canada
#> 3 Cuba
#> 4 Dominican Republic
#> 5 Jamaica
#> 6 Trinidad and Tobago
#> # ... with 110 more rows
#> #
#> # Edge Data: 11,489 x 3
#>    from    to  weight
#>   <int> <int>   <dbl>
#> 1     1     2 180387
#> 2     1     3    587.
#> 3     1     4   5511.
#> # ... with 11,486 more rows
```

```
#> # A tbl_graph: 152 nodes and 839 edges
#> #
#> # A directed acyclic simple graph with 1 component
#> #
#> # Node Data: 152 x 3 (active)
#>   name        type  polity2
#>   <chr>       <lgl>   <dbl>
#> 1 Afghanistan FALSE      -7
#> 2 Albania     FALSE       5
#> 3 Algeria     FALSE      -3
#> 4 Angola      FALSE      -6
#> 5 Argentina   FALSE       8
#> 6 Australia   FALSE      10
#> # ... with 146 more rows
#> #
#> # Edge Data: 839 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1   113      1
#> 2     1   114      1
#> 3     1   115      0
#> # ... with 836 more rows
```

Details

`mpn_cow_trade` is a one-mode matrix representing the trade relations between 116 states. The data is derived from the Correlates of War Project (COW) Trade Dataset (v3.0), which contains the annual dyadic and national trade figures for states (listed in COW) between 1870 to 2009. This network is based only on the dyadic trade figures in 2009 for the 116 states listed in Appendix 7.1 in *Multimodal Political Networks* (Knoke et al., 2021). The value in each cell of the matrix represents the value of exports from the 116 row states to the 116 column states.

`mpn_cow_igo` is a two-mode graph representing the membership of 116 states in 40 intergovernmental organizations (IGOs). The data is derived from the Correlates of War Project (COW) Intergovernmental Organizations Dataset (v3.0), which contains information about intergovernmental organizations from 1815-2014, such as founding year and membership. This network contains only a subset of the states and IGOs listed in COW, with 116 states listed in Appendix 7.1 in *Multimodal Political Networks* and 40 IGOs from Table 7.1 in *Multimodal Political Networks* that also overlap with the COW dataset (Knoke et al., 2021).

Source

The Correlates of War Project. 2012. *Trade*.

Barbieri, Katherine and Omar Keshk. 2012. Correlates of War Project Trade Data Set Codebook, Version 3.0.

The Correlates of War Project. 2019. *Intergovernmental Organization v3*.

References

Barbieri, Katherine, Omar M. G. Keshk, and Brian Pollins. 2009. "TRADING DATA: Evaluating our Assumptions and Coding Rules." *Conflict Management and Peace Science* 26(5): 471-491. doi:10.1177/0738894209343887.

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

Pevehouse, Jon C.W., Timothy Nordstron, Roseanne W McManus, Anne Spencer Jamison. 2020. "Tracking Organizations in the World: The Correlates of War IGO Version 3.0 datasets". *Journal of Peace Research* 57(3): 492-503. doi:10.1177/0022343319881175.

`mpn_elite_mex` *One-mode Mexican power elite database (Knoke 1990)*

Description

This data contains the full network of 35 members of the Mexican power elite. The undirected lines connecting pairs of men represent any formal, informal, or organizational relation between a dyad; for example, "common belonging (school, sports, business, political participation), or a common interest (political power)" (Mendieta et al. 1997: 37). Additional nodal attributes include their full name, place of birth, state, and region (1=North, 2=Centre, 3=South, original coding added by Frank Heber), as well as their year of entry into politics and whether they are civilian (0) or affiliated with the military (1). An additional variable "in_mpn" can be used to subset this network to a network of

11 core members of the 1990s Mexican power elite (Knoke 2017), three of which were successively elected presidents of Mexico: José López Portillo (1976-82), Miguel de la Madrid (1982-88), and Carlos Salinas de Gortari (1988-94, who was also the son of another core member, Raúl Salinas Lozano).

Usage

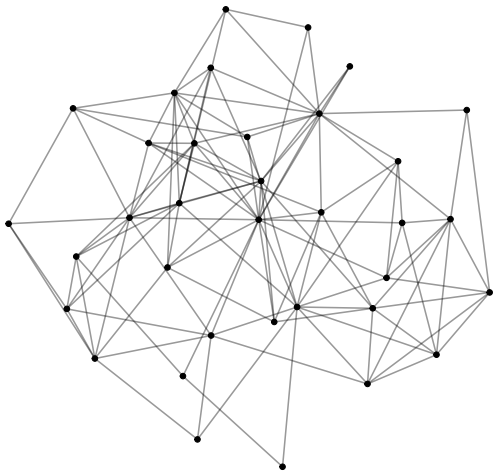
```
data(mpn_elite_mex)
```

Format

```
#> # A tbl_graph: 35 nodes and 117 edges
#> #
#> # An undirected simple graph with 1 component
#> #
#> # Node Data: 35 x 8 (active)
#>   name    full_name          entry_year military in_mpn PlaceOf~ state region
#>   <chr>   <chr>                   <dbl>    <dbl>  <dbl> <chr>    <chr>  <dbl>
#> 1 Trevino  Trevino, Jacinto B.       1910        1      0 Guerrero Coah~      1
#> 2 Madero   Madero, Francisco         1911        0      0 Parras ~ Coah~      1
#> 3 Carranza Carranza, Venustiano      1913        1      0 Cuatro ~ Coah~      1
#> 4 Aguilar  Aguilar, Candido          1918        1      0 Cordoba  Vera~      3
#> 5 Obregon  Obregon, Alvaro           1920        1      0 Siquisi~ Sono~      1
#> 6 Calles   Calles, Plutarco E.       1924        1      0 Guaymas  Sono~      1
#> # ... with 29 more rows
#> #
#> # Edge Data: 117 x 2
#>    from    to
#>   <int> <int>
#> 1     2     3
#> 2     2     5
#> 3     2     6
#> # ... with 114 more rows
```

Details

mpn_elite_mex



Source

Knoke, David. 1990. *Political Networks*.

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

`mpn_elite_usa` *Two-mode and three-mode American power elite database (Domhoff 2016)*

Description

`mpn_elite_usa_advice` is a 2-mode network of persons serving as directors or trustees of think tanks. Think tanks are "public-policy research analysis and engagement organizations that generate policy-oriented research, analysis, and advice on domestic and international issues, thereby enabling policymakers and the public to make informed decisions about public policy" (McGann 2016: 6). The Power Elite Database (Domhoff 2016) includes information on the directors of 33 prominent think tanks in 2012. Here we include only 14 directors who held three or more seats among 20 think tanks.

`mpn_elite_usa_money` is based on 26 elites who sat on the boards of directors for at least two of six economic policy making organizations (Domhoff 2016), and also made campaign contributions to one or more of six candidates running in the primary election contests for the 2008 Presidential nominations of the Republican Party (Rudy Giuliani, John McCain, Mitt Romney) or the Democratic Party (Hillary Clinton, Christopher Dodd, Barack Obama).

Usage

```
data(mpn_elite_usa_advice)

data(mpn_elite_usa_money)
```

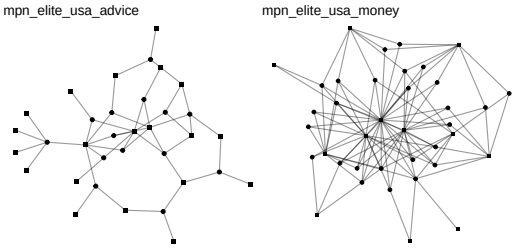
Format

```
#> # A tbl_graph: 34 nodes and 46 edges
#> #
#> # A bipartite simple graph with 1 component
#> #
#> # Node Data: 34 x 2 (active)
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Albright
#> 2 FALSE Argyros
#> 3 FALSE Armitage
#> 4 FALSE Curry
#> 5 FALSE Fukuyama
#> 6 FALSE Gray
#> # ... with 28 more rows
#> #
#> # Edge Data: 46 x 2
#>    from    to
#>   <int> <int>
#> 1     1    17
#> 2     1    19
#> 3     1    21
#> # ... with 43 more rows

#> # A tbl_graph: 38 nodes and 103 edges
#> #
#> # A bipartite simple graph with 1 component
#> #
#> # Node Data: 38 x 2 (active)
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Adkerson
#> 2 FALSE Akins
#> 3 FALSE Banga
#> 4 FALSE Boyce
#> 5 FALSE Britt
#> 6 FALSE Cannon
#> # ... with 32 more rows
#> #
#> # Edge Data: 103 x 2
#>    from    to
#>   <int> <int>
```

```
#> 1     1    27
#> 2     1    28
#> 3     1    34
#> # ... with 100 more rows
```

Details



References

Domhoff, G William. 2016. "Who Rules America? Power Elite Database."

The Center for Responsive Politics. 2019. "OpenSecrets."

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

`mpn_evs` *Two-mode European Values Survey, 1990 and 2008 (EVS 2020)*

Description

6 two-mode matrices containing individuals' memberships to 14 different types of associations in three countries (Italy, Germany, and the UK) in 1990 and 2008. The Italy data has 658 respondents in 1990 and 540 in 2008. The Germany data has 1369 respondents in 1990 and 503 in 2008. The UK data has 738 respondents in 1990 and 664 in 2008.

Usage

```
data(mpn_IT_1990)

data(mpn_IT_1990)

data(mpn_IT_2008)

data(mpn_DE_1990)

data(mpn_DE_2008)

data(mpn_UK_1990)

data(mpn_UK_2008)
```

Format

`tbl_graph` object based on an association matrix with 14 columns:

Welfare 1 if individual associated

Religious 1 if individual associated

Education.culture 1 if individual associated

Unions 1 if individual associated

Parties 1 if individual associated

Local.political.groups 1 if individual associated

Human.rights 1 if individual associated

Environmental.animal 1 if individual associated

Professional 1 if individual associated

Youth 1 if individual associated

Sports 1 if individual associated

Women 1 if individual associated

Peace 1 if individual associated

Health 1 if individual associated

An object of class `tbl_graph` (inherits from `igraph`) of length 672.

An object of class `tbl_graph` (inherits from `igraph`) of length 554.

An object of class `tbl_graph` (inherits from `igraph`) of length 1383.

An object of class `tbl_graph` (inherits from `igraph`) of length 517.

An object of class `tbl_graph` (inherits from `igraph`) of length 752.

An object of class `tbl_graph` (inherits from `igraph`) of length 678.

Source

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

References

EVS (2020). European Values Study Longitudinal Data File 1981-2008 (EVS 1981-2008). GESIS Data Archive, Cologne. ZA4804 Data file Version 3.1.0, doi:10.4232/1.13486.

`mpn_ryanair` *One-mode EU policy influence network, June 2004 (Christopoulos 2006)*

Description

Network of anonymised actors reacting to the Ryanair/Charleroi decision of the EU Commission in February 2004. The relationships mapped comprise an account of public records of interaction supplemented with the cognitive network of key informants. Examination of relevant communiques, public statements and a number of off-the-record interviews provides confidence that the network mapped closely approximated interactions between 29 January and 12 February 2004. The time point mapped is at the height of influence and interest intermediation played by actors in the AER, a comparatively obscure body representing the interests of a number of European regional bodies at the EU institutions.

Usage

```
data(mpn_ryanair)
```

Format

```
#> # A tbl_graph: 20 nodes and 177 edges
#> #
#> # A directed simple graph with 1 component
#> #
#> # Node Data: 20 x 1 (active)
#>   name
#>   <chr>
#> 1 1 AER
#> 2 2 AER
#> 3 5 AER/COR
#> 4 7 RYANAIR
#> 5 8 DG TRANSPORT
#> 6 9 COR
#> # ... with 14 more rows
#> #
#> # Edge Data: 177 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1     2      1
#> 2     1     3      1
#> 3     1     4      1
#> # ... with 174 more rows
```

Source

Christopoulos, Dimitrios C. 2006. "Relational Attributes of Political Entrepreneurs: a Network Perspective." *Journal of European Public Policy* 13(5): 757–78. doi:10.1080/13501760600808964.

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

`mpn_senate112` *Two-mode 112th Congress Senate Voting (Knoke et al. 2021)*

Description

These datasets list the U.S. Senators who served in the 112th Congress, which met from January 3, 2011 to January 3, 2013. Although the Senate has 100 seats, 103 persons served during this period due to two resignations and a death. However, the third replacement occurred only two days before the end and cast no votes on the bills investigated here. Hence, the number of Senators analyzed is 102.

CQ Almanac identified 25 key bills on which the Senate voted during the 112th Congress, and which Democratic and Republican Senators voting "yea" and "nay" on each proposal.

Lastly, we obtained data on campaign contributions made by 92 PACs from the Open Secrets Website. We recorded all contributions made during the 2008, 2010, and 2012 election campaigns to the 102 persons who were Senators in the 112th Congress. The vast majority of PAC contributions to a candidate during a campaign was for $10,000 (the legal maximum is $5,000 each for a primary and the general election). We aggregated the contributions across all three electoral cycles, then dichotomized the sums into no contribution (0) and any contribution (1).

Usage

```
data(mpn_DemSxP)

data(mpn_RepSxP)

data(mpn_OverSxP)
```

Format

```
#> # A tbl_graph: 114 nodes and 2791 edges
#> #
#> # A directed acyclic simple graph with 1 component
#> #
#> # Node Data: 114 x 2 (active)
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Baucus
#> 2 FALSE Begich
#> 3 FALSE Bennet
#> 4 FALSE Blumenthal
#> 5 FALSE Boxer
#> 6 FALSE BrownSh
#> # ... with 108 more rows
```

```
#> #
#> # Edge Data: 2,791 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1    52      1
#> 2     1    53      1
#> 3     1    54      1
#> # ... with 2,788 more rows

#> # A tbl_graph: 134 nodes and 3675 edges
#> #
#> # A directed acyclic simple graph with 1 component
#> #
#> # Node Data: 134 x 2 (active)
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Alexander
#> 2 FALSE Ayotte
#> 3 FALSE Barrasso
#> 4 FALSE Baucus
#> 5 FALSE Blunt
#> 6 FALSE Boozman
#> # ... with 128 more rows
#> #
#> # Edge Data: 3,675 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1    64      1
#> 2     1    66      1
#> 3     1    67      1
#> # ... with 3,672 more rows

#> # A tbl_graph: 52 nodes and 614 edges
#> #
#> # A directed acyclic simple graph with 1 component
#> #
#> # Node Data: 52 x 2 (active)
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Baucus
#> 2 FALSE Cardin
#> 3 FALSE Carper
#> 4 FALSE Casey
#> 5 FALSE Collins
#> 6 FALSE Feinstein
#> # ... with 46 more rows
#> #
#> # Edge Data: 614 x 3
#>    from    to weight
```

```
#>   <int> <int>  <dbl>
#> 1     1    21      1
#> 2     1    22      1
#> 3     1    23      1
#> # ... with 611 more rows
```

References

Knoke, David, Mario Diani, James Hollway, and Dimitris C Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press. Cambridge University Press.

`network_census` *Censuses of motifs at the network level*

Description

Censuses of motifs at the network level

Usage

```
network_dyad_census(object)

network_triad_census(object)

network_mixed_census(object, object2)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package
`object2`	A second, two-mode migraph-consistent object.

Functions

- `network_dyad_census()`: Returns a census of dyad motifs in a network
- `network_triad_census()`: Returns a census of triad motifs in a network
- `network_mixed_census()`: Returns a census of triad motifs that span a one-mode and a two-mode network

Source

Alejandro Espinosa 'netmem'

References

Davis, James A., and Samuel Leinhardt. 1967. "The Structure of Positive Interpersonal Relations in Small Groups." 55.

Hollway, James, Alessandro Lomi, Francesca Pallotti, and Christoph Stadtfeld. 2017. "Multilevel Social Spaces: The Network Dynamics of Organizational Fields." *Network Science* 5(2): 187–212. doi:10.1017/nws.2017.8

See Also

Other motifs: `brokerage_census`, `node_census`

Examples

```
network_dyad_census(ison_algebra)
network_triad_census(ison_adolescents)
marvel_friends <- to_unsigned(ison_marvel_relationships, "positive")
(mixed_cen <- network_mixed_census(marvel_friends, ison_marvel_teams))
```

---

node_census *Censuses of nodes' motifs*

Description

These functions include ways to take a census of the positions of nodes in a network. These include a triad census based on the triad profile of nodes, but also a tie census based on the particular tie partners of nodes. Included also are group census functions for summarising the profiles of clusters of nodes in a network.

#' @export node_igraph_census <- function(object, normalized = FALSE) out <- igraph::motifs(as_igraph(object), 4) if(is_labelled(object)) rownames(out) <- node_names(object) colnames(out) <- c("co-K4", "co-diamond", "co-C4", "co-paw", "co-claw", "P4", "claw", "paw", "C4", "diamond", "K4") make_node_motif(out, object)

Usage

```
node_tie_census(object)

node_triad_census(object)

node_quad_census(object)

node_path_census(object)
```

Arguments

`object` An object of a migraph-consistent class:

- matrix (adjacency or incidence) from `{base}` R
- edgelist, a data frame from `{base}` R or tibble from `{tibble}`
- igraph, from the `{igraph}` package
- network, from the `{network}` package
- tbl_graph, from the `{tidygraph}` package

Details

The quad census uses the `{oaqc}` package to do the heavy lifting of counting the number of each orbits. See `vignette('oaqc')`. However, our function relabels some of the motifs to avoid conflicts and improve some consistency with other census-labelling practices. The letter-number pairing of these labels indicate the number and configuration of ties. For now, we offer a rough translation:

migraph	Ortmann and Brandes
E4	co-K4
I40, I41	co-diamond
H4	co-C4
L42, L41, L40	co-paw
D42, D40	co-claw
U42, U41	P4
Y43, Y41	claw
P43, P42, P41	paw
04	C4
Z42, Z43	diamond
X4	K4

See also this list of graph classes.

Functions

- `node_tie_census()`: Returns a census of the ties in a network. For directed networks, out-ties and in-ties are bound together.
- `node_triad_census()`: Returns a census of the triad configurations nodes are embedded in.
- `node_quad_census()`: Returns a census of nodes' positions in motifs of four nodes.
- `node_path_census()`: Returns the shortest path lengths of each node to every other node in the network.

References

Davis, James A., and Samuel Leinhardt. 1967. "The Structure of Positive Interpersonal Relations in Small Groups." 55.

Ortmann, Mark, and Ulrik Brandes. 2017. "Efficient Orbit-Aware Triad and Quad Census in Directed and Undirected Graphs." *Applied Network Science* 2(1):13. doi:10.1007/s4110901700272.

Dijkstra, Edsger W. 1959. "A note on two problems in connexion with graphs". *Numerische Mathematik* 1, 269-71. doi:10.1007/BF01386390.

Opsahl, Tore, Filip Agneessens, and John Skvoretz. 2010. "Node centrality in weighted networks: Generalizing degree and shortest paths". *Social Networks* 32(3): 245-51. doi:10.1016/j.socnet.2010.03.006.

See Also

Other motifs: brokerage_census, network_census

Examples

```
task_eg <- to_named(to_uniplex(ison_algebra, "task_tie"))
(tie_cen <- node_tie_census(task_eg))
(triad_cen <- node_triad_census(task_eg))
node_quad_census(ison_southern_women)
node_path_census(ison_adolescents)
node_path_census(ison_southern_women)
```

---

partition_layouts *Layout algorithms based on bi- or other partitions*

Description

Layout algorithms based on bi- or other partitions

Usage

```
layout_tbl_graph_hierarchy(object, circular = FALSE, times = 1000)

layout_tbl_graph_alluvial(object, circular = FALSE, times = 1000)

layout_tbl_graph_railway(object, circular = FALSE, times = 1000)

layout_tbl_graph_ladder(object, circular = FALSE, times = 1000)

layout_tbl_graph_concentric(
  object,
  membership = NULL,
  radius = NULL,
  order.by = NULL,
  circular = FALSE,
  times = 1000
)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
`circular`	Should the layout be transformed into a radial representation. Only possible for some layouts. Defaults to FALSE
`times`	Maximum number of iterations, where appropriate
`membership`	A vector of partition memberships.
`radius`	A vector of radii at which the concentric circles should be located. By default this is equal placement around an empty centre, unless one (the core) is a single node, in which case this node occupies the centre of the graph.
`order.by`	An attribute label indicating the (decreasing) order for the nodes around the circles. By default ordering is given by a bipartite placement that reduces the number of edge crossings.

Source

Diego Diez, Andrew P. Hutchins and Diego Miranda-Saavedra. 2014. "Systematic identification of transcriptional regulatory modules from protein-protein interaction networks". *Nucleic Acids Research*, 42 (1) e6.

See Also

Other mapping: `auto_graph`, `grid_layouts`

Examples

```
(autographr(ison_southern_women, "hierarchy") /
autographr(ison_southern_women, "railway")) |
autographr(ison_southern_women, "concentric")
autographr(ison_karateka, "hierarchy")
```

---

play *Functions to play games on networks*

Description

Functions to play games on networks

Usage

```
play_diffusion(
  object,
  seeds = 1,
  thresholds = 1,
  transmissibility = 1,
  latency = 0,
  recovery = 0,
  waning = 0,
  immune = NULL,
  steps
)

play_diffusions(
  object,
  seeds = 1,
  thresholds = 1,
  transmissibility = 1,
  latency = 0,
  recovery = 0,
  waning = 0,
  immune = NULL,
  steps,
  times = 5,
  strategy = "sequential",
  verbose = FALSE
)

play_learning(object, beliefs, steps, epsilon = 5e-04)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
seeds	A valid mark vector the length of the number of nodes in the network.
thresholds	A numeric vector indicating the thresholds each node has. By default 1. A single number means a generic threshold; for thresholds that vary among the population please use a vector the length of the number of nodes in the network. If 1 or larger, the threshold is interpreted as a simple count of the number of contacts/exposures sufficient for infection. If less than 1, the threshold is interpreted as complex, where the threshold concerns the proportion of contacts.

`transmissibility`
: A proportion indicating the transmission rate, β. By default 1, which means any node for which the threshold is met or exceeded will become infected. Anything lower means a correspondingly lower probability of adoption, even when the threshold is met or exceeded.

`latency`
: A proportion indicating the rate at which those exposed become infectious (infected), σ. For example, if exposed individuals take, on average, four days to become infectious, then $\sigma = 0.25$. By default 0, which means those exposed become immediately infectious (i.e. an SI model). Anything higher results in e.g. a SEI model.

`recovery`
: A proportion indicating the rate of recovery, γ. For example, if infected individuals take, on average, four days to recover, then $\gamma = 0.25$. By default 0, which means there is no recovery (i.e. an SI model). Anything higher results in an SIR model.

`waning`
: A proportion indicating the rate at which those who are recovered become susceptible again, ξ. For example, if recovered individuals take, on average, four days to lose their immunity, then $\xi = 0.25$. By default 0, which means any recovered individuals retain lifelong immunity (i.e. an SIR model). Anything higher results in e.g. a SIRS model. $\xi = 1$ would mean there is no period of immunity, e.g. an SIS model.

`immune`
: A logical or numeric vector identifying nodes that begin the diffusion process as already recovered. This could be interpreted as those who are vaccinated or equivalent. Note however that a waning parameter will affect these nodes too. By default NULL, indicating that no nodes begin immune.

`steps`
: The number of steps forward in the diffusion to play. By default the number of nodes in the network. If `steps = Inf` then the diffusion process will continue until there are no new infections or all nodes are infected.

`times`
: Integer indicating number of simulations used for quantile estimation. (Relevant to the null hypothesis test only - the analysis itself is unaffected by this parameter.) Note that, as for all Monte Carlo procedures, convergence is slower for more extreme quantiles. By default, `times=1000`. 1,000 - 10,000 repetitions recommended for publication-ready results.

`strategy`
: If `{furrr}` is installed, then multiple cores can be used to accelerate the function. By default `"sequential"`, but if multiple cores available, then `"multisession"` or `"multicore"` may be useful. Generally this is useful only when `times` > 1000. See `{furrr}` for more.

`verbose`
: Whether the function should report on its progress. By default FALSE. See `{progressr}` for more.

`beliefs`
: A vector indicating the probabilities nodes put on some outcome being 'true'.

`epsilon`
: The maximum difference in beliefs accepted for convergence to a consensus.

Functions

- `play_diffusion()`: Playing compartmental diffusion on networks.
- `play_diffusions()`: Playing multiple compartmental diffusions on networks.
- `play_learning()`: Playing DeGroot learning on networks.

See Also

Other models: regression, tests

Examples

```
plot(play_diffusion(generate_smallworld(15, 0.025)))
plot(play_diffusion(generate_smallworld(15, 0.025), thresholds = 0.4))
plot(play_diffusion(generate_smallworld(15, 0.025), recovery = 0.4))
plot(play_diffusions(generate_smallworld(15, 0.025), times = 20))
play_learning(ison_networkers,
      rbinom(network_nodes(ison_networkers),1,prob = 0.25))
```

---

read *Make networks from/to external formats*

Description

Researchers regularly need to work with a variety of external data formats. The following functions offer ways to import from some common external file formats into objects that {migraph} and other graph/network packages in R can work with.

Note that these functions are not as actively maintained as others in the package, so please let us know if any are not currently working for you or if there are missing import routines by raising an issue on Github.

Usage

```
read_matrix(file = file.choose(), sv = c("comma", "semi-colon"), ...)

read_edgelist(file = file.choose(), sv = c("comma", "semi-colon"), ...)

write_edgelist(object, filename, name, ...)

read_nodelist(file = file.choose(), sv = c("comma", "semi-colon"), ...)

write_nodelist(object, filename, name, ...)

read_pajek(file = file.choose(), ties = NULL, ...)

write_pajek(object, filename, ...)

read_ucinet(file = file.choose())

write_ucinet(object, filename, name)

read_dynetml(file = file.choose())
```

Arguments

`file`	A character string with the system path to the file to import. If left unspecified, an OS-specific file picker is opened to help users select it. Note that in `read_ucinet()` the file path should be to the header file (.##h), if it exists and that it is currently not possible to import multiple networks from a single UCINET file. Please convert these one by one.
`sv`	Allows users to specify whether their csv file is `"comma"` (English) or `"semi-colon"` (European) separated.
`...`	Additional parameters passed to the read/write function.
`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from `{base}` R • edgelist, a data frame from `{base}` R or tibble from `{tibble}` • igraph, from the `{igraph}` package • network, from the `{network}` package • tbl_graph, from the `{tidygraph}` package
`filename`	UCINET filename (without ## extension). By default the files will have the same name as the object and be saved to the working directory.
`name`	name of matrix to be known in UCINET. By default the name will be the same as the object.
`ties`	Where there are

Details

There are a number of repositories for network data that hold various datasets in different formats. See for example:

- UCINET data
- Pajek data

See also:

- networkdata
- GML datasets
- UCIrvine Network Data Repository
- KONECT project
- SNAP Stanford Large Network Dataset Collection

Please let us know if you identify any further repositories of social or political networks and we would be happy to add them here.

The `_ucinet` functions only work with relatively recent UCINET file formats, e.g. type 6406 files. To import earlier UCINET file types, you will need to update them first. To import multiple matrices packed into a single UCINET file, you will need to unpack them and convert them one by one.

Value

`read_edgelist()` and `read_nodelist()` will import into edgelist (tibble) format which can then be coerced or combined into different graph objects from there.

`read_pajek()` and `read_ucinet()` will import into a tidygraph format, since they already contain both edge and attribute data. `read_matrix()` will import into tidygraph format too. Note that all graphs can be easily coerced into other formats with `{migraph}`'s `as_` methods.

The `write_`functions export to different file formats, depending on the function.

A pair of UCINET files in V6404 file format (.##h, .##d)

Functions

- `read_matrix()`: Reading adjacency matrices from Excel/csv files
- `read_edgelist()`: Reading edgelists from Excel/csv files
- `write_edgelist()`: Writing edgelists to csv files
- `read_nodelist()`: Reading nodelists from Excel/csv files
- `write_nodelist()`: Writing nodelists to csv files
- `read_pajek()`: Reading pajek (.net/.paj) files
- `write_pajek()`: Writing pajek .net files
- `read_ucinet()`: Reading UCINET files
- `write_ucinet()`: Writing UCINET files
- `read_dynetml()`: Reading DynetML files

Source

`read_ucinet()` and `write_ucinet()` kindly supplied by Christian Steglich, constructed on 18 June 2015.

See Also

as

Other makes: create, generate

Examples

```
## Not run:
# import Roethlisberger & Dickson's horseplay game data set:
horseplay <- read_ucinet("WIRING-RDGAM.##h")

## End(Not run)
## Not run:
# export it again to UCINET under a different name:
write_ucinet(horseplay, "R&D-horseplay")

## End(Not run)
```

`reformat` *Tools for reformatting networks, graphs, and matrices*

Description

These functions offer tools for reformatting migraph-consistent objects (matrices, igraph, tidygraph, or network objects). Unlike the `as_*()` group of functions, these functions always return the same object type as they are given, only transforming these objects' properties.

Usage

```
to_uniplex(object, edge)

to_undirected(object)

to_redirected(object)

to_unweighted(object, threshold = 1)

to_unsigned(object, keep = c("positive", "negative"))

to_unnamed(object)

to_named(object, names = NULL)

to_simplex(object)

to_onemode(object)

to_multilevel(object)

to_twomode(object, mark)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
`edge`	Character string naming an edge attribute to retain from a graph.
`threshold`	For a matrix, the threshold to binarise/dichotomise at.
`keep`	In the case of a signed network, whether to retain the "positive" or "negative" ties.

names	Character vector of the node names. NULL by default.
mark	A logical vector marking two types or modes. By default "type".

Details

Since some modifications are easier to implement for some objects than others, here are the currently implemented modifications:

to_	edgelists	matrices	igraph	tidygraph	network
unweighted	X	X	X	X	X
undirected		X	X	X	X
redirected	X	X	X	X	
unsigned	X	X	X	X	
uniplex			X	X	
unnamed	X	X	X	X	X
named	X	X	X	X	X
simplex		X	X	X	
onemode			X	X	
multilevel		X	X	X	

Value

All to_ functions return an object of the same class as that provided. So passing it an igraph object will return an igraph object and passing it a network object will return a network object, with certain modifications as outlined for each function.

Functions

- to_uniplex(): Returns an object that includes only a single type of tie
- to_undirected(): Returns an object that has any edge direction removed, so that any pair of nodes with at least one directed edge will be connected by an undirected edge in the new network. This is equivalent to the "collapse" mode in {igraph}.
- to_redirected(): Returns an object that has any edge direction transposed, or flipped, so that senders become receivers and receivers become senders. This essentially has no effect on undirected networks or reciprocated ties.
- to_unweighted(): Returns an object that has all edge weights removed
- to_unsigned(): Returns a network with either just the "positive" ties or just the "negative" ties
- to_unnamed(): Returns an object with all vertex names removed
- to_named(): Returns an object that has random vertex names added
- to_simplex(): Returns an object that has all loops or self-ties removed
- to_onemode(): Returns an object that has any type/mode attributes removed, but otherwise includes all the same nodes and ties. Note that this is not the same as to_mode1() or to_mode2(), which return only some of the nodes and new ties established by coincidence.
- to_multilevel(): Returns a network that is not divided into two mode types but embeds two or more modes into a multimodal network structure.

- `to_twomode()`: Returns a network that divides the nodes into two mode types.

See Also

Other manipulations: `add`, `as()`, `grab`, `split()`, `transform()`

Examples

```
autographr(ison_algebra)
a <- to_uniplex(ison_algebra, "friends")
autographr(a)
a <- to_giant(a)
autographr(a)
a <- to_undirected(a)
autographr(a)
a <- to_unweighted(a)
autographr(a)
```

---

regression *Linear and logistic regression for network data*

Description

This function provides an implementation of the multiple regression quadratic assignment procedure (MRQAP) for both one-mode and two-mode network linear models. It offers several advantages:

- it works with combined graph/network objects such as igraph and network objects by constructing the various dependent and independent matrices for the user.
- it uses a more intuitive formula-based system for specifying the model, with several ways to specify how nodal attributes should be handled.
- it can handle categorical variables (factors/characters) and interactions intuitively, naming the reference variable where appropriate.
- it relies on {furrr} for parallelising and {progressr} for reporting progress to the user, which can be useful when many simulations are required.
- results are {broom}-compatible, with `tidy()` and `glance()` reports to facilitate comparison with results from different models. Note that a *t*- or *z*-value is always used as the test statistic, and properties of the dependent network – modes, directedness, loops, etc – will always be respected in permutations and analysis.

Usage

```
network_reg(
  formula,
  object,
  method = c("qap", "qapy"),
  times = 1000,
```

```
  strategy = "sequential",
  verbose = FALSE
)
```

Arguments

formula	A formula describing the relationship being tested. Several additional terms are available to assist users investigate the effects they are interested in. These include: • ego() constructs a matrix where the cells reflect the value of a named nodal attribute for an edge's sending node • alter() constructs a matrix where the cells reflect the value of a named nodal attribute for an edge's receiving node • same() constructs a matrix where a 1 reflects if two nodes' attribute values are the same • dist() constructs a matrix where the cells reflect the absolute difference between the attribute's values for the sending and receiving nodes • sim() constructs a matrix where the cells reflect the proportional similarity between the attribute's values for the sending and receiving nodes • tertius() constructs a matrix where the cells reflect some aggregate of an attribute associated with a node's other ties. Currently "mean" and "sum" are available aggregating functions. 'ego' is excluded from these calculations. See Haunss and Hollway (2023) for more on this effect. • dyadic covariates (other networks) can just be named
object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
method	A method for establishing the null hypothesis. Note that "qap" uses Dekker et al's (2007) double semi-partialling technique, whereas "qapy" permutes only the y variable. "qap" is the default.
times	Integer indicating number of simulations used for quantile estimation. (Relevant to the null hypothesis test only - the analysis itself is unaffected by this parameter.) Note that, as for all Monte Carlo procedures, convergence is slower for more extreme quantiles. By default, times=1000. 1,000 - 10,000 repetitions recommended for publication-ready results.
strategy	If {furrr} is installed, then multiple cores can be used to accelerate the function. By default "sequential", but if multiple cores available, then "multisession" or "multicore" may be useful. Generally this is useful only when times > 1000. See {furrr} for more.
verbose	Whether the function should report on its progress. By default FALSE. See {progressr} for more.

References

Krackhardt, David. 1988. "Predicting with Networks: Nonparametric Multiple Regression Analysis of Dyadic Data." *Social Networks* 10(4):359–81. doi:10.1016/03788733(88)900044.

Dekker, David, David Krackhard, and Tom A. B. Snijders. 2007. "Sensitivity of MRQAP tests to collinearity and autocorrelation conditions." *Psychometrika* 72(4): 563-581. doi:10.1007/s11336-00790161.

See Also

`vignette("p7linearmodel")`

Other models: play, tests

Examples

```
networkers <- ison_networkers %>% to_subgraph(Discipline == "Sociology")
model1 <- network_reg(weight ~ alter(Citations) + sim(Citations),
                      networkers, times = 20)
# Should be run many more `times` for publication-ready results
tidy(model1)
glance(model1)
plot(model1)
```

---

split *Tools for splitting networks, graphs, and matrices*

Description

These functions offer tools for splitting migraph-consistent objects (matrices, igraph, tidygraph, or network objects). Splitting means that the returned object will be a list of objects.

Usage

```
to_egos(object, max_dist = 1, min_dist = 0)

to_subgraphs(object, attribute)

to_components(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package

max_dist	The maximum breadth of the neighbourhood. By default 1.
min_dist	The minimum breadth of the neighbourhood. By default 0. Increasing this to 1 excludes the ego, and 2 excludes ego's direct alters.
attribute	A character string indicating the categorical attribute in a network used to split into subgraphs.

Functions

- to_egos(): Returns a list of ego (or focal) networks.
- to_subgraphs(): Returns a list of subgraphs on some given attribute.
- to_components(): Returns a list of the components in a network.

See Also

Other manipulations: add, as(), grab, reformat, transform()

Examples

```
autographs(to_egos(ison_adolescents))
autographs(to_egos(ison_adolescents,2))
to_components(ison_marvel_relationships)
```

---

tests *Conditional uniform graph and permutation tests*

Description

These functions conduct conditional uniform graph (CUG) or permutation (QAP) tests of any graph-level statistic.

Usage

```
test_random(
  object,
  FUN,
  ...,
  times = 1000,
  strategy = "sequential",
  verbose = FALSE
)

test_permutation(
  object,
  FUN,
  ...,
  times = 1000,
  strategy = "sequential",
  verbose = FALSE
)
```

Arguments

object	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
FUN	A graph-level statistic function to test.
...	Additional arguments to be passed on to FUN, e.g. the name of the attribute.
times	Integer indicating number of simulations used for quantile estimation. (Relevant to the null hypothesis test only - the analysis itself is unaffected by this parameter.) Note that, as for all Monte Carlo procedures, convergence is slower for more extreme quantiles. By default, times=1000. 1,000 - 10,000 repetitions recommended for publication-ready results.
strategy	If {furrr} is installed, then multiple cores can be used to accelerate the function. By default "sequential", but if multiple cores available, then "multisession" or "multicore" may be useful. Generally this is useful only when times > 1000. See {furrr} for more.
verbose	Whether the function should report on its progress. By default FALSE. See {progressr} for more.

Functions

- test_random(): Returns test results for some measure on an object against a distribution of measures on random networks of the same dimensions
- test_permutation(): Returns test results for some measure on an object against a distribution of measures on permutations of the original network

See Also

Other models: play, regression

Examples

```
marvel_friends <- to_unsigned(ison_marvel_relationships)
marvel_friends <- to_giant(marvel_friends) %>%
  to_subgraph(PowerOrigin == "Human")
(cugtest <- test_random(marvel_friends, network_homophily, attribute = "Attractive",
  times = 200))
plot(cugtest)
(qaptest <- test_permutation(marvel_friends,
                network_homophily, attribute = "Attractive",
                times = 200))
plot(qaptest)
```

`tie_centrality` *Measures of tie centrality*

Description

Measures of tie centrality

Usage

```
tie_degree(object, normalized = TRUE)

tie_closeness(object, normalized = TRUE)

tie_betweenness(object, normalized = TRUE)

tie_eigenvector(object, normalized = TRUE)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix (adjacency or incidence) from {base} R • edgelist, a data frame from {base} R or tibble from {tibble} • igraph, from the {igraph} package • network, from the {network} package • tbl_graph, from the {tidygraph} package
`normalized`	Logical scalar, whether the centrality scores are normalized. Different denominators are used depending on whether the object is one-mode or two-mode, the type of centrality, and other arguments.

Functions

- `tie_degree()`: Calculate the degree centrality of edges in a network
- `tie_closeness()`: Calculate the closeness of each edge to each other edge in the network.
- `tie_betweenness()`: Calculate number of shortest paths going through an edge
- `tie_eigenvector()`: Calculate the eigenvector centrality of edges in a network

See Also

Other measures: `centralisation`, `centrality`, `closure`, `cohesion()`, `diversity`, `features`, `holes`

Examples

```
tie_degree(ison_adolescents)
(ec <- tie_closeness(ison_adolescents))
plot(ec)
ison_adolescents %>%
  activate(edges) %>% mutate(weight = ec) %>%
  autographr()
(tb <- tie_betweenness(ison_adolescents))
plot(tb)
ison_adolescents %>%
  activate(edges) %>% mutate(weight = tb) %>%
  autographr()
tie_eigenvector(ison_adolescents)
```

---

transform *Tools for transforming networks, graphs, and matrices*

Description

These functions offer tools for transforming migraph-consistent objects (matrices, igraph, tidygraph, or network objects). Transforming means that the returned object may have different dimensions than the original object.

Usage

```
to_mode1(object, similarity = c("count", "jaccard", "rand", "pearson", "yule"))

to_mode2(object, similarity = c("count", "jaccard", "rand", "pearson", "yule"))

to_giant(object)

to_subgraph(object, ...)

to_ties(object)

to_blocks(object, membership, FUN = mean)

to_matching(object, mark = "type")

to_anti(object)
```

Arguments

`object` An object of a migraph-consistent class:

- matrix (adjacency or incidence) from {base} R
- edgelist, a data frame from {base} R or tibble from {tibble}
- igraph, from the {igraph} package

- network, from the {network} package
- tbl_graph, from the {tidygraph} package

`similarity` Method for establishing ties, currently "count" (default), "jaccard", or "rand". "count" calculates the number of coinciding ties, and can be interpreted as indicating the degree of opportunities between nodes. "jaccard" uses this count as the numerator in a proportion, where the denominator consists of any cell where either node has a tie. It can be interpreted as opportunity weighted by participation. "rand", or the Simple Matching Coefficient, is a proportion where the numerator consists of the count of cells where both nodes are present or both are absent, over all possible cells. It can be interpreted as the (weighted) degree of behavioral mirroring between two nodes. "pearson" (Pearson's coefficient) and "yule" (Yule's Q) produce correlations for valued and binary data, respectively. Note that Yule's Q has a straightforward interpretation related to the odds ratio.

`...` Arguments passed on to dplyr::filter

`membership` A vector of partition memberships.

`FUN` A function for summarising block content. By default `mean`. Other recommended options include `median`, `sum`, `min` or `max`.

`mark` A logical vector marking two types or modes. By default "type".

Details

Since some modifications are easier to implement for some objects than others, here are the currently implemented modifications:

to_	edgelists	matrices	igraph	tidygraph	network
mode1	X	X	X	X	X
mode2	X	X	X	X	X
giant	X	X	X	X	X
subgraph	X	X	X	X	X
ties	X	X	X	X	X
blocks	X	X	X	X	X
matching	X	X	X	X	X

Functions

- `to_mode1()`: Results in a weighted one-mode object that retains the row nodes from a two-mode object, and weights the ties between them on the basis of their joint ties to nodes in the second mode (columns)
- `to_mode2()`: Results in a weighted one-mode object that retains the column nodes from a two-mode object, and weights the ties between them on the basis of their joint ties to nodes in the first mode (rows).
- `to_giant()`: Returns an object that includes only the main component without any smaller components or isolates
- `to_subgraph()`: Returns a network subgraph filtered on the basis of some node-related logical statement.

- `to_ties()`: Returns a matrix (named if possible) where the edges are the nodes
- `to_blocks()`: Returns a reduced graph from a given partition membership vector. Reduced graphs provide summary representations of network structures by collapsing groups of connected nodes into single nodes while preserving the topology of the original structures.
- `to_matching()`: Returns a network with only matching ties
- `to_anti()`: Returns the complement of a network where only ties *not* present in the original network are included in the new network.

to_matching

`to_matching()` uses {igraph}'s `max_bipartite_match()` to return a network in which each node is only tied to one of its previous ties. The number of these ties left is its *cardinality*, and the algorithm seeks to maximise this such that, where possible, each node will be associated with just one node in the other mode or some other mark. The algorithm used is the push-relabel algorithm with greedy initialization and a global relabelling after every $\frac{n}{2}$ steps, where n is the number of nodes in the network.

References

Goldberg, A V; Tarjan, R E (1986). "A new approach to the maximum flow problem". *Proceedings of the eighteenth annual ACM symposium on Theory of computing – STOC '86*. p. 136. doi:10.1145/12130.12144

See Also

Other manipulations: `add`, `as()`, `grab`, `reformat`, `split()`

Examples

```
autographr(ison_southern_women) /
(autographr(to_mode1(ison_southern_women)) |
autographr(to_mode2(ison_southern_women)))
autographr(ison_adolescents) +
autographr(to_ties(ison_adolescents))
(adolblock <- to_blocks(ison_adolescents,
  node_regular_equivalence(ison_adolescents, k = 3)))
autographr(adolblock)
autographr(to_matching(ison_southern_women), "hierarchy")
autographr(to_anti(ison_southern_women), "hierarchy")
```

Index