Package 'migraph'

November 2, 2023

Title Multimodal Network Analysis and More

Version 1.1.5

Date 2023-11-02

Description A set of tools for analysing multimodal networks.
It includes functions for measuring
centrality, centralization, cohesion, closure, constraint and diversity,
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.
Built on the 'manynet' package, 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.

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

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

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.6.0), manynet

Imports dplyr (>= 1.1.0), generics, ggplot2, igraph (>= 1.5.0),
network, future, furrr, pillar, purrr, rlang, sna, tidygraph,
tidyr

Suggests concaveman, covr, ggdendro, minMSE, oaqc, roxygen2,
rsconnect, testthat, xml2

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-11-02 10:10:02 UTC

R topics documented:

`between_centrality` *Measures of betweenness-like centrality and centralisation*

Description

Measures of betweenness-like centrality and centralisation

Usage

```
node_betweenness(.data, normalized = TRUE, cutoff = NULL)

tie_betweenness(.data, normalized = TRUE)

network_betweenness(
  .data,
  normalized = TRUE,
  direction = c("all", "out", "in")
)
```

Arguments

`.data`	An object of a {manynet}-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.
`cutoff`	The maximum path length to consider when calculating betweenness. If negative or NULL (the default), there's no limit to the path lengths considered.
`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.

Value

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

Functions

- `node_betweenness()`: Calculate the betweenness centralities of nodes in a network
- `tie_betweenness()`: Calculate number of shortest paths going through a tie
- `network_betweenness()`: Calculate the betweenness centralization for a network

See Also

Other measures: `close_centrality`, `closure`, `cohesion()`, `degree_centrality`, `eigenv_centrality`, `features`, `heterogeneity`, `hierarchy`, `holes`

Other centrality: `close_centrality`, `degree_centrality`, `eigenv_centrality`

Examples

```
node_betweenness(mpn_elite_mex)
node_betweenness(ison_southern_women)
(tb <- tie_betweenness(ison_adolescents))
plot(tb)
#ison_adolescents %>% mutate_ties(weight = tb) %>%
#   autographr()
network_betweenness(ison_southern_women, direction = "in")
```

---

brokerage_census *Censuses of brokerage motifs*

Description

Censuses of brokerage motifs

Usage

```
node_brokerage_census(.data, membership, standardized = FALSE)

network_brokerage_census(.data, membership, standardized = FALSE)
```

Arguments

`.data`	An object of a {manynet}-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(manynet::ison_networkers, "Discipline")
network_brokerage_census(manynet::ison_networkers, "Discipline")
```

---

close_centrality *Measures of closeness-like centrality and centralisation*

Description

Measures of closeness-like centrality and centralisation

Usage

```
node_closeness(.data, normalized = TRUE, direction = "out", cutoff = NULL)

node_reach(.data, normalized = TRUE, k = 2)

node_harmonic(.data, normalized = TRUE, k = -1)

tie_closeness(.data, normalized = TRUE)

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

network_reach(.data, normalized = TRUE, k = 2)

network_harmonic(.data, normalized = TRUE, k = 2)
```

Arguments

`.data`	An object of a {manynet}-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.
`cutoff`	Maximum path length to use during calculations.
`k`	Integer of steps out to calculate reach

Functions

- `node_closeness()`: Calculate the closeness centrality of nodes in a network
- `node_reach()`: Calculate nodes' reach centrality or how many nodes they can reach within *k* steps
- `node_harmonic()`: Calculate nodes' harmonic centrality or valued centrality. This is thought to behave better than reach centrality for disconnected networks.
- `tie_closeness()`: Calculate the closeness of each edge to each other edge in the network.
- `network_closeness()`: Calculate a network's closeness centralization
- `network_reach()`: Calculate a network's reach centralization
- `network_harmonic()`: Calculate a network's harmonic centralization

References

Marchiori, M, and V Latora. 2000. "Harmony in the small-world". *Physica A* 285: 539-546.

Dekker, Anthony. 2005. "Conceptual distance in social network analysis". *Journal of Social Structure* 6(3).

See Also

Other measures: `between_centrality`, `closure`, `cohesion()`, `degree_centrality`, `eigenv_centrality`, `features`, `heterogeneity`, `hierarchy`, `holes`

Other centrality: `between_centrality`, `degree_centrality`, `eigenv_centrality`

Examples

```
node_closeness(mpn_elite_mex)
node_closeness(ison_southern_women)
node_reach(ison_adolescents)
(ec <- tie_closeness(ison_adolescents))
plot(ec)
#ison_adolescents %>%
#   activate(edges) %>% mutate(weight = ec) %>%
#   autographr()
network_closeness(ison_southern_women, direction = "in")
```

`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(.data, method = "default")

node_reciprocity(.data)

network_transitivity(.data)

node_transitivity(.data)

network_equivalency(.data)

network_congruency(.data, object2)
```

Arguments

`.data`	An object of a {manynet}-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`	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: `between_centrality`, `close_centrality`, `cohesion()`, `degree_centrality`, `eigenv_centrality`, `features`, `heterogeneity`, `hierarchy`, `holes`

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(.data, 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.
`.data`	An object of a {manynet}-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(.data)

network_components(.data)

network_cohesion(.data)

network_adhesion(.data)

network_diameter(.data)

network_length(.data)
```

Arguments

`.data`	An object of a {manynet}-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 `manynet::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: between_centrality, close_centrality, closure, degree_centrality, eigenv_centrality, features, heterogeneity, hierarchy, holes

Examples

```
network_density(mpn_elite_mex)
network_density(mpn_elite_usa_advice)
network_components(mpn_ryanair)
network_components(manynet::to_undirected(mpn_ryanair))
network_cohesion(manynet::ison_marvel_relationships)
network_cohesion(manynet::to_giant(manynet::ison_marvel_relationships))
network_adhesion(manynet::ison_marvel_relationships)
network_adhesion(manynet::to_giant(manynet::ison_marvel_relationships))
network_diameter(manynet::ison_marvel_relationships)
network_diameter(manynet::to_giant(manynet::ison_marvel_relationships))
network_length(manynet::ison_marvel_relationships)
network_length(manynet::to_giant(manynet::ison_marvel_relationships))
```

---

community *Community partitioning algorithms*

Description

These functions offer different algorithms useful for partitioning networks into sets of communities. The different algorithms offer various advantages in terms of computation time, availability on different types of networks, ability to maximise modularity, and their logic or domain of inspiration.

Usage

```
node_optimal(.data)

node_kernighanlin(.data)

node_edge_betweenness(.data)

node_fast_greedy(.data)

node_leading_eigen(.data)

node_walktrap(.data, times = 50)

node_infomap(.data, times = 50)
```

```
node_spinglass(.data, max_k = 200, resolution = 1)

node_louvain(.data, resolution = 1)

node_leiden(.data, resolution = 1)
```

Arguments

.data	An object of a {manynet}-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.
max_k	Integer constant, the number of spins to use as an upper limit of communities to be found. Some sets can be empty at the end.
resolution	The Reichardt-Bornholdt "gamma" resolution parameter for modularity. By default 1, making existing and non-existing ties equally important. Smaller values make existing ties more important, and larger values make missing ties more important.

Functions

- node_optimal(): A problem-solving algorithm that seeks to maximise modularity over all possible partitions.
- node_kernighanlin(): A greedy, iterative, deterministic partitioning algorithm that results in a graph with two equally-sized communities
- 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.
- node_leading_eigen(): A top-down, hierarchical algorithm.
- node_walktrap(): A hierarchical, agglomerative algorithm based on random walks.
- node_infomap(): A hierarchical algorithm based on the information in random walks.
- node_spinglass(): A greedy, iterative, probabilistic algorithm, based on analogy to model from statistical physics.
- node_louvain(): An agglomerative multilevel algorithm that seeks to maximise modularity over all possible partitions.
- node_leiden(): An agglomerative multilevel algorithm that seeks to maximise the Constant Potts Model over all possible partitions.

Optimal

The general idea is to calculate the modularity of all possible partitions, and choose the community structure that maximises this modularity measure. Note that this is an NP-complete problem with exponential time complexity. The guidance in the igraph package is networks of <50-200 nodes is probably fine.

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.

Leading eigenvector

In each step, the network is bifurcated such that modularity increases most. The splits are determined according to the leading eigenvector of the modularity matrix. A stopping condition prevents tightly connected groups from being split further. Note that due to the eigenvector calculations involved, this algorithm will perform poorly on degenerate networks, but will likely obtain a higher modularity than fast-greedy (at some cost of speed).

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.

Infomap

Motivated by information theoretic principles, this algorithm tries to build a grouping that provides the shortest description length for a random walk, where the description length is measured by the expected number of bits per node required to encode the path.

Spin-glass

This is motivated by analogy to the Potts model in statistical physics. Each node can be in one of k "spin states", and ties (particle interactions) provide information about which pairs of nodes want similar or different spin states. The final community definitions are represented by the nodes' spin states after a number of updates. A different implementation than the default is used in the case of signed networks, such that nodes connected by negative ties will be more likely found in separate communities.

Louvain

The general idea is to take a hierarchical approach to optimising the modularity criterion. Nodes begin in their own communities and are re-assigned in a local, greedy way: each node is moved to the community where it achieves the highest contribution to modularity. When no further modularity-increasing reassignments are possible, the resulting communities are considered nodes (like a reduced graph), and the process continues.

Leiden

The general idea is to optimise the Constant Potts Model, which does not suffer from the resolution limit, instead of modularity. As outlined in the `{igraph}` package, the Constant Potts Model object function is:

$$\frac{1}{2m}\sum_{ij}(A_{ij} - \gamma n_i n_j)\delta(\sigma_i, \sigma_j)$$

where *m* is the total tie weight, A_{ij} is the tie weight between *i* and *j*, γ is the so-called resolution parameter, n_i is the node weight of node *i*, and $\delta(\sigma_i, \sigma_j) = 1$ if and only if *i* and *j* are in the same communities and 0 otherwise.

References

Brandes, Ulrik, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, Dorothea Wagner. 2008. "On Modularity Clustering", *IEEE Transactions on Knowledge and Data Engineering* 20(2):172-188.

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

Newman, M, and M Girvan. 2004. "Finding and evaluating community structure in networks." *Physical Review E* 69: 026113.

Clauset, A, MEJ Newman, MEJ and C Moore. "Finding community structure in very large networks."

Newman, MEJ. 2006. "Finding community structure using the eigenvectors of matrices" *Physical Review E* 74:036104.

Pons, Pascal, and Matthieu Latapy "Computing communities in large networks using random walks".

Rosvall, M, and C. T. Bergstrom. 2008. "Maps of information flow reveal community structure in complex networks", *PNAS* 105:1118. doi:10.1073/pnas.0706851105

Rosvall, M., D. Axelsson, and C. T. Bergstrom. 2009. "The map equation", *Eur. Phys. J. Special Topics* 178: 13. doi:10.1140/epjst/e2010011791

Reichardt, Jorg, and Stefan Bornholdt. 2006. "Statistical Mechanics of Community Detection" *Physical Review E*, 74(1): 016110–14. doi:10.1073/pnas.0605965104

Traag, VA, and Jeroen Bruggeman. 2008. "Community detection in networks with positive and negative links".

Blondel, Vincent, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre. 2008. "Fast unfolding of communities in large networks", *J. Stat. Mech.* P10008.

Traag, V. A., L Waltman, and NJ van Eck. 2019. "From Louvain to Leiden: guaranteeing well-connected communities", *Scientific Reports*, 9(1):5233. doi:10.1038/s4159801941695z

See Also

Other memberships: components(), core, equivalence

Examples

```
node_optimal(ison_adolescents)
node_kernighanlin(ison_adolescents)
node_kernighanlin(ison_southern_women)
node_edge_betweenness(ison_adolescents)
node_fast_greedy(ison_adolescents)
node_leading_eigen(ison_adolescents)
node_walktrap(ison_adolescents)
node_infomap(ison_adolescents)
node_spinglass(ison_adolescents)
node_louvain(ison_adolescents)
node_leiden(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(.data)

node_weak_components(.data)

node_strong_components(.data)
```

Arguments

`.data`	An object of a {manynet}-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.

See Also

Other memberships: community, core, equivalence

Examples

```
node_components(mpn_bristol)
```

---

core *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(.data, method = c("degree", "eigenvector"))

node_coreness(.data)
```

Arguments

.data	An object of a {manynet}-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	Which method to use to identify cores and periphery. By default this is "degree", which relies on the heuristic that high degree nodes are more likely to be in the core. An alternative is "eigenvector", which instead begins with high eigenvector nodes. Other methods, such as a genetic algorithm, CONCOR, and Rombach-Porter, can be added if there is interest.

Functions

- node_coreness(): Returns k-cores

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)
node_coreness(ison_adolescents)
```

---

degree_centrality *Measures of degree-like centrality and centralisation*

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(
  .data,
  normalized = TRUE,
  alpha = 0,
  direction = c("all", "out", "in")
)

node_outdegree(.data, normalized = TRUE, alpha = 0)

node_indegree(.data, normalized = TRUE, alpha = 0)

tie_degree(.data, normalized = TRUE)

network_degree(.data, normalized = TRUE, direction = c("all", "out", "in"))

network_outdegree(.data, normalized = TRUE)

network_indegree(.data, normalized = TRUE)
```

Arguments

.data	An object of a {manynet}-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.

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

Functions

- `node_degree()`: Calculates the degree centrality of nodes in an unweighted network, or weighted degree/strength of nodes in a weighted network.
- `node_outdegree()`: Wraps node_degree(..., direction = "out")
- `node_indegree()`: Wraps node_degree(..., direction = "in")
- `tie_degree()`: Calculate the degree centrality of edges in a network
- `network_degree()`: Calculate the degree centralization for a graph
- `network_outdegree()`: Wraps network_degree(..., direction = "out")
- `network_indegree()`: Wraps network_degree(..., direction = "out")

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

See Also

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

Other measures: `between_centrality`, `close_centrality`, `closure`, `cohesion()`, `eigenv_centrality`, `features`, `heterogeneity`, `hierarchy`, `holes`

Other centrality: `between_centrality`, `close_centrality`, `eigenv_centrality`

Examples

```
node_degree(mpn_elite_mex)
node_degree(ison_southern_women)
tie_degree(ison_adolescents)
network_degree(ison_southern_women, direction = "in")
```

`eigenv_centrality` *Measures of eigenvector-like centrality and centralisation*

Description

Measures of eigenvector-like centrality and centralisation

Usage

```
node_eigenvector(.data, normalized = TRUE, scale = FALSE)

node_power(.data, normalized = TRUE, scale = FALSE, exponent = 1)

node_alpha(.data, alpha = 0.85)

node_pagerank(.data)

network_eigenvector(.data, normalized = TRUE)

tie_eigenvector(.data, normalized = TRUE)
```

Arguments

`.data`	An object of a {manynet}-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.
`scale`	Logical scalar, whether to rescale the vector so the maximum score is 1.
`exponent`	Decay rate for the Bonacich power centrality score.
`alpha`	A constant that trades off the importance of external influence against the importance of connection. When $\alpha = 0$, only the external influence matters. As α gets larger, only the connectivity matters and we reduce to eigenvector centrality. By default $\alpha = 0.85$.

Details

We use {igraph} routines behind the scenes here for consistency and because they are often faster. For example, `igraph::eigencentrality()` is approximately 25% faster than `sna::evcent()`.

Value

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

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

Functions

- `node_eigenvector()`: Calculate the eigenvector centrality of nodes in a network
- `node_power()`: Calculate the Bonacich, beta, or power centrality of nodes in a network
- `node_alpha()`: Calculate the alpha or Katz centrality of nodes in a network
- `node_pagerank()`: Calculate the pagerank centrality of nodes in a network
- `network_eigenvector()`: Calculate the eigenvector centralization for a network
- `tie_eigenvector()`: Calculate the eigenvector centrality of edges in a network

Eigenvector centrality

Eigenvector centrality operates as a measure of a node's influence in a network. The idea is that being connected to well-connected others results in a higher score. Each node's eigenvector centrality can be defined as:

$$x_i = \frac{1}{\lambda} \sum_{j \in N} a_{i,j} x_j$$

where $a_{i,j} = 1$ if i is linked to j and 0 otherwise, and λ is a constant representing the principal eigenvalue. Rather than performing this iteration, most routines solve the eigenvector equation $Ax = \lambda x$.

Power centrality

Power or beta (or Bonacich) centrality

Alpha centrality

Alpha or Katz (or Katz-Bonacich) centrality operates better than eigenvector centrality for directed networks. Eigenvector centrality will return 0s for all nodes not in the main strongly-connected component. Each node's alpha centrality can be defined as:

$$x_i = \frac{1}{\lambda} \sum_{j \in N} a_{i,j} x_j + e_i$$

where $a_{i,j} = 1$ if i is linked to j and 0 otherwise, λ is a constant representing the principal eigenvalue, and e_i is some external influence used to ensure that even nodes beyond the main strongly connected component begin with some basic influence. Note that many equations replace $\frac{1}{\lambda}$ with α, hence the name.

For example, if $\alpha = 0.5$, then each direct connection (or alter) would be worth $(0.5)^1 = 0.5$, each secondary connection (or tertius) would be worth $(0.5)^2 = 0.25$, each tertiary connection would be worth $(0.5)^3 = 0.125$, and so on.

Rather than performing this iteration though, most routines solve the equation $x = (I - \frac{1}{\lambda} A^T)^{-1} e$.

References

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.

Katz, Leo 1953. "A new status index derived from sociometric analysis". *Psychometrika*. 18(1): 39–43.

Bonacich, P. and Lloyd, P. 2001. "Eigenvector-like measures of centrality for asymmetric relations" *Social Networks*. 23(3):191-201.

Brin, Sergey and Page, Larry. 1998. "The anatomy of a large-scale hypertextual web search engine". *Proceedings of the 7th World-Wide Web Conference*. Brisbane, Australia.

See Also

Other measures: `between_centrality`, `close_centrality`, `closure`, `cohesion()`, `degree_centrality`, `features`, `heterogeneity`, `hierarchy`, `holes`

Other centrality: `between_centrality`, `close_centrality`, `degree_centrality`

Examples

```
node_eigenvector(mpn_elite_mex)
node_eigenvector(ison_southern_women)
node_power(ison_southern_women, exponent = 0.5)
network_eigenvector(mpn_elite_mex)
network_eigenvector(ison_southern_women)
tie_eigenvector(ison_adolescents)
```

---

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(
  .data,
  census,
  k = c("silhouette", "elbow", "strict"),
  cluster = c("hierarchical", "concor"),
 distance = c("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"),
```

```
  range = 8L
)

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

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

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

Arguments

.data	An object of a {manynet}-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

See Also

Other memberships: community, components(), core

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(.data, membership = NULL)

network_richclub(.data)

network_factions(.data, membership = NULL)

network_modularity(.data, membership = NULL, resolution = 1)

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

network_scalefree(.data)

network_balance(.data)
```

Arguments

`.data`	An object of a {manynet}-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_richclub()`: Returns rich-club coefficient
- `network_factions()`: Returns correlation between a given network and a component model with the same dimensions. If no 'membership' vector is given for the data, `node_kernaghinlin()` is used to obtain a partition into two groups.
- `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: between_centrality, close_centrality, closure, cohesion(), degree_centrality, eigenv_centrality, heterogeneity, hierarchy, holes

Examples

```
network_core(ison_adolescents)
network_core(ison_southern_women)
network_richclub(ison_adolescents)
  network_factions(mpn_elite_mex)
  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)
```

`heterogeneity` *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(.data, attribute)

node_richness(.data, attribute)

network_diversity(.data, attribute, clusters = NULL)

node_diversity(.data, attribute)

network_heterophily(.data, attribute)

node_heterophily(.data, attribute)

network_assortativity(.data)
```

Arguments

`.data`	An object of a {manynet}-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_heterophily()`: Calculates how embedded nodes in the network are within groups of nodes with the same attribute
- `node_heterophily()`: 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: `between_centrality`, `close_centrality`, `closure`, `cohesion()`, `degree_centrality`, `eigenv_centrality`, `features`, `hierarchy`, `holes`

Examples

```
network_richness(mpn_bristol)
node_richness(mpn_bristol, "type")
marvel_friends <- manynet::to_unsigned(manynet::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_heterophily(marvel_friends, "Gender")
network_heterophily(marvel_friends, "Attractive")
node_heterophily(marvel_friends, "Gender")
node_heterophily(marvel_friends, "Attractive")
network_assortativity(mpn_elite_mex)
```

`hierarchy` *Graph theoretic dimensions of hierarchy*

Description

Graph theoretic dimensions of hierarchy

Usage

```
network_connectedness(.data)

network_efficiency(.data)

network_upperbound(.data)
```

Arguments

`.data`	An object of a {manynet}-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_connectedness()`: Returns the proportion of dyads in the network that are reachable, otherwise degree to which network is a single component
- `network_efficiency()`: Returns the Krackhardt efficiency score
- `network_upperbound()`: Returns the Krackhardt (least) upper bound score

References

Krackhardt, David. 1994. Graph theoretical dimensions of informal organizations. In Carley and Prietula (eds) *Computational Organizational Theory*, Hillsdale, NJ: Lawrence Erlbaum Associates. Pp. 89-111.

Everett, Martin, and David Krackhardt. 2012. "A second look at Krackhardt's graph theoretical dimensions of informal organizations." *Social Networks*, 34: 159-163. doi:10.1016/j.socnet.2011.10.006

See Also

Other measures: between_centrality, close_centrality, closure, cohesion(), degree_centrality, eigenv_centrality, features, heterogeneity, holes

Examples

```
network_connectedness(ison_networkers)
1 - network_reciprocity(ison_networkers)
network_efficiency(ison_networkers)
network_upperbound(ison_networkers)
```

---

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(.data)

node_redundancy(.data)

node_effsize(.data)

node_efficiency(.data)

node_constraint(.data)

node_hierarchy(.data)

node_eccentricity(.data)

node_neighbours_degree(.data)

tie_cohesion(.data)
```

Arguments

.data	An object of a {manynet}-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
- `node_eccentricity()`: Returns nodes' eccentricity or Koenig number, a measure of farness based on number of links needed to reach most distant node in the network
- `node_neighbours_degree()`: Returns nodes' average nearest neighbors degree, or knn, a measure of the type of local environment a node finds itself in
- `tie_cohesion()`: Returns the ratio between common neighbors to ties' adjacent nodes and the total number of adjacent nodes, where high values indicate ties' embeddedness in dense local environments

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.

Barrat, Alain, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. 2004. "The architecture of complex weighted networks", *Proc. Natl. Acad. Sci.* 101: 3747.

See Also

Other measures: `between_centrality`, `close_centrality`, `closure`, `cohesion()`, `degree_centrality`, `eigenv_centrality`, `features`, `heterogeneity`, `hierarchy`

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_connected(.data)

is_perfect_matching(.data, mark = "type")

is_eulerian(.data)

is_acyclic(.data)

is_aperiodic(.data, max_path_length = 4)
```

Arguments

`.data`	An object of a {manynet}-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
`mark`	A logical vector marking two types or modes. By default "type".
`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.

Value

TRUE if the condition is met, or FALSE otherwise.

Functions

- `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 `manynet::to_undirected()`.
- `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
- `is_acyclic()`: Tests whether network is a directed acyclic graph
- `is_aperiodic()`: Tests whether network is aperiodic

Source

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

See Also

Other marks: `mark_nodes`, `mark_ties`

Examples

```
is_connected(ison_southern_women)
is_perfect_matching(ison_southern_women)
is_eulerian(ison_brandes)
is_acyclic(ison_algebra)
is_aperiodic(ison_algebra)
```

`kselect` *Methods for selecting clusters*

Description

These functions are generally not user-facing but used internally in e.g. the `*_equivalence()` functions. They are documented here.

Usage

```
k_strict(hc, .data)

k_elbow(hc, .data, census, range)

k_silhouette(hc, .data, range)
```

Arguments

hc	A hierarchical clustering object.
.data	An object of a {manynet}-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 motif census object.
range	An integer indicating the maximum number of options to consider. The minimum of this and the number of nodes in the network is used.

Functions

- k_strict(): Selects a number of clusters in which there is no distance between cluster members.
- k_elbow(): Selects a number of clusters in which there is a fair trade-off between parsimony and fit according to the elbow method.
- k_silhouette(): Selects a number of clusters that optimises the silhouette score.

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.

---

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(.data)

node_is_isolate(.data)

node_is_core(.data)

node_is_random(.data, size = 1)

node_is_max(node_measure, ranks = 1)

node_is_min(node_measure, ranks = 1)
```

Arguments

.data	An object of a {manynet}-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(.data)

tie_is_loop(.data)

tie_is_reciprocated(.data)

tie_is_bridge(.data)

tie_is_max(tie_measure)

tie_is_min(tie_measure)
```

Arguments

`.data`	An object of a {manynet}-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 labelled, two-mode network with 264 nodes and 1496 ties
#> # A tibble: 264 x 3
#>   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
#> # i 258 more rows
```

```
#> # A tibble: 1,496 x 2
#>    from    to
#>   <int> <int>
#> 1    36   151
#> 2    40   151
#> 3    73   151
#> 4    94   151
#> 5   138   151
#> 6   145   151
#> # i 1,490 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 labelled, weighted, directed network with 116 nodes and 11489 arcs
#> # A tibble: 116 x 1
#>   name
#>   <chr>
#> 1 United States of America
#> 2 Canada
#> 3 Cuba
#> 4 Dominican Republic
#> 5 Jamaica
#> 6 Trinidad and Tobago
#> # i 110 more rows
#> # A tibble: 11,489 x 3
#>    from    to  weight
#>   <int> <int>   <dbl>
#> 1     1     2 180387
#> 2     1     3    587.
#> 3     1     4   5511.
#> 4     1     5   1896.
#> 5     1     6   2188.
#> 6     1     7 123677
#> # i 11,483 more rows
```

```
#> # A labelled, weighted, two-mode network with 152 nodes and 839 ties
#> # A tibble: 152 x 3
#>   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
#> # i 146 more rows
#> # A tibble: 839 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1   113      1
#> 2     1   114      1
#> 3     1   115      0
#> 4     1   116      0
#> 5     1   117      1
#> 6     1   118      0
#> # i 833 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

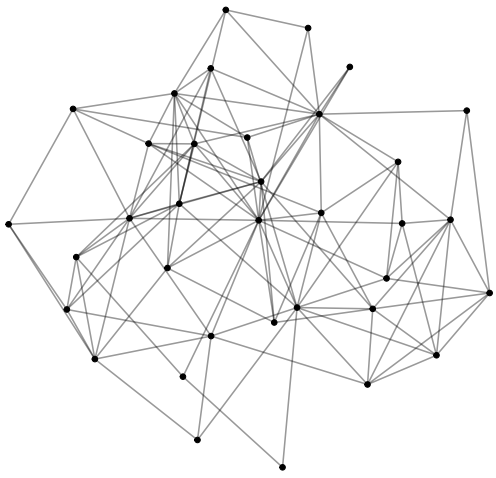
```
data(mpn_elite_mex)
```

Format

```
#> # A labelled, undirected network with 35 nodes and 117 ties
#> # A tibble: 35 x 8
#>   name    full_name       entry_year military in_mpn PlaceOfBirth state region
#>   <chr>   <chr>                <dbl>    <dbl>  <dbl> <chr>       <chr>  <dbl>
#> 1 Trevino  Trevino, Jacint~      1910        1      0 Guerrero    Coah~      1
#> 2 Madero   Madero, Francis~      1911        0      0 Parras de l~ Coah~      1
#> 3 Carranza Carranza, Venus~      1913        1      0 Cuatro Cien~ Coah~      1
#> 4 Aguilar  Aguilar, Candido      1918        1      0 Cordoba     Vera~      3
#> 5 Obregon  Obregon, Alvaro       1920        1      0 Siquisiva, ~ Sono~      1
#> 6 Calles   Calles, Plutarc~      1924        1      0 Guaymas     Sono~      1
#> # i 29 more rows
#> # A tibble: 117 x 2
#>    from    to
#>   <int> <int>
#> 1     2     3
#> 2     2     5
#> 3     2     6
#> 4     2     4
#> 5     1     2
#> 6     2     8
#> # i 111 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)
```

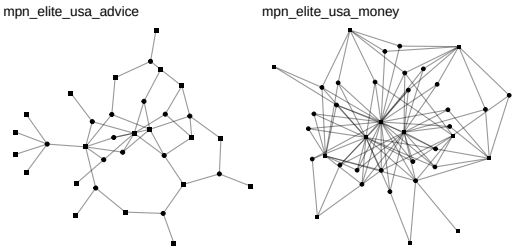
Format

```
#> # A labelled, two-mode network with 34 nodes and 46 ties
#> # A tibble: 34 x 2
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Albright
#> 2 FALSE Argyros
#> 3 FALSE Armitage
#> 4 FALSE Curry
#> 5 FALSE Fukuyama
#> 6 FALSE Gray
#> # i 28 more rows
#> # A tibble: 46 x 2
#>    from    to
#>   <int> <int>
```

```
#> 1     1    17
#> 2     1    19
#> 3     1    21
#> 4     2    22
#> 5     2    23
#> 6     2    27
#> # i 40 more rows

#> # A labelled, two-mode network with 38 nodes and 103 ties
#> # A tibble: 38 x 2
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Adkerson
#> 2 FALSE Akins
#> 3 FALSE Banga
#> 4 FALSE Boyce
#> 5 FALSE Britt
#> 6 FALSE Cannon
#> # i 32 more rows
#> # A tibble: 103 x 2
#>    from    to
#>   <int> <int>
#> 1     1    27
#> 2     1    28
#> 3     1    34
#> 4     1    36
#> 5     2    28
#> 6     2    32
#> # i 97 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 labelled, weighted, directed network with 20 nodes and 177 arcs
#> # A tibble: 20 x 1
#>   name
#>   <chr>
#> 1 1 AER
#> 2 2 AER
```

```
#> 3 5 AER/COR
#> 4 7 RYANAIR
#> 5 8 DG TRANSPORT
#> 6 9 COR
#> # i 14 more rows
#> # A tibble: 177 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1     2      1
#> 2     1     3      1
#> 3     1     4      1
#> 4     1     5      1
#> 5     1     6      1
#> 6     1     7      1
#> # i 171 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 labelled, weighted, two-mode network with 114 nodes and 2791 ties
#> # A tibble: 114 x 2
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Baucus
#> 2 FALSE Begich
#> 3 FALSE Bennet
#> 4 FALSE Blumenthal
#> 5 FALSE Boxer
#> 6 FALSE BrownSh
#> # i 108 more rows
#> # A tibble: 2,791 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1    52      1
#> 2     1    53      1
#> 3     1    54      1
#> 4     1    55      1
#> 5     1    56      1
#> 6     1    57      1
#> # i 2,785 more rows

#> # A labelled, weighted, two-mode network with 134 nodes and 3675 ties
#> # A tibble: 134 x 2
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Alexander
#> 2 FALSE Ayotte
#> 3 FALSE Barrasso
#> 4 FALSE Baucus
#> 5 FALSE Blunt
#> 6 FALSE Boozman
#> # i 128 more rows
#> # A tibble: 3,675 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1    64      1
#> 2     1    66      1
#> 3     1    67      1
#> 4     1    70      1
#> 5     1    71      1
#> 6     1    72      1
#> # i 3,669 more rows
```

```
#> # A labelled, weighted, two-mode network with 52 nodes and 614 ties
#> # A tibble: 52 x 2
#>   type  name
#>   <lgl> <chr>
#> 1 FALSE Baucus
#> 2 FALSE Cardin
#> 3 FALSE Carper
#> 4 FALSE Casey
#> 5 FALSE Collins
#> 6 FALSE Feinstein
#> # i 46 more rows
#> # A tibble: 614 x 3
#>    from    to weight
#>   <int> <int>  <dbl>
#> 1     1    21      1
#> 2     1    22      1
#> 3     1    23      1
#> 4     1    24      1
#> 5     1    25      1
#> 6     1    26      1
#> # i 608 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(.data)

network_triad_census(.data)

network_mixed_census(.data, object2)
```

Arguments

`.data`	An object of a {manynet}-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(manynet::ison_algebra)
network_triad_census(manynet::ison_adolescents)
marvel_friends <- manynet::to_unsigned(manynet::ison_marvel_relationships, "positive")
(mixed_cen <- network_mixed_census(marvel_friends, manynet::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.

Usage

```
node_tie_census(.data)

node_triad_census(.data)

node_quad_census(.data)

node_path_census(.data)
```

Arguments

`.data`	An object of a {manynet}-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 <- manynet::to_named(manynet::to_uniplex(manynet::ison_algebra, "task_tie"))
(tie_cen <- node_tie_census(task_eg))
(triad_cen <- node_triad_census(task_eg))
node_quad_census(manynet::ison_southern_women)
node_path_census(manynet::ison_adolescents)
node_path_census(manynet::ison_southern_women)
```

---

over *Helper functions for measuring over splits of networks*

Description

Helper functions for measuring over splits of networks

Usage

```
over_waves(
  .data,
  FUN,
  ...,
  attribute = "wave",
  strategy = "sequential",
  verbose = FALSE
)

over_time(
```

```
  .data,
  FUN,
  ...,
  attribute = "time",
  slice = NULL,
  strategy = "sequential",
  verbose = FALSE
)
```

Arguments

.data	An object of a {manynet}-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 function to run over all splits.
...	Further arguments to be passed on to FUN.
attribute	A string naming the attribute to be split upon.
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.
slice	Optionally, a vector of specific slices. Otherwise all observed slices will be returned.

Functions

- over_waves(): Runs a function, e.g. a measure, over waves of a panel network
- over_time(): Runs a function, e.g. a measure, over time slices of a dynamic network

---

play *Functions to play games on networks*

Description

Functions to play games on networks

Usage

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

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

play_learning(.data, beliefs, steps, epsilon = 5e-04)

play_segregation(
  .data,
  attribute,
  heterophily = 0,
  who_moves = c("ordered", "random", "most_dissatisfied"),
  choice_function = c("satisficing", "optimising"),
  steps
)
```

Arguments

`.data` An object of a {manynet}-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.
attribute	A string naming some nodal attribute in the network. Currently only tested for binary attributes.
heterophily	A score ranging between -1 and 1 as a threshold for how heterophilous nodes will accept their neighbours to be. A single proportion means this threshold is shared by all nodes, but it can also be a vector the same length of the nodes in the network for issuing different thresholds to different nodes. By default this is 0, meaning nodes will be dissatisfied if more than half of their neighbours differ on the given attribute.
who_moves	One of the following options: "ordered" (the default) checks each node in turn for whether they are dissatisfied and there is an available space that they can move to, "random" will check a node at random, and "most_dissatisfied" will check (one of) the most dissatisfied nodes first.
choice_function	One of the following options: "satisficing" (the default) will move the node to any coordinates that satisfy their heterophily threshold, whereas "optimising" will move the node to coordinates that are most homophilous.

Functions

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

See Also

Other models: `regression`, `tests`

Examples

```
smeg <- manynet::generate_smallworld(15, 0.025)
plot(play_diffusion(smeg))
plot(play_diffusion(smeg, recovery = 0.4))
plot(play_diffusions(smeg, times = 20))
play_learning(ison_networkers,
    rbinom(manynet::network_nodes(ison_networkers),1,prob = 0.25))
startValues <- rbinom(100,1,prob = 0.5)
startValues[sample(seq_len(100), round(100*0.2))] <- NA
latticeEg <- manynet::create_lattice(100)
latticeEg <- manynet::add_node_attribute(latticeEg, "startValues", startValues)
latticeEg
play_segregation(latticeEg, "startValues", 0.5)
#manynet::autographr(latticeEg, node_color = "startValues", node_size = 5)
#manynet::autographr(play_segregation(latticeEg, "startValues", 0.5),
#                    node_color = "startValues", node_size = 5)
```

`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,
  .data,
  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

`.data` An object of a {manynet}-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)
```

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

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

Arguments

`.data`	An object of a {manynet}-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_heterophily, attribute = "Attractive",
  times = 200))
plot(cugtest)
(qaptest <- test_permutation(marvel_friends,
                network_heterophily, attribute = "Attractive",
                times = 200))
plot(qaptest)
```

Index