Package 'netrics'

March 26, 2026

Title Many Ways to Measure and Classify Membership for Networks, Nodes, and Ties

Version 0.2.0

Date 2026-03-21

Description Many tools for calculating network, node, or tie marks, measures, motifs and memberships of many different types of networks. Marks identify structural positions, measures quantify network properties, memberships classify nodes into groups, and motifs tabulate substructure participation. All functions operate with all classes of network data covered in 'manynet', and on directed, undirected, multiplex, multimodal, signed, and other networks.

URL https://stocnet.github.io/netrics/

BugReports https://github.com/stocnet/netrics/issues

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.6.0), manynet

Imports dplyr, igraph (>= 2.1.0)

Suggests autograph, sna, testthat (>= 3.0.0)

Config/Needs/build roxygen2, devtools

Config/Needs/check covr, lintr, spelling

Config/Needs/website pkgdown

Config/testthat/parallel true

Config/testthat/edition 3

Config/testthat/start-first measure_net

NeedsCompilation no

Author James Hollway [cre, aut, ctb] (IHEID, ORCID: <https://orcid.org/0000-0002-8361-9647>)

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

Repository CRAN

Date/Publication 2026-03-26 09:30:09 UTC

Contents

`mark_core` *Marking nodes as core or periphery*

Description

`node_is_core()` identifies whether nodes belong to the core of the network, as opposed to the periphery.

Usage

```
node_is_core(.data, centrality = c("degree", "eigenvector"))
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`centrality`	Which centrality measure 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.

Value

A `node_mark` logical vector the length of the nodes in the network, giving either TRUE or FALSE for each node depending on whether the condition is matched.

Core-periphery

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.

References

On core-periphery partitioning:

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 core-periphery: measure_core, member_core

Other marks: mark_degree, mark_diff, mark_dyads, mark_nodes, mark_select_node, mark_select_tie, mark_ties, mark_triangles

Other nodal: mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_is_core(ison_adolescents)
ison_adolescents %>%
   mutate(corep = node_is_core())
```

---

mark_degree *Marking nodes based on degree properties*

Description

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

- `node_is_isolate()` marks nodes that are isolates, with neither incoming nor outgoing ties.
- `node_is_pendant()` marks nodes that are pendants, with exactly one incoming or outgoing tie.
- `node_is_universal()` identifies whether nodes are adjacent to all other nodes in the network.

Usage

```
node_is_isolate(.data)

node_is_pendant(.data)

node_is_universal(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `node_mark` logical vector the length of the nodes in the network, giving either TRUE or FALSE for each node depending on whether the condition is matched.

Universal/dominating node

A universal node is adjacent to all other nodes in the network. It is also sometimes called the dominating vertex because it represents a one-element dominating set. A network with a universal node is called a cone, and its universal node is called the apex of the cone. A classic example of a cone is a star graph, but friendship, wheel, and threshold graphs are also cones.

See Also

Other degree: `measure_central_degree`, `measure_centralisation_degree`, `measure_centralities_degree`

Other marks: `mark_core`, `mark_diff`, `mark_dyads`, `mark_nodes`, `mark_select_node`, `mark_select_tie`, `mark_ties`, `mark_triangles`

Other nodal: `mark_core`, `mark_diff`, `mark_nodes`, `mark_select_node`, `measure_assort_node`, `measure_broker_node`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_closure_node`, `measure_core`, `measure_diffusion_node`, `measure_diverse_node`, `member_brokerage`, `member_cliques`, `member_community`, `member_community_hier`, `member_community_non`, `member_components`, `member_core`, `member_diffusion`, `member_equivalence`, `motif_brokerage_node`, `motif_exposure`, `motif_node`, `motif_path`

Examples

```
node_is_isolate(ison_brandes)
node_is_universal(create_star(11))
```

`mark_diff` *Marking nodes based on diffusion properties*

Description

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

- `node_is_infected()` marks nodes that are infected by a particular time point.
- `node_is_exposed()` marks nodes that are exposed to a given (other) mark.
- `node_is_latent()` marks nodes that are latent at a particular time point.
- `node_is_recovered()` marks nodes that are recovered at a particular time point.

Usage

```
node_is_latent(.data, time = 0)

node_is_infected(.data, time = 0)

node_is_recovered(.data, time = 0)

node_is_exposed(.data, mark, time = 0)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`time`	A time step at which nodes are identified.
`mark`	vector denoting which nodes are infected

Value

A `node_mark` logical vector the length of the nodes in the network, giving either TRUE or FALSE for each node depending on whether the condition is matched.

Exposed

`node_is_exposed()` is similar to `node_exposure()`, but returns a mark (TRUE/FALSE) vector indicating which nodes are currently exposed to the diffusion content. This diffusion content can be expressed in the 'mark' argument. If no 'mark' argument is provided, and '.data' is a diff_model object, then the function will return nodes exposure to the seed nodes in that diffusion.

See Also

Other marks: mark_core, mark_degree, mark_dyads, mark_nodes, mark_select_node, mark_select_tie, mark_ties, mark_triangles

Other nodal: mark_core, mark_degree, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Other diffusion: measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, member_diffusion, motif_exposure, motif_hazard

Examples

```
# To mark nodes that are latent by a particular time point
node_is_latent(play_diffusion(create_tree(6), latency = 1), time = 1)
# To mark nodes that are infected by a particular time point
node_is_infected(play_diffusion(create_tree(6)), time = 1)
# To mark nodes that are recovered by a particular time point
node_is_recovered(play_diffusion(create_tree(6), recovery = 0.5), time = 3)
# To mark which nodes are currently exposed
(expos <- node_is_exposed(manynet::create_tree(14), mark = c(1,3)))
which(expos)
```

---

mark_dyads *Marking ties based on dyadic properties*

Description

These functions return logical vectors the length of the ties in a network identifying which are embedded within particular dyads.

- `tie_is_multiple()` marks ties that are multiples.
- `tie_is_reciprocated()` marks ties that are mutual/reciprocated.

They are most useful in highlighting parts of the network where relationships are denser.

Usage

```
tie_is_multiple(.data)

tie_is_reciprocated(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `tie_mark` logical vector the length of the ties in the network, giving either TRUE or FALSE for each tie depending on whether the condition is matched.

See Also

Other marks: `mark_core`, `mark_degree`, `mark_diff`, `mark_nodes`, `mark_select_node`, `mark_select_tie`, `mark_ties`, `mark_triangles`

Other tie: `mark_select_tie`, `mark_ties`, `mark_triangles`, `measure_broker_tie`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`

Examples

```
tie_is_multiple(fict_marvel)
tie_is_reciprocated(ison_algebra)
```

---

mark_nodes *Marking nodes based on structural properties*

Description

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

- `node_is_independent()` marks nodes that are members of the largest independent set, aka largest internally stable set.
- `node_is_cutpoint()` marks nodes that cut or act as articulation points in a network, increasing the number of connected components when removed.
- `node_is_fold()` marks nodes that are in a structural fold between two or more triangles that are only connected by that node.
- `node_is_mentor()` marks a proportion of high indegree nodes as 'mentors' (see details).
- `node_is_neighbor()` marks nodes that are neighbours of a given node.

Usage

```
node_is_independent(.data)

node_is_cutpoint(.data)

node_is_fold(.data)

node_is_mentor(.data, elites = 0.1)

node_is_neighbor(.data, node)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
elites	The proportion of nodes to be selected as mentors. By default this is set at 0.1. This means that the top 10% of nodes in terms of degree, or those equal to the highest rank degree in the network, whichever is the higher, will be used to select the mentors. Note that if nodes are equidistant from two mentors, they will choose one at random. If a node is without a path to a mentor, for example because they are an isolate, a tie to themselves (a loop) will be created instead. Note that this is a different default behaviour than that described in Valente and Davis (1999).
node	A node ID or name for which neighbors are to be marked.

Value

A node_mark logical vector the length of the nodes in the network, giving either TRUE or FALSE for each node depending on whether the condition is matched.

References

On independent sets:

Tsukiyama, Shuji, Mikio Ide, Hiromu Ariyoshi, and Isao Shirawaka. 1977. "A new algorithm for generating all the maximal independent sets". *SIAM Journal on Computing*, 6(3):505–517. doi:10.1137/0206036

On articulation or cut-points:

Tarjan, Robert E. and Uzi Vishkin. 1985. "An Efficient Parallel Biconnectivity Algorithm", *SIAM Journal on Computing* 14(4): 862-874. doi:10.1137/0214061

On structural folds:

Vedres, Balazs, and David Stark. 2010. "Structural folds: Generative disruption in overlapping groups", *American Journal of Sociology* 115(4): 1150-1190. doi:10.1086/649497

On mentoring:

Valente, Thomas, and Rebecca Davis. 1999. "Accelerating the Diffusion of Innovations Using Opinion Leaders", *Annals of the American Academy of Political and Social Science* 566: 56-67.

See Also

Other marks: mark_core, mark_degree, mark_diff, mark_dyads, mark_select_node, mark_select_tie, mark_ties, mark_triangles

Other nodal: mark_core, mark_degree, mark_diff, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_is_independent(ison_adolescents)
node_is_cutpoint(ison_brandes)
node_is_fold(create_explicit(A-B, B-C, A-C, C-D, C-E, D-E))
```

`mark_select_node` *Marking nodes based on measures*

Description

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

- `node_is_random()` marks one or more nodes at random.
- `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_random(.data, select = 1)

node_is_max(node_measure, ranks = 1)

node_is_min(node_measure, ranks = 1)

node_is_mean(node_measure, ranks = 1)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`select`	Number of elements 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`.

Value

A `node_mark` logical vector the length of the nodes in the network, giving either TRUE or FALSE for each node depending on whether the condition is matched.

See Also

Other selection: mark_select_tie

Other marks: mark_core, mark_degree, mark_diff, mark_dyads, mark_nodes, mark_select_tie, mark_ties, mark_triangles

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_is_random(ison_brandes, 2)
node_is_max(node_by_degree(ison_brandes))
node_is_min(node_by_degree(ison_brandes))
node_is_mean(node_by_degree(ison_brandes))
```

---

mark_select_tie *Marking ties based on measures*

Description

These functions return logical vectors the length of the ties in a network:

- tie_is_random() marks one or more ties at random.
- tie_is_max() and tie_is_min() are more useful for converting the results from some tie measure into a mark-class object. They can be particularly useful for highlighting which tie or ties are key because they minimise or, more often, maximise some measure.

Usage

```
tie_is_random(.data, select = 1)

tie_is_max(tie_measure)

tie_is_min(tie_measure)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
select	Number of elements to select (as TRUE).
tie_measure	An object created by a tie_ measure.

Value

A `tie_mark` logical vector the length of the ties in the network, giving either TRUE or FALSE for each tie depending on whether the condition is matched.

See Also

Other marks: `mark_core`, `mark_degree`, `mark_diff`, `mark_dyads`, `mark_nodes`, `mark_select_node`, `mark_ties`, `mark_triangles`

Other tie: `mark_dyads`, `mark_ties`, `mark_triangles`, `measure_broker_tie`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`

Other selection: `mark_select_node`

Examples

```
tie_is_max(tie_by_betweenness(ison_brandes))
tie_is_min(tie_by_betweenness(ison_brandes))
```

---

mark_ties *Marking ties based on structural properties*

Description

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

- `tie_is_loop()` marks ties that are loops.
- `tie_is_feedback()` marks ties that are feedback arcs causing the network to not be acyclic.
- `tie_is_bridge()` marks ties that cut or act as articulation points in a network.
- `tie_is_path()` marks ties on a path from one node to another.

They are most useful in highlighting parts of the network that are particularly well- or poorly-connected.

Usage

```
tie_is_loop(.data)

tie_is_feedback(.data)

tie_is_bridge(.data)

tie_is_path(.data, from, to, all_paths = FALSE)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
from	The index or name of the node from which the path should start.
to	The index or name of the node to which the path should be traced.
all_paths	Whether to return a list of paths or sample just one. By default FALSE, sampling just a single path.

Value

A tie_mark logical vector the length of the ties in the network, giving either TRUE or FALSE for each tie depending on whether the condition is matched.

See Also

Other marks: mark_core, mark_degree, mark_diff, mark_dyads, mark_nodes, mark_select_node, mark_select_tie, mark_triangles

Other tie: mark_dyads, mark_select_tie, mark_triangles, measure_broker_tie, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Examples

```
tie_is_loop(fict_marvel)
tie_is_feedback(ison_algebra)
tie_is_bridge(ison_brandes)
ison_adolescents %>%
  mutate_ties(route = tie_is_path(from = "Jane", to = 7))
```

---

mark_triangles *Marking ties based on triangular properties*

Description

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

- tie_is_triangular() marks ties that are part of triangles.
- tie_is_cyclical() marks ties that are part of cycles.
- tie_is_triplet() marks ties that are part of transitive triplets.
- tie_is_simmelian() marks ties that are both in a triangle and fully reciprocated.
- tie_is_imbalanced() marks ties that are part of imbalanced triads.
- tie_is_transitive() marks ties that complete transitive closure.

They are most useful in highlighting parts of the network that are cohesively connected.

Usage

```
tie_is_triangular(.data)

tie_is_transitive(.data)

tie_is_triplet(.data)

tie_is_cyclical(.data)

tie_is_simmelian(.data)

tie_is_imbalanced(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A tie_mark logical vector the length of the ties in the network, giving either TRUE or FALSE for each tie depending on whether the condition is matched.

See Also

Other marks: mark_core, mark_degree, mark_diff, mark_dyads, mark_nodes, mark_select_node, mark_select_tie, mark_ties

Other tie: mark_dyads, mark_select_tie, mark_ties, measure_broker_tie, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other cohesion: measure_breadth, measure_cohesion, measure_fragmentation, motif_net, motif_node

Examples

```
ison_monks %>% to_uniplex("like") %>%
  mutate_ties(tri = tie_is_triangular())
ison_adolescents %>% to_directed() %>%
  mutate_ties(trans = tie_is_transitive())
ison_adolescents %>% to_directed() %>%
  mutate_ties(trip = tie_is_triplet())
ison_adolescents %>% to_directed() %>%
  mutate_ties(cyc = tie_is_cyclical())
ison_monks %>% to_uniplex("like") %>%
  mutate_ties(simmel = tie_is_simmelian())
fict_marvel %>% to_uniplex("relationship") %>% tie_is_imbalanced()
```

`measure_assort_net` *Measures of network assortativity*

Description

These functions offer ways to measure the distribution or assortativity of ties in a network:

- `net_by_heterophily()` measures how embedded nodes in the network are within groups of nodes with the same attribute.
- `net_by_homophily()` measures how embedded nodes in the network are within groups of nodes with the same attribute, but with more options for method.
- `net_by_assortativity()` measures the degree assortativity in a network.
- `net_by_spatial()` measures the spatial association/autocorrelation (global Moran's I) in a network.

Note that for two-mode networks, homophily is calculated on the mode with the attribute of interest, and the other mode is ignored. If the attribute is present on both modes, homophily is calculated on the first mode by default, but a message is given and the user can choose to calculate homophily on the other mode instead by subsetting the attribute vector and converting the network to a one-mode network.

Usage

```
net_by_heterophily(.data, attribute)

net_by_homophily(
  .data,
  attribute,
  assortativity = c("ie", "ei", "yule", "geary")
)

net_by_assortativity(.data)

net_by_spatial(.data, attribute)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`attribute`	Name of a nodal attribute, mark, measure, or membership vector.
`assortativity`	Which method to use for `*_homophily()`. Either "ie" (negative E-I index), "ei" (E-I index), "yule" (Yule's Q), or "geary" (Geary's C). Default is "ie". 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.

Yule's Q is a measure of association for two binary variables, calculated as:

$$Q = \frac{ad - bc}{ad + bc}$$

where a is the number of ties between nodes in the same category, b is the number of ties between nodes in different categories, c is the number of non-ties between nodes in the same category, and d is the number of non-ties between nodes in different categories. This value can range from -1 to 1, where 1 indicates perfect association (all ties are between nodes in the same category), -1 indicates perfect disassociation (all ties are between nodes in different categories), and 0 indicates no association.

Geary's C is a measure of spatial autocorrelation, calculated as:

$$C = \frac{(n-1)\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - x_j)^2}{2W\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where n is the number of nodes, w_{ij} is the weight of the tie between nodes i and j, x_i is the attribute value of node i, $\bar{x}$ is the mean attribute value across all nodes, and W is the sum of all tie weights. This value can range from 0 to 2, where values less than 1 indicate positive autocorrelation (similar values are more likely to be connected), values greater than 1 indicate negative autocorrelation (dissimilar values are more likely to be connected), and a value of 1 indicates no autocorrelation. If an incompatible method is chosen for the attribute type, a suitable alternative will be used instead with a message.

Value

A `network_measure` numeric score.

Heterophily

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

On heterophily:

Krackhardt, David, and Robert N. Stern. 1988. Informal networks and organizational crises: an experimental simulation. *Social Psychology Quarterly* 51(2): 123-140. doi:10.2307/2786835

McPherson, Miller, Lynn Smith-Lovin, and James M. Cook. 2001. "Birds of a Feather: Homophily in Social Networks". *Annual Review of Sociology*, 27(1): 415-444. doi:10.1146/annurev.soc.27.1.415

On assortativity:

Newman, Mark E.J. 2002. "Assortative mixing in networks". *Physical Review Letters*, 89(20): 208701. doi:10.1103/physrevlett.89.208701

On spatial autocorrelation:

Moran, Patrick Alfred Pierce. 1950. "Notes on continuous stochastic phenomena". *Biometrika* 37(1): 17-23. doi:10.2307/2332142

See Also

Other diversity: measure_assort_node, measure_diverse_net, measure_diverse_node

Other measures: measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
marvel_friends <- to_unsigned(to_uniplex(fict_marvel, "relationship"), "positive")
net_by_heterophily(marvel_friends, "Gender")
net_by_heterophily(marvel_friends, "Attractive")
net_by_homophily(marvel_friends, "Gender")
net_by_assortativity(ison_networkers)
net_by_spatial(ison_lawfirm, "age")
```

---

measure_assort_node *Measures of nodes assortativity*

Description

These functions offer ways to measure nodes' assortativity in a network:

- node_by_heterophily() measures each node's embeddedness within groups of nodes with the same attribute.
- node_by_homophily() measures each node's embeddedness within groups of nodes with the same attribute, using a variety of methods.

Usage

```
node_by_heterophily(.data, attribute)

node_by_homophily(
  .data,
  attribute,
  assortativity = c("ie", "ei", "yule", "geary")
)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`attribute`	Name of a nodal attribute, mark, measure, or membership vector.
`assortativity`	Which method to use for `*_homophily()`. Either "ie" (negative E-I index), "ei" (E-I index), "yule" (Yule's Q), or "geary" (Geary's C). Default is "ie". 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. Yule's Q is a measure of association for two binary variables, calculated as: $$Q = \frac{ad - bc}{ad + bc}$$ where a is the number of ties between nodes in the same category, b is the number of ties between nodes in different categories, c is the number of non-ties between nodes in the same category, and d is the number of non-ties between nodes in different categories. This value can range from -1 to 1, where 1 indicates perfect association (all ties are between nodes in the same category), -1 indicates perfect disassociation (all ties are between nodes in different categories), and 0 indicates no association. Geary's C is a measure of spatial autocorrelation, calculated as: $$C = \frac{(n-1)\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - x_j)^2}{2W\sum_{i=1}^{n}(x_i - \bar{x})^2}$$ where n is the number of nodes, w_{ij} is the weight of the tie between nodes i and j, x_i is the attribute value of node i, $\bar{x}$ is the mean attribute value across all nodes, and W is the sum of all tie weights. This value can range from 0 to 2, where values less than 1 indicate positive autocorrelation (similar values are more likely to be connected), values greater than 1 indicate negative autocorrelation (dissimilar values are more likely to be connected), and a value of 1 indicates no autocorrelation. If an incompatible method is chosen for the attribute type, a suitable alternative will be used instead with a message.

Value

A `node_measure` numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

See Also

Other diversity: `measure_assort_net`, `measure_diverse_net`, `measure_diverse_node`

Other measures: `measure_assort_net`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`,

measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
marvel_friends <- to_unsigned(to_uniplex(fict_marvel, "relationship"), "positive")
node_by_heterophily(marvel_friends, "Gender")
node_by_heterophily(marvel_friends, "Attractive")
```

---

measure_breadth *Measures of network breadth*

Description

These functions return values or vectors relating to how broad a network is.

- `net_by_diameter()` measures the maximum path length in the network.
- `net_by_length()` measures the average path length in the network.

Usage

```
net_by_diameter(.data)

net_by_length(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `network_measure` numeric score.

See Also

Other cohesion: mark_triangles, measure_cohesion, measure_fragmentation, motif_net, motif_node

Other measures: measure_assort_net, measure_assort_node, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
net_by_diameter(fict_marvel)
net_by_diameter(to_giant(fict_marvel))
net_by_length(fict_marvel)
net_by_length(to_giant(fict_marvel))
```

---

measure_brokerage *Measures of brokerage*

Description

These functions include ways to measure nodes' brokerage activity and exclusivity in a network:

- `node_by_brokering_activity()` measures nodes' brokerage activity.
- `node_by_brokering_exclusivity()` measures nodes' brokerage exclusivity.

Usage

```
node_by_brokering_activity(.data, membership)

node_by_brokering_exclusivity(.data, membership)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`membership`	A character string naming an existing node attribute in the network, or a categorical vector of the same length as the number of nodes in the network where each element indicates the group membership of the corresponding node. While this may often be a vector created using `node_in_*()` functions, it can be any character vector that assigns nodes to groups or categories.

Value

A node_measure numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

References

On brokerage activity and exclusivity:

Hamilton, Matthew, Jacob Hileman, and Orjan Bodin. 2020. "Evaluating heterogeneous brokerage: New conceptual and methodological approaches and their application to multi-level environmental governance networks" *Social Networks* 61: 1-10. doi:10.1016/j.socnet.2019.08.002

See Also

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Other brokerage: measure_broker_node, measure_broker_tie, member_brokerage, motif_brokerage_net, motif_brokerage_node

Examples

```
node_by_brokering_exclusivity(ison_networkers, "Discipline")
```

---

measure_broker_node *Measuring nodes brokerage*

Description

These function provide different measures of the degree to which nodes fill structural holes, as outlined in Burt (1992):

- node_by_bridges() measures the sum of bridges to which each node is adjacent.
- node_by_redundancy() measures the redundancy of each nodes' contacts.
- node_by_effsize() measures nodes' effective size.

- `node_by_efficiency()` measures nodes' efficiency.
- `node_by_constraint()` measures nodes' constraint scores for one-mode networks according to Burt (1992) and for two-mode networks according to Hollway et al (2020).
- `node_by_hierarchy()` measures nodes' exposure to hierarchy, where only one or two contacts are the source of closure.
- `node_by_neighbours_degree()` measures nodes' average nearest neighbors degree, or knn, a measure of the type of local environment a node finds itself in

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_by_bridges(.data)

node_by_redundancy(.data)

node_by_effsize(.data)

node_by_efficiency(.data)

node_by_constraint(.data)

node_by_hierarchy(.data)

node_by_neighbours_degree(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

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.

Value

A `node_measure` numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

References

On structural holes:

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.

Burchard, Jake, and Benjamin Cornwell. 2018. "Structural Holes and Bridging in Two-Mode Networks." *Social Networks* 55:11–20. doi:10.1016/j.socnet.2018.04.001

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

On neighbours average degree:

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 brokerage: measure_broker_tie, measure_brokerage, member_brokerage, motif_brokerage_net, motif_brokerage_node

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_bridges(ison_adolescents)
node_by_bridges(ison_southern_women)
node_by_redundancy(ison_adolescents)
node_by_redundancy(ison_southern_women)
node_by_effsize(ison_adolescents)
node_by_effsize(ison_southern_women)
node_by_efficiency(ison_adolescents)
node_by_efficiency(ison_southern_women)
node_by_constraint(ison_southern_women)
```

```
node_by_hierarchy(ison_adolescents)
node_by_hierarchy(ison_southern_women)
```

`measure_broker_tie` *Measuring ties brokerage*

Description

`tie_by_cohesion()` measures 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.

Usage

```
tie_by_cohesion(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `tie_measure` numeric vector the length of the ties in the network, providing the scores for each tie. If the network is labelled, then the scores will be labelled with the ties' adjacent nodes' names.

See Also

Other brokerage: `measure_broker_node`, `measure_brokerage`, `member_brokerage`, `motif_brokerage_net`, `motif_brokerage_node`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_net`, `measure_diverse_node`, `measure_features`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Other tie: `mark_dyads`, `mark_select_tie`, `mark_ties`, `mark_triangles`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`

`measure_centralisation_between` *Measuring networks betweenness-like centralisation*

Description

`net_by_betweenness()` measures the betweenness centralization for a network.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`normalized`	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
`direction`	Character string, "out" bases the measure on outgoing ties, "in" on incoming ties, and "all" on either/the sum of the two. By default "all".

Value

A `network_measure` numeric score.

See Also

Other betweenness: `measure_central_between`, `measure_centralities_between`

Other centrality: `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_close`, `measure_centralisation_deg` `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_net`, `measure_diverse_node`, `measure_features`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Examples

```
net_by_betweenness(ison_southern_women, direction = "in")
```

`measure_centralisation_close`

Measuring networks closeness-like centralisation

Description

- `net_by_closeness()` measures a network's closeness centralization.
- `net_by_reach()` measures a network's reach centralization.
- `net_by_harmonic()` measures a network's harmonic centralization.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

net_by_reach(.data, normalized = TRUE, cutoff = 2)

net_by_harmonic(.data, normalized = TRUE, cutoff = 2)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`normalized`	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
`direction`	Character string, "out" bases the measure on outgoing ties, "in" on incoming ties, and "all" on either/the sum of the two. By default "all".
`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.

Value

A `network_measure` numeric score.

See Also

Other closeness: measure_central_close, measure_centralities_close

Other centrality: measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c
measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
net_by_closeness(ison_southern_women, direction = "in")
```

`measure_centralisation_degree`

Measuring networks degree-like centralisation

Description

- `net_by_degree()` measures a network's degree centralization; there are several related shortcut functions:
  - `net_by_indegree()` returns the `direction = 'in'` results.
  - `net_by_outdegree()` returns the `direction = 'out'` results.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

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.

Usage

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

net_by_outdegree(.data, normalized = TRUE)

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

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
direction	Character string, "out" bases the measure on outgoing ties, "in" on incoming ties, and "all" on either/the sum of the two. By default "all".

Value

A network_measure numeric score.

See Also

Other degree: mark_degree, measure_central_degree, measure_centralities_degree

Other centrality: measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c
measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
net_by_degree(ison_southern_women, direction = "in")
```

---

measure_centralisation_eigen

Measuring networks eigenvector-like centralisation

Description

net_by_eigenvector() measures the eigenvector centralization for a network.

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. manynet::to_undirected() functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A network_measure numeric score.

See Also

Other eigenvector: measure_central_eigen, measure_centralities_eigen

Other centrality: measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c
measure_centralisation_degree, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
net_by_eigenvector(ison_southern_women)
```

---

measure_centralities_between

Measuring ties betweenness-like centrality

Description

tie_by_betweenness() measures the number of shortest paths going through a tie.

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. manynet::to_undirected() functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A tie_measure numeric vector the length of the ties in the network, providing the scores for each tie. If the network is labelled, then the scores will be labelled with the ties' adjacent nodes' names.

See Also

Other betweenness: measure_central_between, measure_centralisation_between

Other centrality: measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c
measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other tie: mark_dyads, mark_select_tie, mark_ties, mark_triangles, measure_broker_tie, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Examples

```
(tb <- tie_by_betweenness(ison_adolescents))
ison_adolescents %>% mutate_ties(weight = tb)
```

---

measure_centralities_close

Measuring ties closeness-like centrality

Description

`tie_by_closeness()` measures the closeness of each tie to other ties in the network.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`normalized`	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A `tie_measure` numeric vector the length of the ties in the network, providing the scores for each tie. If the network is labelled, then the scores will be labelled with the ties' adjacent nodes' names.

See Also

Other closeness: `measure_central_close`, `measure_centralisation_close`

Other centrality: `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_degree`, `measure_centralities_eigen`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_net`, `measure_diverse_node`, `measure_features`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Other tie: `mark_dyads`, `mark_select_tie`, `mark_ties`, `mark_triangles`, `measure_broker_tie`, `measure_centralities_between`, `measure_centralities_degree`, `measure_centralities_eigen`

Examples

```
(ec <- tie_by_closeness(ison_adolescents))
ison_adolescents %>% mutate_ties(weight = ec)
```

`measure_centralities_degree` *Measuring ties degree-like centrality*

Description

`tie_by_degree()` measures the degree centrality of ties in a network

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`normalized`	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A `tie_measure` numeric vector the length of the ties in the network, providing the scores for each tie. If the network is labelled, then the scores will be labelled with the ties' adjacent nodes' names.

See Also

Other degree: `mark_degree`, `measure_central_degree`, `measure_centralisation_degree`

Other centrality: `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_eigen`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c`
`measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_net`, `measure_diverse_node`, `measure_features`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Other tie: `mark_dyads`, `mark_select_tie`, `mark_ties`, `mark_triangles`, `measure_broker_tie`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_eigen`

Examples

```
tie_by_degree(ison_adolescents)
```

`measure_centralities_eigen`

Measuring ties eigenvector-like centrality

Description

`tie_by_eigenvector()` measures the eigenvector centrality of ties in a network.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`normalized`	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A `tie_measure` numeric vector the length of the ties in the network, providing the scores for each tie. If the network is labelled, then the scores will be labelled with the ties' adjacent nodes' names.

See Also

Other eigenvector: `measure_central_eigen`, `measure_centralisation_eigen`

Other centrality: `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c`
`measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`,

measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other tie: mark_dyads, mark_select_tie, mark_ties, mark_triangles, measure_broker_tie, measure_centralities_between, measure_centralities_close, measure_centralities_degree

Examples

```
tie_by_eigenvector(ison_adolescents)
```

---

measure_central_between

Measuring nodes betweenness-like centrality

Description

These functions calculate common betweenness-related centrality measures for one- and two-mode networks:

- `node_by_betweenness()` measures the betweenness centralities of nodes in a network.
- `node_by_induced()` measures the induced betweenness centralities of nodes in a network.
- `node_by_flow()` measures the flow betweenness centralities of nodes in a network, which uses an electrical current model for information spreading in contrast to the shortest paths model used by normal betweenness centrality.
- `node_by_stress()` measures the stress centrality of nodes in a network.
- `tie_by_betweenness()` measures the number of shortest paths going through a tie.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

node_by_induced(.data, normalized = TRUE, cutoff = NULL)

node_by_flow(.data, normalized = TRUE)

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`normalized`	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
`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.

Value

A `node_measure` numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

Betweenness centrality

Betweenness centrality is based on the number of shortest paths between other nodes that a node lies upon:

$$C_B(i) = \sum_{j,k:j\neq k,j\neq i,k\neq i} \frac{g_{jik}}{g_{jk}}$$

Induced centrality

Induced centrality or vitality centrality concerns the change in total betweenness centrality between networks with and without a given node:

$$C_I(i) = C_B(G) - C_B(G\ i)$$

Flow betweenness centrality

Flow betweenness centrality concerns the total maximum flow, f, between other nodes j, k in a network G that a given node mediates:

$$C_F(i) = \sum_{j,k:j\neq k,j\neq i,k\neq i} f(j,k,G) - f(j,k,G\ i)$$

When normalized (by default) this sum of differences is divided by the sum of flows $f(i,j,G)$.

Stress centrality

Stress centrality is the number of all shortest paths or geodesics, g, between other nodes that a given node mediates:

$$C_S(i) = \sum_{j,k:j\neq k,j\neq i,k\neq i} g_{jik}$$

High stress nodes lie on a large number of shortest paths between other nodes, and thus associated with bridging or spanning boundaries.

References

On betweenness centrality:

Freeman, Linton. 1977. "A set of measures of centrality based on betweenness". *Sociometry*, 40(1): 35–41. doi:10.2307/3033543

On induced centrality:

Everett, Martin and Steve Borgatti. 2010. "Induced, endogenous and exogenous centrality" *Social Networks*, 32: 339-344. doi:10.1016/j.socnet.2010.06.004

On flow centrality:

Freeman, Lin, Stephen Borgatti, and Douglas White. 1991. "Centrality in Valued Graphs: A Measure of Betweenness Based on Network Flow". *Social Networks*, 13(2), 141-154.

Koschutzki, D., K.A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. 2005. "Centrality Indices". In U. Brandes and T. Erlebach (eds.), *Network Analysis: Methodological Foundations*. Berlin: Springer.

On stress centrality:

Shimbel, A. 1953. "Structural Parameters of Communication Networks". *Bulletin of Mathematical Biophysics*, 15:501-507. doi:10.1007/BF02476438

See Also

Other betweenness: measure_centralisation_between, measure_centralities_between

Other centrality: measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_betweenness(ison_southern_women)
node_by_induced(ison_adolescents)
```

`measure_central_close` *Measuring nodes closeness-like centrality*

Description

These functions calculate common closeness-related centrality measures that rely on path-length for one- and two-mode networks:

- `node_by_closeness()` measures the closeness centrality of nodes in a network.
- `node_by_harmonic()` measures nodes' harmonic centrality or valued centrality, which is thought to behave better than reach centrality for disconnected networks.
- `node_by_reach()` measures nodes' reach centrality, or how many nodes they can reach within *k* steps.
- `node_by_information()` measures nodes' information centrality or current-flow closeness centrality.
- `node_by_eccentricity()` measures nodes' eccentricity or maximum distance from another node in the network.
- `node_by_distance()` measures nodes' geodesic distance from or to a given node.
- `node_by_vitality()` measures a network's closeness vitality centrality, or the change in closeness centrality between networks with and without a given node.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

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

node_by_harmonic(.data, normalized = TRUE, cutoff = -1)

node_by_reach(.data, normalized = TRUE, cutoff = 2)

node_by_information(.data, normalized = TRUE)

node_by_eccentricity(.data, normalized = TRUE)

node_by_distance(.data, from, to, normalized = TRUE)

node_by_vitality(.data, normalized = TRUE)

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

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
direction	Character string, "out" bases the measure on outgoing ties, "in" on incoming ties, and "all" on either/the sum of the two. By default "all".
cutoff	Maximum path length to use during calculations.
from, to	Index or name of a node to calculate distances from or to.

Value

A node_measure numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

Closeness centrality

Closeness centrality, status centrality, or barycenter centrality is defined as the reciprocal of the farness or distance, d, from a node to all other nodes in the network:

$$C_C(i) = \frac{1}{\sum_j d(i,j)}$$

When (more commonly) normalised, the numerator is instead $N-1$.

Harmonic centrality

Harmonic centrality or valued centrality reverses the sum and reciprocal operations compared to closeness centrality:

$$C_H(i) = \sum_{i,i \neq j} \frac{1}{d(i,j)}$$

where $\frac{1}{d(i,j)} = 0$ where there is no path between i and j. Normalization is by $N-1$. Since the harmonic mean performs better than the arithmetic mean on unconnected networks, i.e. networks with infinite distances, harmonic centrality is to be preferred in these cases.

Reach centrality

In some cases, longer path lengths are irrelevant and 'closeness' should be defined as how many others are in a local neighbourhood. How many steps out this neighbourhood should be defined as is given by the 'cutoff' parameter. This is usually termed k or m in equations, which is why this is sometimes called (m- or) k-step reach centrality:

$$C_R(i) = \sum_j d(i,j) \leq k$$

The maximum reach score is $N-1$, achieved when the node can reach all other nodes in the network in k steps or less, but the normalised version, $\frac{C_R}{N-1}$, is more common. Note that if $k = 1$ (i.e. cutoff = 1), then this returns the node's degree. At higher cutoff reach centrality returns the size of the node's component.

Information centrality

Information centrality, also known as current-flow centrality, is a hybrid measure relating to both path-length and walk-based measures. The information centrality of a node is the harmonic average of the "bandwidth" or inverse path-length for all paths originating from the node.

As described in the {sna} package, information centrality works on an undirected but potentially weighted network excluding isolates (which take scores of zero). It is defined as:

$$C_I = \frac{1}{T + \frac{\sum T - 2 \sum C_1}{|N|}}$$

where $C = B^{-}1$ with B is a pseudo-adjacency matrix replacing the diagonal of $1 - A$ with $1 + k$, and T is the trace of C and S_R an arbitrary row sum (all rows in C have the same sum).

Nodes with higher information centrality have a large number of short paths to many others in the network, and are thus considered to have greater control of the flow of information.

Eccentricity centrality

Eccentricity centrality, graph centrality, or the Koenig number, is the (if normalized, inverse of) the distance to the furthest node:

$$C_E(i) = \frac{1}{max_{j \in N} d(i,j)}$$

where the distance from i to j is ∞ if unconnected. As such it is only well defined for connected networks.

Closeness vitality centrality

The closeness vitality of a node is the change in the sum of all distances in a network, also known as the Wiener Index, when that node is removed. Note that the closeness vitality may be negative infinity if removing that node would disconnect the network. Formally:

$$C_V(i) = \sum_{j,k} d(j,k) - \sum_{j,k} d(j,k,G\ i)$$

where $d(j,k,G\ i)$ is the distance between nodes j and k in the network with node i removed.

Random walk closeness centrality

Random walk closeness centrality is based on the average length of random walks starting at all other nodes to reach a given node. It is defined as the inverse of the average hitting time to a node. This means that higher values are given to nodes that can be reached more quickly on average by random walks starting at other nodes. Formally:

$$C_{RW}(i) = \frac{1}{\frac{1}{N-1} \sum_{j \neq i} H_{ji}}$$

where H_{ji} is the hitting time from node j to node i.

References

On closeness centrality:

Bavelas, Alex. 1950. "Communication Patterns in Task-Oriented Groups". *The Journal of the Acoustical Society of America*, 22(6): 725–730. doi:10.1121/1.1906679

Harary, Frank. 1959. "Status and Contrastatus". *Sociometry*, 22(1): 23–43. doi:10.2307/2785610

On harmonic centrality:

Marchiori, Massimo, and Vito Latora. 2000. "Harmony in the small-world". *Physica A* 285: 539-546. doi:10.1016/S03784371(00)003113

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

On reach centrality:

Borgatti, Stephen P., Martin G. Everett, and J.C. Johnson. 2013. *Analyzing social networks*. London: SAGE Publications Limited.

On information centrality:

Stephenson, Karen, and Marvin Zelen. 1989. "Rethinking centrality: Methods and examples". *Social Networks* 11(1):1-37. doi:10.1016/03788733(89)900166

Brandes, Ulrik, and Daniel Fleischer. 2005. "Centrality Measures Based on Current Flow". *Proc. 22nd Symp. Theoretical Aspects of Computer Science* LNCS 3404: 533-544. doi:10.1007/9783-540318569_44

On eccentricity centrality:

Hage, Per, and Frank Harary. 1995. "Eccentricity and centrality in networks". *Social Networks*, 17(1): 57-63. doi:10.1016/03788733(94)002489

On closeness vitality centrality:

Koschuetzki, Dirk, Katharina Lehmann, Leon Peeters, Stefan Richter, Dagmar Tenfelde-Podehl, and Oliver Zlotowski. 2005. "Centrality Indices", in Brandes, Ulrik, and Thomas Erlebach (eds.). *Network Analysis: Methodological Foundations*. Springer: Berlin, pp. 16-61.

On random walk closeness centrality:

Noh, J.D. and R. Rieger. 2004. "Random Walks on Complex Networks". *Physical Review Letters*, 92(11): 118701. doi:10.1103/PhysRevLett.92.118701

See Also

Other closeness: `measure_centralisation_close`, `measure_centralities_close`

Other centrality: `measure_central_between`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`,

measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_closeness(ison_southern_women)
node_by_reach(ison_adolescents)
```

`measure_central_degree`

Measuring nodes degree-like centrality

Description

These functions calculate common degree-related centrality measures for one- and two-mode networks:

- `node_by_degree()` measures the degree centrality of nodes in an unweighted network, or weighted degree/strength of nodes in a weighted network; there are several related shortcut functions:
  - `node_by_deg()` returns the unnormalised results.
  - `node_by_indegree()` returns the `direction = 'in'` results.
  - `node_by_outdegree()` returns the `direction = 'out'` results.
- `node_by_multidegree()` measures the ratio between types of ties in a multiplex network.
- `node_by_posneg()` measures the PN (positive-negative) centrality of a signed network.
- `node_by_leverage()` measures the leverage centrality of nodes in a network.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

```
node_by_degree(
  .data,
  normalized = TRUE,
  alpha = 1,
```

```
  direction = c("all", "out", "in")
)

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

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

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

node_by_multidegree(.data, tie1, tie2)

node_by_posneg(.data)

node_by_leverage(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
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. By default "all".
tie1	Character string indicating the first uniplex network.
tie2	Character string indicating the second uniplex network.

Value

A node_measure numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

Degree centrality

The degree of a node is the number of connections it has. It is also sometimes called the valency of a node, $d(v)$. The maximum degree in a network is often denoted $\Delta(G)$ and the minimum degree in a network $\delta(G)$. The total degree of a network is the sum of all degrees, $\sum_v d(v)$. The

degree sequence is the set of all nodes' degrees, ordered from largest to smallest. Directed networks discriminate between outdegree (degree of outgoing ties) and indegree (degree of incoming ties).

Leverage centrality

Leverage centrality concerns the degree of a node compared with that of its neighbours, J:

$$C_L(i) = \frac{1}{d(i)} \sum_{j \in J(i)} \frac{d(i) - d(j)}{d(i) + d(j)}$$

References

On multimodal centrality:

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

On strength centrality:

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

On signed centrality:

Everett, Martin G., and Stephen P. Borgatti. 2014. "Networks Containing Negative Ties." *Social Networks* 38:111–20. doi:10.1016/j.socnet.2014.03.005

On leverage centrality:

Joyce, Karen E., Paul J. Laurienti, Jonathan H. Burdette, and Satoru Hayasaka. 2010. "A New Measure of Centrality for Brain Networks". *PLoS ONE* 5(8): e12200. doi:10.1371/journal.pone.0012200

See Also

Other degree: `mark_degree`, `measure_centralisation_degree`, `measure_centralities_degree`

Other centrality: `measure_central_between`, `measure_central_close`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_close`, `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`,

measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_degree(ison_southern_women)
```

---

measure_central_eigen *Measuring nodes eigenvector-like centrality*

Description

These functions calculate common eigenvector-related centrality measures, or walk-based eigen-measures, for one- and two-mode networks:

- `node_eigenvector()` measures the eigenvector centrality of nodes in a network.
- `node_power()` measures the Bonacich, beta, or power centrality of nodes in a network.
- `node_alpha()` measures the alpha or Katz centrality of nodes in a network.
- `node_pagerank()` measures the pagerank centrality of nodes in a network.
- `node_hub()` measures how well nodes in a network serve as hubs pointing to many authorities.
- `node_authority()` measures how well nodes in a network serve as authorities from many hubs.

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. `manynet::to_undirected()` functions. All centrality and centralization measures return normalized measures by default, including for two-mode networks.

Usage

```
node_by_eigenvector(.data, normalized = TRUE, scale = TRUE)

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

node_by_alpha(.data, alpha = 0.85)

node_by_pagerank(.data)

node_by_authority(.data)
```

```
node_by_hub(.data)

node_by_subgraph(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
scale	Logical scalar, whether to rescale the vector so the maximum score is 1.
exponent	Decay rate or attentuation factor for the Bonacich power centrality score. Can be positive or negative.
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 node_measure numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

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 $Ax = \lambda x$. Note that since {igraph} v2.1.1, the values will always be rescaled so that the maximum is 1.

Power or beta (or Bonacich) centrality

Power centrality includes an exponent that weights contributions to a node's centrality based on how far away those other nodes are.

$$c_b(i) = \sum A(i,j)(\alpha = \beta c(j))$$

Where β is positive, this means being connected to central people increases centrality. Where β is negative, this means being connected to central people decreases centrality (and being connected to more peripheral actors increases centrality). When $\beta = 0$, this is the outdegree. α is calculated to make sure the root mean square equals the network size.

Alpha centrality

Alpha or Katz (or Katz-Bonacich) centrality operates better than eigenvector centrality for directed networks because 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 $x = (I - \frac{1}{\lambda} A^T)^{-1} e$.

Subgraph centrality

Subgraph centrality measures the participation of a node in all subgraphs in the network, giving higher weight to smaller subgraphs. It is defined as:

$$C_S(i) = \sum_{k=0}^{\infty} \frac{(A^k)_{ii}}{k!}$$

where $(A^k)_{ii}$ is the ith diagonal element of the kth power of the adjacency matrix A, representing the number of closed walks of length k starting and ending at node i. Weighting by $\frac{1}{k!}$ ensures that shorter walks contribute more to the centrality score than longer walks.

Subgraph centrality is a good choice of measure when the focus is on local connectivity and clustering around a node, as it captures the extent to which a node is embedded in tightly-knit groups within the network. Note though that because of the way spectral decomposition is used to calculate this measure, this is not a good measure for very large graphs.

References

On eigenvector centrality:

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

On power centrality:

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

On alpha centrality:

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.

On pagerank centrality:

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.

On hub and authority centrality:

Kleinberg, Jon. 1999. "Authoritative sources in a hyperlinked environment". *Journal of the ACM* 46(5): 604–632. doi:10.1145/324133.324140

On subgraph centrality:

Estrada, Ernesto and Rodríguez-Velázquez, Juan A. 2005. "Subgraph centrality in complex networks". *Physical Review E* 71(5): 056103. doi:10.1103/PhysRevE.71.056103

See Also

Other eigenvector: measure_centralisation_eigen, measure_centralities_eigen

Other centrality: measure_central_between, measure_central_close, measure_central_degree, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_centralisation_between, measure_centralisation_close, measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_eigenvector(ison_southern_women)
node_by_power(ison_southern_women, exponent = 0.5)
```

`measure_closure` *Measuring network closure*

Description

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

- `net_by_reciprocity()` measures reciprocity in a (usually directed) network.
- `net_by_transitivity()` measures transitivity in a network.
- `net_by_equivalency()` measures equivalence or reinforcement in a (usually two-mode) network.
- `net_by_congruency()` measures congruency across two two-mode networks.

Usage

```
net_by_reciprocity(.data, method = "default")

net_by_transitivity(.data)

net_by_equivalency(.data)

net_by_congruency(.data, object2)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`method`	For reciprocity, either `default` or `ratio`. See `?igraph::reciprocity`
`object2`	Optionally, a second (two-mode) matrix, igraph, or tidygraph

Details

For one-mode networks, shallow wrappers of igraph versions exist via `net_reciprocity` and `net_transitivity`.

For two-mode networks, `net_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, `net_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.

Value

A `network_measure` numeric score.

Equivalency

The net_by_equivalency() function calculates the Robins and Alexander (2004) clustering coefficient for two-mode networks. Note that for weighted two-mode networks, the result is divided by the average tie weight.

References

On equivalency or four-cycles:

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.

On congruency:

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: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
net_by_reciprocity(ison_southern_women)
net_by_transitivity(ison_adolescents)
net_by_equivalency(ison_southern_women)
```

---

measure_closure_node *Measuring node closure*

Description

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

- node_by_reciprocity() measures nodes' reciprocity.
- node_by_transitivity() measures nodes' transitivity.
- node_by_equivalency() measures nodes' equivalence or reinforcement in a (usually two-mode) network.

Usage

```
node_by_reciprocity(.data)

node_by_transitivity(.data)

node_by_equivalency(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Details

For one-mode networks, shallow wrappers of igraph versions exist via node_by_reciprocity and node_by_transitivity.

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

Value

A node_measure numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

See Also

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_reciprocity(to_unweighted(ison_networkers))
node_by_transitivity(ison_adolescents)
```

`measure_cohesion` *Measures of network cohesion*

Description

These functions return values or vectors relating to how cohesive a network is:

- `net_by_density()` measures the ratio of ties to the number of possible ties.
- `net_by_components()` measures the number of (strong) components in the network.
- `net_by_independence()` measures the independence number, or size of the largest independent set in the network.

Usage

```
net_by_density(.data)

net_by_components(.data)

net_by_independence(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `network_measure` numeric score.

Components

To get the 'weak' components of a directed graph, please use `manynet::to_undirected()` first.

See Also

Other cohesion: `mark_triangles`, `measure_breadth`, `measure_fragmentation`, `motif_net`, `motif_node`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c` `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_net`, `measure_diverse_node`, `measure_features`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Examples

```
net_by_density(ison_adolescents)
net_by_density(ison_southern_women)
  net_by_components(fict_thrones)
  net_by_components(to_undirected(fict_thrones))
net_by_independence(ison_adolescents)
```

`measure_core` *Measuring nodes' coreness*

Description

These functions identify nodes belonging to (some level of) the core of a network:

- `node_by_coreness()` returns a continuous measure of how closely each node resembles a typical core node.
- `node_by_kcoreness()` assigns nodes to their level of k-coreness.

Usage

```
node_by_kcoreness(.data)

node_by_coreness(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `node_measure` numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

k-coreness

k-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.

References

On k-coreness:

Seidman, Stephen B. 1983. "Network structure and minimum degree". *Social Networks*, 5(3), 269-287. doi:10.1016/03788733(83)90028X

Batagelj, Vladimir, and Matjaz Zaversnik. 2003. "An O(m) algorithm for cores decomposition of networks". *arXiv preprint* cs/0310049. doi:10.48550/arXiv.cs/0310049

See Also

Other core-periphery: mark_core, member_core

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_kcoreness(ison_adolescents)
node_by_coreness(ison_adolescents)
```

---

measure_diffusion_infection

Measures of network infection

Description

These functions allow measurement of various features of a diffusion process at the network level:

- net_by_infection_complete() measures the number of time steps until (the first instance of) complete infection. For diffusions that are not observed to complete, this function returns the value of Inf (infinity). This makes sure that at least ordinality is respected.
- net_by_infection_total() measures the proportion or total number of nodes that are infected/activated at some time by the end of the diffusion process. This includes nodes that subsequently recover. Where reinfection is possible, the proportion may be higher than 1.
- net_by_infection_peak() measures the number of time steps until the highest infection rate is observed.

Usage

```
net_by_infection_complete(.data)

net_by_infection_total(.data, normalized = TRUE)

net_by_infection_peak(.data)
```

Arguments

.data	Network data with nodal changes, as created by play_diffusion(), or a valid network diffusion model, as created by as_diffusion().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A network_measure numeric score.

See Also

Other diffusion: mark_diff, measure_diffusion_net, measure_diffusion_node, member_diffusion, motif_exposure, motif_hazard

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Examples

```
smeg <- generate_smallworld(15, 0.025)
smeg_diff <- play_diffusion(smeg)
net_by_infection_complete(smeg_diff)
net_by_infection_total(smeg_diff)
net_by_infection_peak(smeg_diff)
```

---

measure_diffusion_net *Measures of network diffusion*

Description

These functions allow measurement of various features of a diffusion process at the network level:

- net_by_transmissibility() measures the average transmissibility observed in a diffusion simulation, or the number of new infections over the number of susceptible nodes.
- net_by_recovery() measures the average number of time steps nodes remain infected once they become infected.
- net_by_reproduction() measures the observed reproductive number in a diffusion simulation as the network's transmissibility over the network's average infection length.
- net_by_immunity() measures the proportion of nodes that would need to be protected through vaccination, isolation, or recovery for herd immunity to be reached.

Usage

```
net_by_transmissibility(.data)

net_by_recovery(.data, censor = TRUE)

net_by_reproduction(.data)

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

Arguments

.data	Network data with nodal changes, as created by play_diffusion(), or a valid network diffusion model, as created by as_diffusion().
censor	Where some nodes have not yet recovered by the end of the simulation, right censored values can be replaced by the number of steps. By default TRUE. Note that this will likely still underestimate recovery.
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.

Value

A network_measure numeric score.

Transmissibility

net_transmissibility() measures how many directly susceptible nodes each infected node will infect in each time period, on average. That is:

$$T = \frac{1}{n} \sum_{j=1}^{n} \frac{i_j}{s_j}$$

where i is the number of new infections in each time period, $j \in n$, and s is the number of nodes that could have been infected in that time period (note that $s \neq S$, or the number of nodes that are susceptible in the population). T can be interpreted as the proportion of susceptible nodes that are infected at each time period.

Recovery time

net_recovery() measures the average number of time steps that nodes in a network remain infected. Note that in a diffusion model without recovery, average infection length will be infinite. This will also be the case where there is right censoring. The longer nodes remain infected, the longer they can infect others.

Reproduction number

net_reproduction() measures a given diffusion's reproductive number. Here it is calculated as:

$$R = \min\left(\frac{T}{1/L}, \bar{k}\right)$$

where T is the observed transmissibility in a diffusion and L is the observed recovery length in a diffusion. Since L can be infinite where there is no recovery or there is right censoring, and since network structure places an upper limit on how many nodes each node may further infect (their degree), this function returns the minimum of R_0 and the network's average degree.

Interpretation of the reproduction number is oriented around R = 1. Where $R > 1$, the 'disease' will 'infect' more and more nodes in the network. Where $R < 1$, the 'disease' will not sustain itself and eventually die out. Where $R = 1$, the 'disease' will continue as endemic, if conditions allow.

Herd immunity

`net_immunity()` estimates the proportion of a network that need to be protected from infection for herd immunity to be achieved. This is known as the Herd Immunity Threshold or HIT:

$$1 - \frac{1}{R}$$

where R is the reproduction number from `net_reproduction()`. The HIT indicates the threshold at which the reduction of susceptible members of the network means that infections will no longer keep increasing. Note that there may still be more infections after this threshold has been reached, but there should be fewer and fewer. These excess infections are called the *overshoot*. This function does *not* take into account the structure of the network, instead using the average degree.

Interpretation is quite straightforward. A HIT or immunity score of 0.75 would mean that 75% of the nodes in the network would need to be vaccinated or otherwise protected to achieve herd immunity. To identify how many nodes this would be, multiply this proportion with the number of nodes in the network.

References

On epidemiological models:

Kermack, William O., and Anderson Gray McKendrick. 1927. "A contribution to the mathematical theory of epidemics". *Proc. R. Soc. London A* 115: 700-721. doi:10.1098/rspa.1927.0118

On the basic reproduction number:

Diekmann, Odo, Hans J.A.P. Heesterbeek, and Hans J.A.J. Metz. 1990. "On the definition and the computation of the basic reproduction ratio R0 in models for infectious diseases in heterogeneous populations". *Journal of Mathematical Biology*, 28(4): 365–82. doi:10.1007/BF00178324

Kenah, Eben, and James M. Robins. 2007. "Second look at the spread of epidemics on networks". *Physical Review E*, 76(3 Pt 2): 036113. doi:10.1103/PhysRevE.76.036113

On herd immunity:

Garnett, G.P. 2005. "Role of herd immunity in determining the effect of vaccines against sexually transmitted disease". *The Journal of Infectious Diseases*, 191(1): S97-106. doi:10.1086/425271

See Also

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c`
`measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`,

measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other diffusion: mark_diff, measure_diffusion_infection, measure_diffusion_node, member_diffusion, motif_exposure, motif_hazard

Examples

```
smeg <- generate_smallworld(15, 0.025)
smeg_diff <- play_diffusion(smeg, recovery = 0.2)
plot(smeg_diff)
# To calculate the average transmissibility for a given diffusion model
net_by_transmissibility(smeg_diff)
# To calculate the average infection length for a given diffusion model
net_by_recovery(smeg_diff)
# To calculate the reproduction number for a given diffusion model
net_by_reproduction(smeg_diff)
# Calculating the proportion required to achieve herd immunity
net_by_immunity(smeg_diff)
# To find the number of nodes to be vaccinated
net_by_immunity(smeg_diff, normalized = FALSE)
```

---

measure_diffusion_node

Measures of nodes in a diffusion

Description

These functions allow measurement of various features of a diffusion process:

- `node_by_adopt_time()`: Measures the number of time steps until nodes adopt/become infected
- `node_by_adopt_threshold()`: Measures nodes' thresholds from the amount of exposure they had when they became infected
- `node_by_adopt_recovery()`: Measures the average length nodes that become infected remain infected in a compartmental model with recovery
- `node_by_adopt_exposure()`: Measures how many exposures nodes have to a given mark

Usage

```
node_by_adopt_time(.data)

node_by_adopt_threshold(.data, normalized = TRUE, lag = 1)

node_by_adopt_recovery(.data)

node_by_adopt_exposure(.data, mark, time = 0)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
normalized	Logical scalar, whether scores are normalized. Different denominators may be used depending on the measure, whether the object is one-mode or two-mode, and other arguments. By default TRUE.
lag	The number of time steps back upon which the thresholds are inferred.
mark	A valid 'node_mark' object or logical vector (TRUE/FALSE) of length equal to the number of nodes in the network.
time	A time point until which infections/adoptions should be identified. By default time = 0.

Value

A node_measure numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

Adoption time

node_by_adopt_time() measures the time units it took until each node became infected. Note that an adoption time of 0 indicates that this was a seed node.

Thresholds

node_by_adopt_threshold() infers nodes' thresholds based on how much exposure they had when they were infected. This inference is of course imperfect, especially where there is a sudden increase in exposure, but it can be used heuristically. In a threshold model, nodes activate when $\sum_{j:\text{active}} w_{ji} \geq \theta_i$, where w is some (potentially weighted) matrix, j are some already activated nodes, and $theta$ is some pre-defined threshold value. Where a fractional threshold is used, the equation is $\frac{\sum_{j:\text{active}} w_{ji}}{\sum_j w_{ji}} \geq \theta_i$. That is, $theta$ is now a proportion, and works regardless of whether w is weighted or not.

Recovery

node_by_adopt_recovery() measures the average length of time that nodes that become infected remain infected in a compartmental model with recovery. Infections that are not concluded by the end of the study period are calculated as infinite.

Exposure

node_exposure() calculates the number of infected/adopting nodes to which each susceptible node is exposed. It usually expects network data and an index or mark (TRUE/FALSE) vector of those nodes which are currently infected, but if a diff_model is supplied instead it will return nodes exposure at $t = 0$.

References

On diffusion measures:
Valente, Tom W. 1995. *Network models of the diffusion of innovations* (2nd ed.). Cresskill N.J.: Hampton Press.

See Also

Other diffusion: mark_diff, measure_diffusion_infection, measure_diffusion_net, member_diffusion, motif_exposure, motif_hazard

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
smeg <- generate_smallworld(15, 0.025)
smeg_diff <- play_diffusion(smeg, recovery = 0.2)
plot(smeg_diff)
# To measure when nodes adopted a diffusion/were infected
(times <- node_by_adopt_time(smeg_diff))
# To infer nodes' thresholds
node_by_adopt_threshold(smeg_diff)
# To measure how long each node remains infected for
node_by_adopt_recovery(smeg_diff)
# To measure how much exposure nodes have to a given mark
node_by_adopt_exposure(smeg, mark = c(1,3))
node_by_adopt_exposure(smeg_diff)
```

---

measure_diverse_net *Measures of network diversity*

Description

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

- `net_by_richness()` measures the number of unique categories in a network attribute.
- `net_by_diversity()` measures the heterogeneity of ties across a network.

Usage

```
net_by_richness(.data, attribute)

net_by_diversity(
  .data,
  attribute,
  diversity = c("blau", "teachman", "variation", "gini")
)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`attribute`	Name of a nodal attribute, mark, measure, or membership vector.
`diversity`	Which method to use for `*_diversity()`. Either "blau" (Blau's index) or "teachman" (Teachman's index) for categorical attributes, or "variation" (coefficient of variation) or "gini" (Gini coefficient) for numeric attributes. Default is "blau". If an incompatible method is chosen for the attribute type, a suitable alternative will be used instead with a message.

Value

A `network_measure` numeric score.

Richness

Richness is a simple count of the number of different categories present for a given attribute.

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.

Teachman's index (1980) is based on information theory and is calculated as:

$$-\sum_{i=1}^{k} p_i \log(p_i)$$

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 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. It thus shares similar properties to Blau's index, but includes also a notion of richness that tends to give more weight to rare categories and thus tends to highlight imbalances more.

The coefficient of variation (CV) is a standardised measure of dispersion of a probability distribution or frequency distribution. It is defined as the ratio of the standard deviation σ to the mean μ:

$$CV = \frac{\sigma}{\mu}$$

It is often expressed as a percentage. The CV is useful because the standard deviation of data must always be understood in the context of the mean of the data. The CV is particularly useful when comparing the degree of variation from one data series to another, even if the means are drastically different from each other.

The Gini coefficient is a measure of statistical dispersion that is intended to represent the income or wealth distribution of a nation's residents, and is commonly used as a measure of inequality. It is defined as a ratio with values between 0 and 1, where 0 corresponds with perfect equality (everyone has the same income) and 1 corresponds with perfect inequality (one person has all the income, and everyone else has zero income). The Gini coefficient can be calculated from the Lorenz curve, which plots the proportion of the total income of the population that is cumulatively earned by the bottom x% of the population. The Gini coefficient is defined as the area between the line of equality and the Lorenz curve, divided by the total area under the line of equality.

References

On richness:

Magurran, Anne E. 1988. *Ecological Diversity and Its Measurement*. Princeton: Princeton University Press. doi:10.1007/9789401573580

On diversity:

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

Teachman, Jay D. 1980. Analysis of population diversity: Measures of qualitative variation. *Sociological Methods & Research*, 8:341-362. doi:10.1177/004912418000800305

Page, Scott E. 2010. *Diversity and Complexity*. Princeton: Princeton University Press. doi:10.1515/9781400835140

See Also

Other diversity: `measure_assort_net`, `measure_assort_node`, `measure_diverse_node`

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c` `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_node`, `measure_features`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Examples

```
net_by_richness(ison_networkers)
marvel_friends <- to_unsigned(to_uniplex(fict_marvel, "relationship"), "positive")
net_by_diversity(marvel_friends, "Gender")
net_by_diversity(marvel_friends, "Appearances")
```

`measure_diverse_node` *Measures of nodes diversity*

Description

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

- `node_by_richness()` measures the number of unique categories of an attribute to which each node is connected.
- `node_by_diversity()` measures the heterogeneity of each node's local neighbourhood.

Usage

```
node_by_richness(.data, attribute)

node_by_diversity(
  .data,
  attribute,
  diversity = c("blau", "teachman", "variation", "gini")
)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`attribute`	Name of a nodal attribute, mark, measure, or membership vector.
`diversity`	Which method to use for `*_diversity()`. Either "blau" (Blau's index) or "teachman" (Teachman's index) for categorical attributes, or "variation" (coefficient of variation) or "gini" (Gini coefficient) for numeric attributes. Default is "blau". If an incompatible method is chosen for the attribute type, a suitable alternative will be used instead with a message.

Value

A `node_measure` numeric vector the length of the nodes in the network, providing the scores for each node. If the network is labelled, then the scores will be labelled with the nodes' names.

See Also

Other diversity: measure_assort_net, measure_assort_node, measure_diverse_net

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_features, measure_fragmentation, measure_hierarchy, measure_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_by_richness(ison_networkers, "Discipline")
marvel_friends <- to_unsigned(to_uniplex(fict_marvel, "relationship"), "positive")
node_by_diversity(marvel_friends, "Gender")
node_by_diversity(marvel_friends, "Attractive")
```

---

measure_features *Measuring network topological features*

Description

These functions measure certain topological features of networks:

- net_by_core() measures the correlation between a network and a core-periphery model with the same dimensions.
- net_by_richclub() measures the rich-club coefficient of a network.
- net_by_factions() measures the correlation between a network and a component model with the same dimensions. If no 'membership' vector is given for the data, node_partition() is used to partition nodes into two groups.
- net_by_modularity() measures the modularity of a network based on nodes' membership in defined clusters.
- net_by_smallworld() measures the small-world coefficient for one- or two-mode networks. Small-world networks can be highly clustered and yet have short path lengths.
- net_by_scalefree() measures the exponent of a fitted power-law distribution. An exponent between 2 and 3 usually indicates a power-law distribution.
- net_by_balance() measures 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.

Usage

```
net_by_core(
  .data,
  mark = NULL,
  method = c("correlation", "ident", "ndiff", "diff")
)

net_by_richclub(.data)

net_by_factions(.data, membership = NULL)

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

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

net_by_scalefree(.data)

net_by_balance(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
mark	A logical vector indicating which nodes belong to the core.
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 $SWI = 1$.

`membership` A character string naming an existing node attribute in the network, or a categorical vector of the same length as the number of nodes in the network where each element indicates the group membership of the corresponding node. While this may often be a vector created using `node_in_*()` functions, it can be any character vector that assigns nodes to groups or categories.

`resolution` A proportion indicating the resolution scale. By default 1, which returns the original definition of modularity. The higher this parameter, the more smaller communities will be privileged. The lower this parameter, the fewer larger communities are likely to be found.

`times` Integer of number of simulations.

Value

A `network_measure` numeric score.

Core-Periphery

`net_core()` calculates the Pearson correlation between the given network, where the nodes in the core are assigned by some given mark, and an ideal typical core-periphery network with the same number of nodes in the core and the periphery.

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. To help manage this problem, a `resolution` parameter is added. Please see the argument definition for more details.

Source

`{signnet}` by David Schoch

References

On core-periphery:

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

On the rich-club coefficient:

Zhou, Shi, and Raul J. Mondragon. 2004. "The Rich-Club Phenomenon in the Internet Topology". *IEEE Communications Letters*, 8(3): 180-182. doi:10.1109/lcomm.2004.823426

On modularity:

Newman, Mark E.J. 2006. "Modularity and community structure in networks", *Proceedings of the National Academy of Sciences* 103(23): 8577-8696. doi:10.1073/pnas.0601602103

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

On small-worldliness:

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

On scale-free networks:

Barabasi, Albert-Laszlo, and Reka Albert. 1999. "Emergence of scaling in random networks", *Science*, 286(5439): 509-512. doi:10.1126/science.286.5439.509

Clauset, Aaron, Cosma Rohilla Shalizi, and Mark E.J. Newman. 2009. "Power-law distributions in empirical data", *SIAM Review*, 51(4): 661-703. doi:10.1137/070710111

Stumpf, Michael P.H., and Mason Porter. 2012. "Critical truths about power laws", *Science*, 335(6069): 665-666. doi:10.1126/science.1216142

Holme, Petter. 2019. "Rare and everywhere: Perspectives on scale-free networks", *Nature Communications*, 10(1): 1016. doi:10.1038/s41467019090388

On balance theory:

Heider, Fritz. 1946. "Attitudes and cognitive organization". *The Journal of Psychology*, 21: 107-112. doi:10.1080/00223980.1946.9917275

Cartwright, D., and Frank Harary. 1956. "Structural balance: A generalization of Heider's theory". *Psychological Review*, 63(5): 277-293. doi:10.1037/h0046049

See Also

`net_by_transitivity()` and `net_by_equivalency()` for how clustering is calculated

Other measures: `measure_assort_net`, `measure_assort_node`, `measure_breadth`, `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_centralisation_between`, `measure_centralisation_c` `measure_centralisation_degree`, `measure_centralisation_eigen`, `measure_centralities_between`, `measure_centralities_close`, `measure_centralities_degree`, `measure_centralities_eigen`, `measure_closure`, `measure_closure_node`, `measure_cohesion`, `measure_core`, `measure_diffusion_infection`, `measure_diffusion_net`, `measure_diffusion_node`, `measure_diverse_net`, `measure_diverse_node`, `measure_fragmentation`, `measure_hierarchy`, `measure_periods`

Examples

```
net_by_core(ison_adolescents)
net_by_core(ison_southern_women)
net_by_richclub(ison_adolescents)
```

```
  net_by_factions(ison_southern_women)
net_by_modularity(ison_adolescents,
  node_in_partition(ison_adolescents))
net_by_modularity(ison_southern_women,
  node_in_partition(ison_southern_women))
net_by_smallworld(ison_brandes)
net_by_smallworld(ison_southern_women)
net_by_scalefree(ison_adolescents)
net_by_scalefree(generate_scalefree(50, 1.5))
net_by_scalefree(create_lattice(100))
net_by_balance(to_uniplex(fict_marvel, "relationship"))
```

`measure_fragmentation` *Measures of network fragmentation*

Description

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

- `net_by_cohesion()` measures the minimum number of nodes to remove from the network needed to increase the number of components.
- `net_by_toughness()` measures the number of nodes that would need to be removed from a network to increase its number of components.
- `net_by_adhesion()` measures the minimum number of ties to remove from the network needed to increase the number of components.
- `net_by_strength()` measures the number of ties that would need to be removed from a network to increase its number of components.

Usage

```
net_by_cohesion(.data)

net_by_adhesion(.data)

net_by_strength(.data)

net_by_toughness(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `network_measure` numeric score.

References

On cohesion:

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. doi:10.1111/00811750.00098

See Also

Other cohesion: mark_triangles, measure_breadth, measure_cohesion, motif_net, motif_node

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_hierarchy, measure_periods

Examples

```
net_by_cohesion(fict_marvel)
net_by_cohesion(to_giant(fict_marvel))
net_by_adhesion(fict_marvel)
net_by_adhesion(to_giant(fict_marvel))
net_by_strength(ison_adolescents)
net_by_toughness(ison_adolescents)
```

---

measure_hierarchy *Measures of hierarchy*

Description

These functions, together with `net_reciprocity()`, are used jointly to measure how hierarchical a network is:

- `net_by_connectedness()` measures the proportion of dyads in the network that are reachable to one another, or the degree to which network is a single component.
- `net_by_efficiency()` measures the Krackhardt efficiency score.
- `net_by_upperbound()` measures the Krackhardt (least) upper bound score.

Usage

```
net_by_connectedness(.data)

net_by_efficiency(.data)

net_by_upperbound(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A network_measure numeric score.

References

On hierarchy:

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: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_periods

Other hierarchy: motif_hierarchy

Examples

```
net_by_connectedness(ison_networkers)
1 - net_by_reciprocity(ison_networkers)
net_by_efficiency(ison_networkers)
net_by_upperbound(ison_networkers)
```

---

measure_periods *Measures of network change*

Description

net_by_waves() measures the number of waves in longitudinal network data.

Usage

```
net_by_waves(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A network_measure numeric score.

See Also

Other change: motif_periods

Other measures: measure_assort_net, measure_assort_node, measure_breadth, measure_broker_node, measure_broker_tie, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_centralisation_between, measure_centralisation_c measure_centralisation_degree, measure_centralisation_eigen, measure_centralities_between, measure_centralities_close, measure_centralities_degree, measure_centralities_eigen, measure_closure, measure_closure_node, measure_cohesion, measure_core, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, measure_diverse_net, measure_diverse_node, measure_features, measure_fragmentation, measure_hierarchy

---

member_brokerage *Memberships in brokerage positions*

Description

node_in_brokerage() returns nodes membership as a powerhouse, connector, linchpin, or sideliner according to Hamilton et al. (2020).

Usage

```
node_in_brokering(.data, membership)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
membership	A character string naming an existing node attribute in the network, or a categorical vector of the same length as the number of nodes in the network where each element indicates the group membership of the corresponding node. While this may often be a vector created using node_in_*() functions, it can be any character vector that assigns nodes to groups or categories.

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

References

On brokerage activity and exclusivity:
Hamilton, Matthew, Jacob Hileman, and Orjan Bodin. 2020. "Evaluating heterogeneous brokerage: New conceptual and methodological approaches and their application to multi-level environmental governance networks" *Social Networks* 61: 1-10. doi:10.1016/j.socnet.2019.08.002

See Also

Other brokerage: measure_broker_node, measure_broker_tie, measure_brokerage, motif_brokerage_net, motif_brokerage_node

Other memberships: member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

---

member_cliques *Memberships in maximally diverse cliques*

Description

These functions create a vector of nodes' memberships in cliques:

- `node_in_roulette()` assigns nodes to maximally diverse groups.

Usage

```
node_in_roulette(.data, num_groups, group_size, times = NULL)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`num_groups`	An integer indicating the number of groups desired.
`group_size`	An integer indicating the desired size of most of the groups. Note that if the number of nodes is not divisible into groups of equal size, there may be some larger or smaller groups.
`times`	An integer of the number of search iterations the algorithm should complete. By default this is the number of nodes in the network multiplied by the number of groups. This heuristic may be insufficient for small networks and numbers of groups, and burdensome for large networks and numbers of groups, but can be overwritten. At every 10th iteration, a stronger perturbation of a number of successive changes, approximately the number of nodes divided by the number of groups, will take place irrespective of whether it improves the objective function.

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

Maximally diverse grouping problem

This well known computational problem is a NP-hard problem with a number of relevant applications, including the formation of groups of students that have encountered each other least or least recently. Essentially, the aim is to return a membership of nodes in cliques that minimises the sum of their previous (weighted) ties:

$$\sum_{g=1}^{m}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} x_{ij}y_{ig}y_{jg}$$

where $y_{ig} = 1$ if node i is in group g, and 0 otherwise.

x_{ij} is the existing network data. If this is an empty network, the function will just return cliques. To run this repeatedly, one can join a clique network of the membership result with the original network, using this as the network data for the next round.

A form of the Lai and Hao (2016) iterated maxima search (IMS) is used here. This performs well for small and moderately sized networks. It includes both weak and strong perturbations to an initial solution to ensure that a robust solution from the broader state space is identified. The user is referred to Lai and Hao (2016) and Lai et al (2021) for more details.

References

On the maximally diverse grouping problem:

Lai, Xiangjing, and Jin-Kao Hao. 2016. "Iterated Maxima Search for the Maximally Diverse Grouping Problem." *European Journal of Operational Research* 254(3):780–800. doi:10.1016/j.ejor.2016.05.018.

Lai, Xiangjing, Jin-Kao Hao, Zhang-Hua Fu, and Dong Yue. 2021. "Neighborhood Decomposition Based Variable Neighborhood Search and Tabu Search for Maximally Diverse Grouping." *European Journal of Operational Research* 289(3):1067–86. doi:10.1016/j.ejor.2020.07.048.

See Also

Other memberships: member_brokerage, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

member_community *Memberships in communities*

Description

node_in_community() runs through all available community detection algorithms for a given network type, finds the algorithm that returns the largest modularity score, and returns the corresponding membership partition. Where feasible (a small enough network), the optimal problem solving technique is used to ensure the maximal modularity partition. For larger networks, it identifies the applicable algorithms and finds the algorithm that maximises modularity and returns that membership vector.

Usage

```
node_in_community(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

See Also

Other community: member_community_hier, member_community_non

Other memberships: member_brokerage, member_cliques, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

`member_community_hier` *Memberships in hierarchical communities*

Description

These functions offer algorithms for hierarchically clustering networks into communities. Since all of the following are hierarchical, their dendrograms can be plotted:

- `node_in_betweenness()` is a hierarchical, decomposition algorithm where edges are removed in decreasing order of the number of shortest paths passing through the edge.
- `node_in_greedy()` is a hierarchical, agglomerative algorithm, that tries to optimize modularity in a greedy manner.
- `node_in_eigen()` is a top-down, hierarchical algorithm.
- `node_in_walktrap()` is a hierarchical, agglomerative algorithm based on random walks.

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_in_betweenness(.data)

node_in_greedy(.data)

node_in_eigen(.data)

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`times`	Integer indicating number of simulations/walks used. By default, `times=50`.

Value

A `node_member` character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

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.

References

On edge-betweenness community detection:

Newman, Mark, and Michelle Girvan. 2004. "Finding and evaluating community structure in networks." *Physical Review E* 69: 026113. doi:10.1103/PhysRevE.69.026113

On fast-greedy community detection:

Clauset, Aaron, Mark E.J. Newman, and Cristopher Moore. 2004. "Finding community structure in very large networks." *Physical Review E*, 70: 066111. doi:10.1103/PhysRevE.70.066111

On leading eigenvector community detection:

Newman, Mark E.J. 2006. "Finding community structure using the eigenvectors of matrices" *Physical Review E* 74:036104. doi:10.1103/PhysRevE.74.036104

On walktrap community detection:

Pons, Pascal, and Matthieu Latapy. 2005. "Computing communities in large networks using random walks". 1-20. doi:10.48550/arXiv.physics/0512106

See Also

Other memberships: `member_brokerage`, `member_cliques`, `member_community`, `member_community_non`, `member_components`, `member_core`, `member_diffusion`, `member_equivalence`

Other nodal: `mark_core`, `mark_degree`, `mark_diff`, `mark_nodes`, `mark_select_node`, `measure_assort_node`, `measure_broker_node`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_closure_node`, `measure_core`, `measure_diffusion_node`, `measure_diverse_node`, `member_brokerage`, `member_cliques`, `member_community`, `member_community_non`, `member_components`, `member_core`, `member_diffusion`, `member_equivalence`, `motif_brokerage_node`, `motif_exposure`, `motif_node`, `motif_path`

Other community: member_community, member_community_non

Examples

```
node_in_betweenness(ison_adolescents)
node_in_greedy(ison_adolescents)
node_in_eigen(ison_adolescents)
node_in_walktrap(ison_adolescents)
```

---

member_community_non *Memberships in non-hierarchical communities*

Description

These functions offer algorithms for partitioning networks into sets of communities:

- node_in_community() runs either optimal or, for larger networks, finds the algorithm that maximises modularity and returns that membership vector.
- node_in_optimal() is a problem-solving algorithm that seeks to maximise modularity over all possible partitions.
- node_in_partition() is a greedy, iterative, deterministic partitioning algorithm that results in two equally-sized communities.
- node_in_infomap() is an algorithm based on the information in random walks.
- node_in_spinglass() is a greedy, iterative, probabilistic algorithm, based on analogy to model from statistical physics.
- node_in_fluid() is a propogation-based partitioning algorithm, based on analogy to model from fluid dynamics.
- node_in_louvain() is an agglomerative multilevel algorithm that seeks to maximise modularity over all possible partitions.
- node_in_leiden() is an agglomerative multilevel algorithm that seeks to maximise the Constant Potts Model over all possible partitions.

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_in_optimal(.data)

node_in_partition(.data)

node_in_infomap(.data, times = 50)

node_in_spinglass(.data, max_k = 200, resolution = 1)

node_in_fluid(.data)
```

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

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

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
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.

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

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.

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.

Fluid

The general idea is to observe how a discrete number of fluids interact, expand and contract, in a non-homogenous environment, i.e. the network structure. Unlike the `{igraph}` implementation that this function wraps, this function iterates over all possible numbers of communities and returns the membership associated with the highest modularity.

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. Compared to the Louvain method, the Leiden algorithm additionally tries to avoid unconnected communities.

References

On optimal community detection:

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.

On partitioning community detection:

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

On infomap community detection:

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

On spinglass community detection:

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

Traag, Vincent A., and Jeroen Bruggeman. 2009. "Community detection in networks with positive and negative links". *Physical Review E*, 80(3): 036115. doi:10.1103/PhysRevE.80.036115

On fluid community detection:

Parés Ferran, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguade, Jesus Labarta, Ulises Cortes, and Toyotaro Suzumura. 2018. "Fluid Communities: A Competitive, Scalable and Diverse Community Detection Algorithm". In: *Complex Networks & Their Applications VI* Springer, 689: 229. doi:10.1007/9783319721507_19

On Louvain community detection:

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

On Leiden community detection:

Traag, Vincent A., Ludo Waltman, and Nees Jan van Eck. 2019. "From Louvain to Leiden: guaranteeing well-connected communities", *Scientific Reports*, 9(1):5233. doi:10.1038/s41598-01941695z

See Also

Other community: member_community, member_community_hier

Other memberships: member_brokerage, member_cliques, member_community, member_community_hier, member_components, member_core, member_diffusion, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_in_optimal(ison_adolescents)
node_in_partition(ison_adolescents)
node_in_partition(ison_southern_women)
node_in_infomap(ison_adolescents)
node_in_spinglass(ison_adolescents)
node_in_fluid(ison_adolescents)
node_in_louvain(ison_adolescents)
node_in_leiden(ison_adolescents)
```

member_components *Memberships in components*

Description

These functions create a vector of nodes' memberships in components:

- node_in_component() assigns nodes' component membership using edge direction where available.
- node_in_weak() assigns nodes' component membership ignoring edge direction.
- node_in_strong() assigns nodes' component membership based on edge direction.

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.

Usage

```
node_in_component(.data)

node_in_weak(.data)

node_in_strong(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

See Also

Other memberships: member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_core, member_diffusion, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
ison_monks %>% to_uniplex("esteem") %>%
  mutate_nodes(comp = node_in_component())
```

`member_core` *Memberships in core-periphery categories*

Description

`node_in_core()` categorizes nodes into two or more core/periphery categories based on their coreness.

Usage

```
node_in_core(.data, groups = 3, cluster_by = c("bins", "quantiles", "kmeans"))
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`groups`	Number of categories to create. Must be at least 2 and at most the number of nodes in the network. Default is 3.
`cluster_by`	Method to use to create the categories. One of "bins" (equal-width bins), "quantiles" (quantile-based bins), or "kmeans" (k-means clustering). Default is "bins".

Value

A `node_member` character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

Core-periphery categories

This function categorizes nodes based on their coreness into a specified number of groups. The groups are labeled as "Core", "Semi-core", "Semi-periphery", and "Periphery" depending on the number of groups specified. The categorization can be done using different methods: equal-width bins, quantile-based bins, or k-means clustering.

References

On core-periphery categorization:

Wallerstein, Immanuel. 1974. "Dependence in an Interdependent World: The Limited Possibilities of Transformation Within the Capitalist World Economy." *African Studies Review*, 17(1), 1-26. doi:10.2307/523574

See Also

Other core-periphery: mark_core, measure_core

Other memberships: member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_diffusion, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
node_in_core(ison_adolescents)
```

---

member_diffusion *Memberships in a diffusion process*

Description

node_in_adopter() classifies membership of nodes into diffusion categories by where on the distribution of adopters they fell. Valente (1995) defines five memberships:

- *Early adopter*: those with an adoption time less than the average adoption time minus one standard deviation of adoptions times
- *Early majority*: those with an adoption time between the average adoption time and the average adoption time minus one standard deviation of adoptions times
- *Late majority*: those with an adoption time between the average adoption time and the average adoption time plus one standard deviation of adoptions times
- *Laggard*: those with an adoption time greater than the average adoption time plus one standard deviation of adoptions times
- *Non-adopter*: those without an adoption time, i.e. never adopted

Usage

```
node_in_adopter(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

References

On adopter classes:

Valente, Tom W. 1995. *Network models of the diffusion of innovations* (2nd ed.). Cresskill N.J.: Hampton Press.

See Also

Other memberships: member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_equivalence

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_equivalence, motif_brokerage_node, motif_exposure, motif_node, motif_path

Other diffusion: mark_diff, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, motif_exposure, motif_hazard

Examples

```
smeg <- generate_smallworld(15, 0.025)
smeg_diff <- play_diffusion(smeg, recovery = 0.2)
# To classify nodes by their position in the adoption curve
(adopts <- node_in_adopter(smeg_diff))
summary(adopts)
```

---

member_equivalence *Memberships in equivalent classes*

Description

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

- node_in_equivalence() assigns nodes membership based on their equivalence with respective to some motif/class. The following functions call this function, together with an appropriate motif.
- node_in_structural() assigns nodes membership based on their having equivalent ties to the same other nodes.
- node_in_regular() assigns nodes membership based on their having equivalent patterns of ties.
- node_in_automorphic() assigns nodes membership based on their having equivalent distances to other nodes.

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

Usage

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

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

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

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

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
motif	A matrix returned by a node_x_*() 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.
Kmax	Integer indicating the maximum number of (k) clusters to evaluate. Ignored when k = "strict" or a discrete number is given for k.

Value

A node_member character vector the length of the nodes in the network, of group memberships "A", "B", etc for each node. If the network is labelled, then the assignments will be labelled with the nodes' names.

Source

https://github.com/aslez/concoR

See Also

Other memberships: member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, motif_brokerage_node, motif_exposure, motif_node, motif_path

Examples

```
(nse <- node_in_structural(ison_algebra))
(nre <- node_in_regular(ison_southern_women,
  cluster = "concor"))
if(require("sna", quietly = TRUE)){
(nae <- node_in_automorphic(ison_southern_women,
  k = "elbow"))
}
```

---

method_cluster *Methods for equivalence clustering*

Description

These functions are used to cluster some motif census object:

- `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).
- `cluster_cosine()` returns a hierarchical clustering object created by `stats::hclust()` on cosine dissimilarities, rather than correlations, as created by `to_cosine()`.

These functions are not intended to be called directly, but are called within `node_in_equivalence()` and related functions. They are exported and listed here to provide more detailed documentation.

Usage

```
cluster_hierarchical(motif, distance)

cluster_cosine(motif, distance)

cluster_concor(.data, motif)
```

Arguments

`motif`	A matrix returned by a `node_x_*()` 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`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A hierarchical clustering object created by `stats::hclust()`, with an additional `distances` attribute containing the distance matrix used for clustering.

Hierarchical clustering

This method uses `stats::hclust()` to create a hierarchical clustering object from a distance matrix created from the correlations between nodes' profiles in the given motif census. First a matrix of Pearson correlation coefficients between each pair of nodes' profiles in the given motif census is created. Then a distance matrix is created by subtracting these correlations from 1, and this is given to `stats::hclust()` to enable dendrogram construction etc.

Cosine similarity

This method is similar to the hierarchical clustering method, but uses cosine similarity rather than correlation as the clustering basis. First a matrix of cosine similarities between each pair of nodes' profiles in the given census is created. Then a distance matrix is created by subtracting these similarities from 1, and this is given to `stats::hclust` to enable dendrogram construction etc.

CONCOR

First a matrix of Pearson correlation coefficients between each pair of nodes' profiles in the given motif 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

On CONCOR clustering:

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:10.1016/0022-2496(75)900280.

`method_kselect` *Methods for selecting clusters*

Description

Finding the optimal number of clusters is generally a balance between optimal fit statistics, parsimony, and interpretability. These functions help select the number of clusters to return from `hc`, some hierarchical clustering object:

- `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 maximises the silhouette score.

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

Usage

```
k_strict(hc, .data)

k_elbow(hc, .data, motif, Kmax)

k_silhouette(hc, .data, Kmax)

k_gap(hc, motif, Kmax, sims = 100)
```

Arguments

hc	A hierarchical clustering object.
.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
motif	A motif census object.
Kmax	An integer indicating the maximum number of options to consider. The minimum of this and the number of nodes in the network is used.
sims	Integer of how many simulations should be generated as a reference distribution.

Value

A single integer indicating the number of clusters to return.

Strict method

The strict method selects the number of clusters in which there is no distance between cluster members. This is a very conservative method that may be appropriate when the goal is to identify clusters of nodes that are exactly the same. However, it may not be appropriate in cases where the data is noisy or when the clusters are not well-defined, as it may result in a large number of small clusters.

Elbow method

The elbow method is a heuristic used in cluster analysis to determine the optimal number of clusters. It is based on the idea of plotting the within cluster correlation as a function of the number of clusters and looking for an "elbow" where there is a significant decrease in the rate of improvement in correlation as the number of clusters increases. The point at which the elbow occurs is often considered a good choice for the number of clusters, as it represents a balance between model complexity and fit to the data.

Silhouette method

The silhouette method is based on the concept of cohesion and separation. Cohesion refers to how closely related the nodes within a cluster are, while separation refers to how distinct the clusters are from each other. The silhouette score combines these two concepts into a single metric that can be used to evaluate the quality of a clustering solution. The silhouette score is calculated as follows: For each node, calculate the average distance to all other nodes in the same cluster (a) and the average distance to all other nodes in the next nearest cluster (b). The silhouette score for each node is then calculated as:

$$S(i) = \frac{b - a}{\max(a, b)}$$

A higher silhouette score indicates that the node is well-matched to its own cluster and poorly matched to neighboring clusters. The silhouette score for the entire clustering is the average silhouette score across all nodes. Maximizing the silhouette score across a range of potential clusterings allows researchers to identify the number of clusters that best captures the underlying structure of the data. It is particularly useful when the clusters are well-separated.

References

On the elbow method:

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

On the silhouette method:

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.

`motif_brokerage_net` *Motifs of network brokerage*

Description

`net_x_brokerage()` returns the Gould-Fernandez brokerage roles in a network.

Usage

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`membership`	A character string naming an existing node attribute in the network, or a categorical vector of the same length as the number of nodes in the network where each element indicates the group membership of the corresponding node. While this may often be a vector created using `node_in_*()` functions, it can be any character vector that assigns nodes to groups or categories.
`standardized`	Whether the score should be standardized into a z-score indicating how many standard deviations above or below the average the score lies.

Value

A `network_motif` named numeric vector or sometimes a data frame with one row and a column for each motif type, giving the count of each motif in the network. This is printed as a tibble to avoid greedy printing of long vectors.

See Also

Other brokerage: `measure_broker_node`, `measure_broker_tie`, `measure_brokerage`, `member_brokerage`, `motif_brokerage_node`

Other motifs: `motif_brokerage_node`, `motif_exposure`, `motif_hazard`, `motif_hierarchy`, `motif_net`, `motif_node`, `motif_path`, `motif_periods`

Examples

```
net_x_brokerage(ison_networkers, "Discipline")
```

`motif_brokerage_node` *Motifs of nodes brokerage*

Description

`node_x_brokerage()` returns the Gould-Fernandez brokerage roles played by nodes in a network.

Usage

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

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`membership`	A character string naming an existing node attribute in the network, or a categorical vector of the same length as the number of nodes in the network where each element indicates the group membership of the corresponding node. While this may often be a vector created using `node_in_*()` functions, it can be any character vector that assigns nodes to groups or categories.
`standardized`	Whether the score should be standardized into a z-score indicating how many standard deviations above or below the average the score lies.

Value

A `node_motif` matrix with one row for each node in the network and a column for each motif type, giving the count of each motif in which each node participates. It is printed as a tibble, however, to avoid greedy printing. If the network is labelled, then the node names will be in a column named `names`.

References

On brokerage motifs:

Gould, Roger V., and Roberto M. Fernandez. 1989. "Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks." *Sociological Methodology*, 19: 89-126. doi:10.2307/270949

Jasny, Lorien, and Mark Lubell. 2015. "Two-Mode Brokerage in Policy Networks." *Social Networks* 41:36–47. doi:10.1016/j.socnet.2014.11.005

See Also

Other brokerage: measure_broker_node, measure_broker_tie, measure_brokerage, member_brokerage, motif_brokerage_net

Other motifs: motif_brokerage_net, motif_exposure, motif_hazard, motif_hierarchy, motif_net, motif_node, motif_path, motif_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_exposure, motif_node, motif_path

Examples

```
node_x_brokerage(ison_networkers, "Discipline")
```

---

motif_exposure *Motifs of nodes exposure*

Description

node_x_exposure() produces a motif matrix of nodes' exposure to infection/adoption by time step.

Usage

```
node_x_exposure(.data)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().

Value

A node_motif matrix with one row for each node in the network and a column for each motif type, giving the count of each motif in which each node participates. It is printed as a tibble, however, to avoid greedy printing. If the network is labelled, then the node names will be in a column named names.

See Also

Other diffusion: mark_diff, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, member_diffusion, motif_hazard

Other motifs: motif_brokerage_net, motif_brokerage_node, motif_hazard, motif_hierarchy, motif_net, motif_node, motif_path, motif_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_node, motif_path

Examples

```
node_x_exposure(play_diffusion(create_tree(12)))
```

`motif_hazard` *Motifs of network hazard*

Description

`net_x_hazard()` measures the hazard rate or instantaneous probability that nodes will adopt/become infected at that time.

Usage

```
net_x_hazard(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `network_motif` named numeric vector or sometimes a data frame with one row and a column for each motif type, giving the count of each motif in the network. This is printed as a tibble to avoid greedy printing of long vectors.

Hazard rate

The hazard rate is the instantaneous probability of adoption/infection at each time point (Allison 1984). In survival analysis, hazard rate is formally defined as:

$$\lambda(t) = \lim_{h \to +0} \frac{F(t+h) - F(t)}{h} \frac{1}{1 - F(t)}$$

By approximating $h = 1$, we can rewrite the equation as

$$\lambda(t) = \frac{F(t+1) - F(t)}{1 - F(t)}$$

If we estimate $F(t)$, the probability of not having adopted the innovation in time t, from the proportion of adopters in that time, such that $F(t) \sim q_t/n$, we now have (ultimately for $t > 1$):

$$\lambda(t) = \frac{q_{t+1}/n - q_t/n}{1 - q_t/n} = \frac{q_{t+1} - q_t}{n - q_t} = \frac{q_t - q_{t-1}}{n - q_{t-1}}$$

where q_i is the number of adopters in time t, and n is the number of vertices in the graph.

The shape of the hazard rate indicates the pattern of new adopters over time. Rapid diffusion with convex cumulative adoption curves will have hazard functions that peak early and decay over time. Slow concave cumulative adoption curves will have hazard functions that are low early and rise over time. Smooth hazard curves indicate constant adoption whereas those that oscillate indicate variability in adoption behavior over time.

Source

{netdiffuseR}

References

On hazard rates:

Allison, Paul D. 1984. *Event history analysis: Regression for longitudinal event data*. London: Sage Publications. doi:10.4135/9781412984195

Wooldridge, Jeffrey M. 2010. *Econometric Analysis of Cross Section and Panel Data* (2nd ed.). Cambridge: MIT Press.

See Also

Other diffusion: mark_diff, measure_diffusion_infection, measure_diffusion_net, measure_diffusion_node, member_diffusion, motif_exposure

Other motifs: motif_brokerage_net, motif_brokerage_node, motif_exposure, motif_hierarchy, motif_net, motif_node, motif_path, motif_periods

Examples

```
# To calculate the hazard rates at each time point
  smeg <- generate_smallworld(15, 0.025)
net_x_hazard(play_diffusion(smeg, transmissibility = 0.3))
```

---

motif_hierarchy *Motifs of network hierarchy*

Description

net_x_hierarchy() collects the measures of hierarchy into a single motif, which can be used to compare the relative hierarchy of different networks. The measures of hierarchy are:

- net_by_connectedness() measures the proportion of dyads in the network that are reachable to one another, or the degree to which network is a single component.
- net_by_efficiency() measures the Krackhardt efficiency score.

- `net_by_upperbound()` measures the Krackhardt (least) upper bound score.
- `net_by_reciprocity()` measures the proportion of ties in the network that are reciprocated, which is a measure of the degree to which the network is non-hierarchical.

Usage

```
net_x_hierarchy(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `network_motif` named numeric vector or sometimes a data frame with one row and a column for each motif type, giving the count of each motif in the network. This is printed as a tibble to avoid greedy printing of long vectors.

References

On hierarchy:

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 hierarchy: `measure_hierarchy`

Other motifs: `motif_brokerage_net`, `motif_brokerage_node`, `motif_exposure`, `motif_hazard`, `motif_net`, `motif_node`, `motif_path`, `motif_periods`

Examples

```
net_x_hierarchy(ison_networkers)
```

`motif_net` *Motifs of network cohesion*

Description

These functions include ways to take a census of the graphlets in a network:

- `net_x_dyad()` returns a census of dyad motifs in a network.
- `net_x_triad()` returns a census of triad motifs in a network.
- `net_x_tetrad()` returns a census of tetrad motifs in a network.
- `net_x_mixed()` returns a census of triad motifs that span a one-mode and a two-mode network.

See also graph classes.

Usage

```
net_x_dyad(.data)

net_x_triad(.data)

net_x_tetrad(.data)

net_x_mixed(.data, object2)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.
`object2`	A second, two-mode network object.

Value

A `network_motif` named numeric vector or sometimes a data frame with one row and a column for each motif type, giving the count of each motif in the network. This is printed as a tibble to avoid greedy printing of long vectors.

Dyad census

The dyad census counts the number of mutual, asymmetric, and null dyads in a network. For directed networks,

- Mutual dyads have ties in both directions
- Asymmetric dyads have a tie in one direction only
- Null dyads have no ties

Note that for undirected and two-mode networks, only mutual and null dyads are possible, as the concept of an asymmetric dyad does not apply.

Triad census

The triad census counts the number of three-node configurations in the network. The function returns a matrix with a special naming convention:

- 003: This is an empty triad; no ties
- 012: This triad includes one tie
- 102: This triad includes two ties, but they are not reciprocated
- 021D: This triad includes two ties, one of which is reciprocated, and the other is directed towards the reciprocated tie
- 021U: This triad includes two ties, one of which is reciprocated, and the other is directed away from the reciprocated tie
- 021C: This triad includes two ties, one of which is reciprocated, and the other is directed between the two non-reciprocated nodes
- 111D: This triad includes three ties, two of which are reciprocated, and the other is directed towards the reciprocated ties
- 111U: This triad includes three ties, two of which are reciprocated, and the other is directed away from the reciprocated ties
- 030T: This triad includes three ties, all of which are directed in a transitive manner (i.e. A->B, B->C, A->C)
- 030C: This triad includes three ties, all of which are directed in a cyclic manner (i.e. A->B, B->C A->C)
- 201: This triad includes three ties, all of which are reciproc ated (i.e. A<->B, B<->C, A<->C)
- 120D: This triad includes four ties, three of which are reciprocated, and the other is directed towards the reciprocated ties
- 120U: This triad includes four ties, three of which are reciprocated, and the other is directed away from the reciprocated ties
- 120C: This triad includes four ties, three of which are reciprocated, and the other is directed between the two non-reciprocated
- 210: This triad includes five ties, four of which are reciprocated, and the other is directed between the two non-reciprocated
- 300: This triad includes six ties, all of which are reciprocated

Note that for undirected and two-mode networks, only 003, 102, and 201 are possible, as the other configurations rely on the concept of directionality.

Tetrad census

The tetrad census counts the number of four-node configurations in the network. The function returns a matrix with a special naming convention:

- E4 (aka co-K4): This is an empty set of four nodes; no ties
- I4 (aka co-diamond): This is a set of four nodes with just one tie
- H4 (aka co-C4): This set of four nodes includes two non-adjacent ties
- L4 (aka co-paw): This set of four nodes includes two adjacent ties

- D4 (aka co-claw): This set of four nodes includes three adjacent ties, in the form of a triangle with one isolate
- U4 (aka P4, four-actor line): This set of four nodes includes three ties arranged in a line
- Y4 (aka claw): This set of four nodes includes three ties all adjacent to a single node
- P4 (aka paw, kite): This set of four nodes includes four ties arranged as a triangle with an extra tie hanging off of one of the nodes
- C4 (aka bifan): This is a symmetric box or 4-cycle or set of shared choices
- Z4 (aka diamond): This resembles C4 but with an extra tie cutting across the box
- X4 (aka K4): This resembles C4 but with two extra ties cutting across the box; a realisation of all possible ties

Graphs of these motifs can be shown using `plot(net_x_tetrad(ison_southern_women))`.

Source

Alejandro Espinosa 'netmem'

References

On the dyad census:

Holland, Paul W., and Samuel Leinhardt. 1970. "A Method for Detecting Structure in Sociometric Data". *American Journal of Sociology*, 76: 492-513. doi:10.1016/B9780124424500.500286

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

On the triad census:

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

On the tetrad census:

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/s41109017-00272.

McMillan, Cassie, and Diane Felmlee. 2020. "Beyond Dyads and Triads: A Comparison of Tetrads in Twenty Social Networks". *Social Psychology Quarterly* 83(4): 383-404. doi:10.1177/0190272520944151

On the mixed census:

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 cohesion: `mark_triangles`, `measure_breadth`, `measure_cohesion`, `measure_fragmentation`, `motif_node`

Other motifs: `motif_brokerage_net`, `motif_brokerage_node`, `motif_exposure`, `motif_hazard`, `motif_hierarchy`, `motif_node`, `motif_path`, `motif_periods`

Examples

```
net_x_dyad(manynet::ison_algebra)
net_x_triad(manynet::ison_adolescents)
net_x_tetrad(ison_southern_women)
net_x_mixed(fict_marvel)
```

`motif_node` *Motifs of nodes cohesion*

Description

These functions include ways to take a census of the positions of nodes in a network:

- `node_x_tie()` returns a census of the ties in a network. For directed networks, out-ties and in-ties are bound together. For multiplex networks, the various types of ties are bound together.
- `node_x_triad()` returns a census of the triad configurations nodes are embedded in.
- `node_x_tetrad()` returns a census of nodes' positions in motifs of four nodes.
- `node_x_path()` returns the shortest path lengths of each node to every other node in the network.

Usage

```
node_x_dyad(.data)

node_x_triad(.data)

node_x_tetrad(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `node_motif` matrix with one row for each node in the network and a column for each motif type, giving the count of each motif in which each node participates. It is printed as a tibble, however, to avoid greedy printing. If the network is labelled, then the node names will be in a column named `names`.

Tetrad census

The nodal tetrad census counts the number of four-node configurations that each node is embedded in. The function returns a matrix with a special naming convention:

- E4 (aka co-K4): This is an empty set of four nodes; no ties
- I4 (aka co-diamond): This is a set of four nodes with just one tie

- H4 (aka co-C4): This set of four nodes includes two non-adjacent ties
- L4 (aka co-paw): This set of four nodes includes two adjacent ties
- D4 (aka co-claw): This set of four nodes includes three adjacent ties, in the form of a triangle with one isolate
- U4 (aka P4, four-actor line): This set of four nodes includes three ties arranged in a line
- Y4 (aka claw): This set of four nodes includes three ties all adjacent to a single node
- P4 (aka paw, kite): This set of four nodes includes four ties arranged as a triangle with an extra tie hanging off of one of the nodes
- C4 (aka bifan): This is a symmetric box or 4-cycle or set of shared choices
- Z4 (aka diamond): This resembles C4 but with an extra tie cutting across the box
- X4 (aka K4): This resembles C4 but with two extra ties cutting across the box; a realisation of all possible ties

Graphs of these motifs can be shown using `plot(node_by_tetrad(ison_southern_women))`.

References

On the dyad census:

Holland, Paul W., and Samuel Leinhardt. 1970. "A Method for Detecting Structure in Sociometric Data". *American Journal of Sociology*, 76: 492-513. doi:10.1016/B9780124424500.500286

On the triad census:

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

On the tetrad census:

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/s41109017-00272.

McMillan, Cassie, and Diane Felmlee. 2020. "Beyond Dyads and Triads: A Comparison of Tetrads in Twenty Social Networks". *Social Psychology Quarterly* 83(4): 383-404. doi:10.1177/0190272520944151

See Also

Other cohesion: `mark_triangles`, `measure_breadth`, `measure_cohesion`, `measure_fragmentation`, `motif_net`

Other motifs: `motif_brokerage_net`, `motif_brokerage_node`, `motif_exposure`, `motif_hazard`, `motif_hierarchy`, `motif_net`, `motif_path`, `motif_periods`

Other nodal: `mark_core`, `mark_degree`, `mark_diff`, `mark_nodes`, `mark_select_node`, `measure_assort_node`, `measure_broker_node`, `measure_brokerage`, `measure_central_between`, `measure_central_close`, `measure_central_degree`, `measure_central_eigen`, `measure_closure_node`, `measure_core`, `measure_diffusion_node`, `measure_diverse_node`, `member_brokerage`, `member_cliques`, `member_community`, `member_community_hier`, `member_community_non`, `member_components`, `member_core`, `member_diffusion`, `member_equivalence`, `motif_brokerage_node`, `motif_exposure`, `motif_path`

Examples

```
node_x_dyad(ison_networkers)
task_eg <- to_named(to_uniplex(ison_algebra, "tasks"))
(triad_cen <- node_x_triad(task_eg))
node_x_tetrad(ison_southern_women)
```

`motif_path` *Motifs of nodes pathing*

Description

These functions include ways to take a census of the positions of nodes in a network:

- `node_x_tie()` returns a census of the ties in a network. For directed networks, out-ties and in-ties are bound together. For multiplex networks, the various types of ties are bound together.
- `node_x_path()` returns the shortest path lengths of each node to every other node in the network.

Usage

```
node_x_tie(.data)

node_x_path(.data)
```

Arguments

`.data`	A network object of class `mnet`, `igraph`, `tbl_graph`, `network`, or similar. For more information on the standard coercion possible, see `manynet::as_tidygraph()`.

Value

A `node_motif` matrix with one row for each node in the network and a column for each motif type, giving the count of each motif in which each node participates. It is printed as a tibble, however, to avoid greedy printing. If the network is labelled, then the node names will be in a column named `names`.

References

On paths:

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: motif_brokerage_net, motif_brokerage_node, motif_exposure, motif_hazard, motif_hierarchy, motif_net, motif_node, motif_periods

Other nodal: mark_core, mark_degree, mark_diff, mark_nodes, mark_select_node, measure_assort_node, measure_broker_node, measure_brokerage, measure_central_between, measure_central_close, measure_central_degree, measure_central_eigen, measure_closure_node, measure_core, measure_diffusion_node, measure_diverse_node, member_brokerage, member_cliques, member_community, member_community_hier, member_community_non, member_components, member_core, member_diffusion, member_equivalence, motif_brokerage_node, motif_exposure, motif_node

Examples

```
task_eg <- to_named(to_uniplex(ison_algebra, "tasks"))
(tie_cen <- node_x_tie(task_eg))
node_x_path(ison_adolescents)
node_x_path(ison_southern_women)
```

---

motif_periods *Motifs of network change*

Description

These functions measure certain topological features of networks:

- net_x_change() measures the Hamming distance between two or more networks.
- net_x_stability() measures the Jaccard index of stability between two or more networks.
- net_x_correlation() measures the product-moment correlation between two networks.

These net_*() functions return a numeric vector the length of the number of networks minus one. E.g., the periods between waves.

Usage

```
net_x_change(.data, object2)

net_x_stability(.data, object2)

net_x_correlation(.data, object2)
```

Arguments

.data	A network object of class mnet, igraph, tbl_graph, network, or similar. For more information on the standard coercion possible, see manynet::as_tidygraph().
object2	A network object.

Value

A `network_motif` named numeric vector or sometimes a data frame with one row and a column for each motif type, giving the count of each motif in the network. This is printed as a tibble to avoid greedy printing of long vectors.

See Also

Other change: `measure_periods`

Other motifs: `motif_brokerage_net`, `motif_brokerage_node`, `motif_exposure`, `motif_hazard`, `motif_hierarchy`, `motif_net`, `motif_node`, `motif_path`

Index