Package 'migraph'

April 15, 2022

Title Multimodal and Multilevel Network Analysis

Version 0.9.3

Date 2022-04-14

Description A set of tools that extend common social network analysis packages for analysing multimodal and multilevel networks. It includes functions for one- and two-mode (and sometimes three-mode) centrality, centralization, clustering, and constraint, as well as for one- and two-mode network regression and block-modelling. All functions operate with matrices, edge lists, and 'igraph', 'network'/'sna', and 'tidygraph' objects. The package is released as a complement to 'Multimodal Political Networks' (2021, ISBN:9781108985000), and includes various datasets used in the book.

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

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

Depends R (>= 3.6.0)

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports concaveman, dplyr, generics, ggdendro, ggforce, ggplot2, ggraph, igraph, network, oaqc, future, furrr, patchwork, purrr, rlang, sna, stringr, tibble, tidygraph, tidyr

Suggests covr, knitr, readxl, rmarkdown, roxygen2, testthat

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2022-04-14 23:32:29 UTC

R topics documented:

add . . . 3
autographr . . . 4
blockmodel . . . 5
blockmodel_vis . . . 7
census . . . 8
centrality . . . 9
cluster . . . 12
coercion . . . 13
cohesion . . . 14
connectedness . . . 16
create . . . 18
diversity . . . 19
generate . . . 20
ggatyear . . . 22
ggevolution . . . 22
gglineage . . . 23
grab . . . 24
graph_balance . . . 25
graph_census . . . 26
graph_smallworld . . . 27
is . . . 28
ison_adolescents . . . 30
ison_algebra . . . 30
ison_brandes . . . 31
ison_karateka . . . 31
ison_marvel . . . 32
ison_networkers . . . 33
ison_projection . . . 34
ison_southern_women . . . 34
layouts . . . 35
mpn_bristol . . . 36
mpn_elite_mex . . . 36
mpn_elite_usa . . . 37
mpn_evs . . . 38
mpn_ryanair . . . 40
mpn_senate112 . . . 40
node_constraint . . . 41
read . . . 42
regression . . . 45
tests . . . 47
to . . . 48

Index **52**

---

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

Description

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

Usage

```
add_node_attributes(object, attr_name, vector)

add_edge_attributes(object, attr_name, vector)

copy_node_attributes(object, object2)

join_edges(object, object2, attr_name)
```

Arguments

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

Functions

- `add_node_attributes`: Insert specified values from a vector into the graph as node attributes
- `add_edge_attributes`: Insert specified values from a vector into the graph as edge attributes
- `copy_node_attributes`: Copies node attributes from a given graph into specified graph
- `join_edges`: Copies edges from another graph to specified graph and adds an edge attribute identifying the edges that were newly added

See Also

Other manipulation: `coercion`, `is()`, `to`

Examples

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

`autographr` *Quickly graph networks with sensible defaults*

Description

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

Usage

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

Arguments

`object`	A migraph-consistent object.
`layout`	An igraph layout algorithm, currently defaults to 'stress'.
`labels`	Logical, whether to print node names as labels if present.
`node_color`	Node variable in quotation marks to be used for colouring the nodes. It is easiest if this is added as a node attribute to the graph before plotting.
`node_group`	Node variable in quotation marks to be used for drawing convex but also concave hulls around clusters of nodes. These groupings will be labelled with the categories of the variable passed.

node_shape	Character string in quotation marks referring to the name of a node attribute already present in the graph to be used for the shapes of the nodes. Shapes follow the ordering "circle", "square", "triangle", so this aesthetic should be used for a variable with only a few categories.
node_size	Node variable in quotation marks to be used for the size of the nodes. This can be any continuous variable on the nodes of the network. Since this function expects this to be an existing variable, it is recommended to calculate all node-related statistics prior to using this function.
node_measure	Name of the node level measure function e.g. node_degree. NULL by default.
identify_function	Name of the function used to determine the highlighted node e.g. max, min, etc. max by default.
...	Extra arguments.

Examples

```
ison_adolescents <- ison_adolescents %>%
 dplyr::mutate(shape = rep(c("circle", "square"), times = 4)) %>%
 dplyr::mutate(color = rep(c("blue", "red"), times = 4))
autographr(ison_adolescents, node_shape = "shape", node_color = "color")
autographr(ison_karateka, node_size = rep(c(0.8), times = 34))
autographr(ison_networkers, node_measure = node_betweenness, identify_function = max)
```

---

blockmodel *Blockmodelling*

Description

Blockmodelling

Usage

```
blockmodel(object, clusters)

blockmodel_concor(
  object,
  p = 1,
  cutoff = 0.999,
  max.iter = 25,
  block.content = "density"
)

## S3 method for class 'block_model'
print(x, ...)

reduce_graph(blockmodel, block_labels = NULL)

summarise_statistics(node_measure, clusters = NULL, sumFUN = mean)
```

Arguments

object	A migraph-consistent object (matrix, igraph, tidygraph).
clusters	the vector of cluster membership for the blockmodel
p	An integer representing the desired number of partitions.
cutoff	A value between 0 and 1 used to determine convergence.
max.iter	An integer representing the maximum number of iterations.
block.content	A string indicating which method to use for calculating block content. Options are: "density", "sum", "meanrowsum", "meancolsum", "median", "min", "max".
x	An object of class "block_model"
...	Additional arguments passed to generic print method
blockmodel	a blockmodel object
block_labels	A character vector manually providing labels for the blocks in the blockmodel
node_measure	A vector or matrix of node-level statistics, such as centrality measures or a census.
sumFUN	A function by which the values should be aggregated or summarised. By default mean.

Source

https://github.com/aslez/concoR

References

Breiger, R.L., Boorman, S.A., and Arabie, P. 1975. An Algorithm for Clustering Relational Data with Applications to Social Network Analysis and Comparison with Multidimensional Scaling. *Journal of Mathematical Psychology*, 12: 328–383.

Examples

```
mex_concor <- blockmodel_concor(mpn_elite_mex)
mex_concor
plot(mex_concor)
usa_concor <- blockmodel_concor(mpn_elite_usa_advice)
usa_concor
plot(usa_concor)
summarise_statistics(node_degree(mpn_elite_mex),
          cutree(cluster_structural_equivalence(mpn_elite_mex), 3))
summarise_statistics(node_triad_census(mpn_elite_mex),
          cutree(cluster_structural_equivalence(mpn_elite_mex), 3))
```

`blockmodel_vis` *ggplot2-based plotting of blockmodel results*

Description

ggplot2-based plotting of blockmodel results

Plots for deciding on the number of network clusters

Usage

```
## S3 method for class 'block_model'
plot(x, ...)

ggtree(hc, k = NULL)

ggidentify_clusters(hc, census, method = c("elbow", "strict"))
```

Arguments

`x`	A blockmodel-class object.
`...`	Additional arguments passed on to ggplot2.
`hc`	a hierarchical cluster object
`k`	number of clusters. By default NULL, but, if specified, `ggtree` will color branches and add a line to indicate where the corresponding cluster cut would be.
`census`	output from some node_*_census function
`method`	only "elbow" is currently implemented.

Examples

```
# Unsigned
usa_concor <- blockmodel_concor(mpn_elite_usa_advice)
plot(usa_concor)

# Signed network (Needs sign)
# marvel_concor <- blockmodel_concor(ison_marvel_relationships)
# plot(marvel)
res <- cluster_regular_equivalence(mpn_elite_mex)
ggtree(res, 3)
ggidentify_clusters(res, node_triad_census(mpn_elite_mex))
```

census *Census by nodes or clusters*

Description

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

Usage

```
node_tie_census(object)

node_triad_census(object)

node_quad_census(object)

group_tie_census(object, clusters, decimals = 2)

group_triad_census(object, clusters, decimals = 2)
```

Arguments

object	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
clusters	a vector of cluster assignment.
decimals	Number of decimal points to round to.

Details

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

migraph	Ortmann and Brandes
E4	co-K4
I40, I41	co-diamond
H4	co-C4
L42, L41, L40	co-paw
D42, D40	co-claw

U42, U41	P4
Y43, Y41	claw
P43, P42, P41	paw
04	C4
Z42, Z43	diamond
X4	K4

Functions

- node_quad_census: Returns a census of nodes' positions in motifs of four nodes.

References

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

Examples

```
task_eg <- to_named(to_uniplex(ison_algebra, "task_tie"))
(tie_cen <- node_tie_census(task_eg))
(triad_cen <- node_triad_census(task_eg))
(quad_cen <- node_quad_census(ison_southern_women))
group_tie_census(task_eg, cutree(cluster_structural_equivalence(task_eg), 4))
group_triad_census(task_eg, cutree(cluster_regular_equivalence(task_eg), 4))
```

---

centrality *Centrality and centralization for one- and two-mode networks*

Description

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

Usage

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

edge_degree(object, normalized = TRUE)

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

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

edge_closeness(object, normalized = TRUE)
```

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

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

edge_betweenness(object, normalized = TRUE)

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

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

edge_eigenvector(object, normalized = TRUE)

graph_eigenvector(object, normalized = TRUE)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
`normalized`	Logical scalar, whether the centrality scores are normalized. Different denominators are used depending on whether the object is one-mode or two-mode, the type of centrality, and other arguments.
`direction`	Character string, “out” bases the measure on outgoing ties, “in” on incoming ties, and "all" on either/the sum of the two. For two-mode networks, "all" uses as numerator the sum of differences between the maximum centrality score for the mode against all other centrality scores in the network, whereas "in" uses as numerator the sum of differences between the maximum centrality score for the mode against only the centrality scores of the other nodes in that mode.
`cutoff`	Maximum path length to use during calculations.
`nobigint`	Whether big integers should be avoided during calculations
`scale`	Logical scalar, whether to rescale the vector so the maximum score is 1.

Value

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

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

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

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

Functions

- `node_degree`: Calculates the degree centrality of nodes in an unweighted network, or weighted degree/strength of nodes in a weighted network.
- `edge_degree`: Calculate the degree centrality of edges in a network
- `graph_degree`: Calculate the degree centralization for a graph
- `node_closeness`: Calculate the closeness centrality of nodes in a network
- `edge_closeness`: Calculate the closeness of each edge to each other edge in the network.
- `graph_closeness`: Calculate the closeness centralization for a graph
- `node_betweenness`: Calculate the betweenness centralities of nodes in a network
- `edge_betweenness`: Calculate number of shortest paths going through an edge
- `graph_betweenness`: Calculate the betweenness centralization for a graph
- `node_eigenvector`: Calculate the eigenvector centrality of nodes in a network
- `edge_eigenvector`: Calculate the eigenvector centrality of edges in a network
- `graph_eigenvector`: Calculate the eigenvector centralization for a graph

References

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

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

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

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

See Also

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

Examples

```
node_degree(mpn_elite_mex)
node_degree(ison_southern_women)
edge_degree(ison_adolescents)
graph_degree(ison_southern_women, direction = "in")
node_closeness(mpn_elite_mex)
node_closeness(ison_southern_women)
(ec <- edge_closeness(ison_adolescents))
plot(ec)
ison_adolescents %>%
  activate(edges) %>% mutate(weight = ec) %>%
  autographr()
```

```
graph_closeness(ison_southern_women, direction = "in")
node_betweenness(mpn_elite_mex)
node_betweenness(ison_southern_women)
(eb <- edge_betweenness(ison_adolescents))
plot(eb)
ison_adolescents %>%
  activate(edges) %>% mutate(weight = eb) %>%
  autographr()
graph_betweenness(ison_southern_women, direction = "in")
node_eigenvector(mpn_elite_mex)
node_eigenvector(ison_southern_women)
edge_eigenvector(ison_adolescents)
graph_eigenvector(mpn_elite_mex)
graph_eigenvector(ison_southern_women)
```

`cluster` *Clustering algorithms*

Description

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

Usage

```
cluster_structural_equivalence(object)

cluster_regular_equivalence(object)
```

Arguments

`object`	A migraph-consistent object.

Examples

```
ggtree(cluster_structural_equivalence(mpn_elite_mex))
ggtree(cluster_regular_equivalence(mpn_elite_mex))
ggtree(cluster_regular_equivalence(mpn_elite_usa_advice))
```

coercion *Coercion between migraph-compatible object classes*

Description

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

- edgelists, as data frames or tibbles
- adjacency and incidence matrices
- {igraph} graph objects
- {tidygraph} tbl_graph objects
- {network} network objects

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

Usage

```
as_edgelist(object)

as_matrix(object)

as_igraph(object, twomode = FALSE)

as_tidygraph(object, twomode = FALSE)

as_network(object)
```

Arguments

object	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
twomode	Logical option used to override the heuristics for distinguishing incidence from adjacency matrices. By default FALSE.

Details

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

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

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

Value

The currently implemented coercions or translations are:

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

See Also

Other manipulation: `add`, `is()`, `to`

Examples

```
test <- data.frame(id1 = c("A","B","B","C","C"),
                   id2 = c("I","G","I","G","H"))
as_edgelist(test)
as_matrix(test)
as_igraph(test)
as_tidygraph(test)
as_network(test)
```

---

cohesion *Cohesion for one-, two-, and three- mode networks*

Description

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

Usage

```
graph_density(object)

edge_reciprocal(object)

graph_reciprocity(object, method = "default")

graph_transitivity(object)

graph_equivalency(object)

graph_congruency(object, object2)

edge_multiple(object)

edge_loop(object)
```

Arguments

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

Details

For one- and two-mode networks, graph_density summarises the ratio of ties to the number of possible ties.

For one-mode networks, shallow wrappers of igraph versions exist via graph_reciprocity and graph_transitivity.

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

Functions

- edge_reciprocal: Identify edges that are mutual/reciprocated
- graph_reciprocity: Calculate reciprocity in a network
- graph_transitivity: Calculate transitivity in a network
- graph_equivalency: Calculate equivalence or reinforcement in a network
- graph_congruency: Calculate congruency in a network
- edge_multiple: Identify edges that are multiples
- edge_loop: Identify edges that are loops

References

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

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

See Also

Other one-mode measures: `node_constraint()`

Other two-mode measures: `graph_smallworld()`, `node_constraint()`

Examples

```
graph_density(mpn_elite_mex)
graph_density(mpn_elite_usa_advice)
edge_reciprocal(ison_algebra)
graph_reciprocity(ison_southern_women)
graph_transitivity(ison_southern_women)
graph_equivalency(ison_southern_women)
edge_multiple(ison_algebra)
edge_loop(ison_algebra)
```

---

connectedness *Network connectedness*

Description

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

Usage

```
graph_components(object, method = c("weak", "strong"))

graph_cohesion(object)

graph_adhesion(object)

graph_length(object)

graph_diameter(object)

node_components(object, method = c("weak", "strong"))

node_cuts(object)

edge_bridges(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
`method`	For directed networks, either `weak` if edge direction is irrelevant, or `strong` if edge direction is salient. Ignored if network undirected.

Functions

- `graph_components`: Returns number of components in the network.
- `graph_cohesion`: Returns the minimum number of nodes needed to remove from the network to increase the number of components.
- `graph_adhesion`: Returns the minimum number of edges needed to remove from the network to increase the number of components.
- `graph_length`: Returns the average path length in the network.
- `graph_diameter`: Returns the maximum path length in the network.
- `node_components`: Returns nodes' component membership.
- `node_cuts`: Returns logical of which nodes cut or act as articulation points in a network.
- `edge_bridges`: Returns logical of which nodes cut or act as articulation points in a network.

References

White, Douglas R and Frank Harary 2001. The Cohesiveness of Blocks In Social Networks: Node Connectivity and Conditional Density. *Sociological Methodology* 31 (1) : 305-359.

Examples

```
graph_cohesion(ison_marvel_relationships)
graph_cohesion(to_main_component(ison_marvel_relationships))
graph_adhesion(ison_marvel_relationships)
graph_adhesion(to_main_component(ison_marvel_relationships))
graph_length(ison_marvel_relationships)
graph_length(to_main_component(ison_marvel_relationships))
graph_diameter(ison_marvel_relationships)
graph_diameter(to_main_component(ison_marvel_relationships))
```

`create` *Create networks with particular structures*

Description

These functions create networks with particular structural properties. They can create either one-mode and two-mode networks, depending on whether the common `n` argument is passed a single integer (the number of nodes in the one-mode network) or a vector of *two* integers to return a two-mode network.

Usage

```
create_empty(n)

create_complete(n)

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

create_components(n, components = 2)

create_star(n, direction = c("undirected", "in", "out"))

create_tree(n, direction = c("undirected", "in", "out"), branches = 2)

create_lattice(n, direction = c("undirected", "in", "out"))
```

Arguments

`n`	Given: • A single integer, e.g. `n = 10`, a one-mode network will be created. • A vector of two integers, e.g. `n = c(5,10)`, a two-mode network will be created. • A migraph-compatible object, a network of the same dimensions will be created.
`width`	The width or breadth of the ring. This is typically double the degree.
`directed`	Whether the graph should be directed. By default FALSE.
`...`	Additional arguments passed on to igraph.
`components`	Number of components to divide the nodes into.
`direction`	One of the following options: "in", "out", or "undirected" (DEFAULT).
`branches`	How many branches at each level.

Value

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

Functions

- `create_empty`: Creates an empty graph of the given dimensions.
- `create_complete`: Creates a filled graph of the given dimensions, with every possible tie realised.
- `create_ring`: Creates a ring or chord graph of the given dimensions that loops around is of a certain width or thickness.
- `create_components`: Creates a graph in which the nodes are clustered into separate components.
- `create_star`: Creates a graph of the given dimensions that has a maximally central node
- `create_tree`: Creates a graph of the given dimensions with successive branches.
- `create_lattice`: Creates a graph of the given dimensions with ties to all neighbouring nodes

See Also

as_matrix as_tidygraph as_network

Other creation: `generate`

Examples

```
autographr(create_empty(c(8,6))) +
autographr(create_complete(c(8,6)))
autographr(create_ring(8, width = 2)) +
autographr(create_ring(c(8,6), width = 2))
autographr(create_components(c(10, 12), components = 3))
autographr(create_star(12, "in")) +
autographr(create_star(12, "out")) +
autographr(create_star(c(12,1), "in"))
autographr(create_tree(15, direction = "out")) +
autographr(create_tree(15, direction = "out"), "tree") +
autographr(create_tree(15, direction = "out", branches = 3), "tree")
autographr(create_lattice(5), layout = "kk") +
autographr(create_lattice(c(5,5))) +
autographr(create_lattice(c(5,5,5)))
```

`diversity` *Measures of network diversity*

Description

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

Usage

```
graph_blau_index(object, attribute, clusters = NULL)

graph_ei_index(object, attribute)

graph_assortativity(object)
```

Arguments

object	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
attribute	The name of a vertex attribute to measure the diversity of.
clusters	A nodal cluster membership vector or name of a vertex attribute.

Functions

- graph_assortativity: Calculates the degree assortativity in a graph

Examples

```
marvel_friends <- to_unsigned(ison_marvel_relationships, "positive")
graph_blau_index(marvel_friends, "Gender")
graph_blau_index(marvel_friends, "Attractive")
graph_blau_index(marvel_friends, "Gender", "Rich")
graph_ei_index(marvel_friends, "Gender")
graph_ei_index(marvel_friends, "Attractive")
graph_assortativity(mpn_elite_mex)
```

---

generate *Create networks from particular probabilities*

Description

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

Usage

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

generate_smallworld(n, p = 0.05)

generate_scalefree(n, p = 1)

generate_permutation(object, with_attr = TRUE)
```

Arguments

n	Given: • A single integer, e.g. n = 10, a one-mode network will be created. • A vector of two integers, e.g. n = c(5,10), a two-mode network will be created. • A migraph-compatible object, a network of the same dimensions will be created.
p	Proportion of possible edges in the network that are realised or, if integer greater than 1, the number of edges in the network.
directed	Whether to generate network as directed. By default FALSE.
object	a migraph-consistent object
with_attr	Logical. Whether any attributes of the object should be retained. By default TRUE.

Functions

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

See Also

Other creation: create

Examples

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

ggatyear *Plotting a network at a particular timepoint (year)*

Description

Plotting a network at a particular timepoint (year)

Usage

```
ggatyear(edgelist, year, ...)
```

Arguments

edgelist	A manyverse edgelist, expecting Beg and End variables, among others
year	Numeric year, gets expanded to first of January that year
...	Additional arguments passed on to autographr().

Value

A plot of the network of agreements signed in the specified year.

Examples

```
## Not run:
ggatyear(membs, 1900)

## End(Not run)
```

ggevolution *Plot the evolution of a network*

Description

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

Usage

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

Arguments

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

Examples

```
mpn_elite_mex <- mpn_elite_mex %>% to_subgraph(in_mpn == 1)
mpn_elite_mex2 <- mpn_elite_mex  %>%
                  tidygraph::activate(edges) %>%
tidygraph::reroute(from = sample.int(11, 44, replace = TRUE),
to = sample.int(11, 44, replace = TRUE))
ggevolution(mpn_elite_mex, mpn_elite_mex2)
ggevolution(mpn_elite_mex, mpn_elite_mex2, based_on = "last")
ggevolution(mpn_elite_mex, mpn_elite_mex2, based_on = "both")
```

---

gglineage *Plot lineage graph*

Description

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

Usage

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

Arguments

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

Examples

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

grab *Helpers to grab various attributes from nodes or edges in a graph*

Description

Helpers to grab various attributes from nodes or edges in a graph

Usage

```
node_names(object)

node_mode(object)

node_attribute(object, attribute)

edge_attribute(object, attribute)

edge_weights(object)

edge_signs(object)

graph_nodes(object)

graph_edges(object)

graph_dims(object)

graph_node_attributes(object)

graph_edge_attributes(object)
```

Arguments

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

Functions

- node_names: Extracts the names of the nodes in a network.
- node_mode: Extracts the mode of the nodes in a network.

- node_attribute: Extracts an attribute's values for the nodes in a network.
- edge_attribute: Extracts an attribute's values for the edges in a network.
- edge_weights: Extracts the weights of the edges in a network.
- edge_signs: Extracts the signs of the edges in a network.
- graph_nodes: Returns the number of nodes (of any mode) in a network.
- graph_edges: Returns the number of edges in a network.
- graph_dims: Returns the dimensions of a network in a vector as long as the number of modes in the network.
- graph_node_attributes: Returns a vector of nodal attributes in a network
- graph_edge_attributes: Returns a vector of edge attributes in a network

Examples

```
node_names(mpn_elite_usa_advice)
node_mode(mpn_elite_usa_advice)
node_attribute(mpn_elite_mex, "full_name")
edge_attribute(ison_algebra, "task_tie")
edge_weights(to_mode1(ison_southern_women))
edge_signs(ison_marvel_relationships)
graph_nodes(ison_southern_women)
graph_edges(ison_southern_women)
graph_dims(ison_southern_women)
graph_dims(to_mode1(ison_southern_women))
graph_node_attributes(mpn_elite_mex)
graph_edge_attributes(mpn_elite_mex)
```

---

graph_balance *Structural balance*

Description

Structural balance

Usage

```
graph_balance(object, method = "triangles")
```

Arguments

object	a migraph-consistent object
method	one of "triangles" (the default), "walk", or "frustration".

Value

"triangles" returns the proportion of balanced triangles, ranging between 0 if all triangles are imbalanced and 1 if all triangles are balanced.

Source

{signnet} by David Schoch

---

graph_census *Censuses for the whole graph*

Description

Censuses for the whole graph

Usage

```
graph_mixed_census(object, object2)

graph_dyad_census(object)

graph_triad_census(object)
```

Arguments

object	A migraph-consistent object.
object2	A second, two-mode migraph-consistent object.

Source

Alejandro Espinosa 'netmem'

References

Hollway, James, Alessandro Lomi, Francesca Pallotti, and Christoph Stadtfeld. "doi: 10.1017/nws.2017.8Multilevel Social Spaces: The Network Dynamics of Organizational Fields." *Network Science* 5, no. 2 (June 2017): 187–212.

Examples

```
marvel_friends <- to_unsigned(ison_marvel_relationships, "positive")
(mixed_cen <- graph_mixed_census(marvel_friends, ison_marvel_teams))
graph_dyad_census(ison_adolescents)
graph_triad_census(ison_adolescents)
```

graph_smallworld *Watts-Strogatz small-world model for two-mode networks*

Description

Calculates small-world metrics for one- and two-mode networks.

Usage

```
graph_smallworld(object, times = 100)
```

Arguments

object	A matrix, igraph graph, or tidygraph object
times	Number of simulations

Details

The first column of the returned table is simply the number of the second-mode column. The next three columns report the observed and expected clustering, and the ratio of the former to the latter. The next three columns report the observed and expected path-length, and the ratio of the former to the later. The last column reports the ratio of the observed/expected clustering ratio to the observed/expected path-length ratio, which is known as a small-world metric. Expected clustering and paths is the mean of twomode_clustering and mean_distance over 100 random simulations with the same row and column sums.

Value

Returns a table of small-world related metrics for each second-mode node.

References

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

See Also

graph_transitivity and graph_equivalency for how clustering is calculated

Other two-mode measures: cohesion(), node_constraint()

Examples

```
graph_smallworld(ison_southern_women)
graph_smallworld(ison_brandes)
```

`is` *Logical tests of network properties*

Description

These functions implement logical tests for various network properties.

Usage

```
is_migraph(object)

is_graph(object)

is_edgelist(object)

is_twomode(object)

is_weighted(object)

is_directed(object)

is_labelled(object)

is_signed(object)

is_connected(object, method = c("weak", "strong"))

is_complex(object)

is_multiplex(object)

is_uniplex(object)

is_acyclic(object)
```

Arguments

`object`	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
`method`	Whether to identify components if only "weak"ly connected or also "strong"ly connected.

Value

TRUE if the condition is met, or FALSE otherwise.

Functions

- `is_migraph`: Tests whether network is migraph-compatible
- `is_graph`: Tests whether network contains graph-level information
- `is_edgelist`: Tests whether data frame is an edgelist
- `is_twomode`: Tests whether network is a two-mode network
- `is_weighted`: Tests whether network is weighted
- `is_directed`: Tests whether network is directed
- `is_labelled`: Tests whether network includes names for the nodes
- `is_signed`: Tests whether network is signed positive/negative
- `is_connected`: Tests whether network is (weakly/strongly) connected
- `is_complex`: Tests whether network contains any loops
- `is_multiplex`: Tests whether network is multiplex, either from multiple rows with the same sender and receiver, or multiple columns to the edgelist.
- `is_uniplex`: Tests whether network is simple (both uniplex and simplex)
- `is_acyclic`: Tests whether network is a directed acyclic graph

See Also

Other manipulation: `add`, `coercion`, `to`

Examples

```
is_twomode(ison_southern_women)
is_weighted(ison_southern_women)
is_directed(ison_southern_women)
is_labelled(ison_southern_women)
is_signed(ison_southern_women)
is_connected(ison_southern_women)
is_complex(ison_southern_women)
is_uniplex(ison_algebra)
is_acyclic(ison_algebra)
```

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

Description

One-mode subset of the adolescent society (Coleman 1961)

Usage

```
data(ison_adolescents)
```

Format

A undirected one-mode tbl_graph object of 8 named nodes and 10 edges.

References

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

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

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

Description

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

Usage

```
data(ison_algebra)
```

Format

Multiplex `tbl_graph` object of friends, social, and task ties between 16 anonymous students.

Details

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

Source

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

References

McFarland, Daniel A. (2001) "Student Resistance." American Journal of Sociology, 107(3), p 612-678.

---

ison_brandes *One-mode centrality demonstration network*

Description

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

Usage

```
data(ison_brandes)
```

Format

A tidygraph `tbl_graph` with 11 nodes and 24 edges.

---

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

Description

One-mode karateka network (Zachary 1977)

Usage

```
data(ison_karateka)
```

Format

Undirected one-mode igraph with 34 named nodes and 78 edges.

Details

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

References

Zachary, Wayne W. 1977. “An Information Flow Model for Conflict and Fission in Small Groups.” Journal of Anthropological Research 33(4):452–73.

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

Description

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

Usage

```
data(ison_marvel_teams)

data(ison_marvel_relationships)
```

Format

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

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

Details

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

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

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

Source

Umut Yüksel, 31 March 2017

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

Description

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

Usage

```
data(ison_networkers)
```

Format

`tbl_graph` network object. The network is directed, simple, named , and weighted. It contains 32 nodes and 440 edges as well as two node level attributes: `Citations`; `Discipline`.

Source

networkdata package

References

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

Wasserman S. and K. Faust (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge.

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

Description

Two-mode projection examples (Hollway 2021)

Usage

```
data(ison_mm)

data(ison_bm)

data(ison_mb)

data(ison_bb)
```

Format

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

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

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

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

Details

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

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

Description

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

Usage

```
data(ison_southern_women)
```

Format

{igraph} two-mode graph object with 18 women and 14 informal social events.

References

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

`layouts` *Layouts for snapping layouts to a grid*

Description

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

Usage

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

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

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

Arguments

`object`	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
`circular`	Should the layout be transformed into a radial representation. Only possible for some layouts. Defaults to FALSE
`times`	Maximum number of iterations, where appropriate

References

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

Examples

```
autographr(mpn_elite_mex, "frgrid")
autographr(mpn_ryanair, "frgrid")
autographr(mpn_elite_mex, "kkgrid")
autographr(mpn_ryanair, "kkgrid")
autographr(mpn_elite_mex, "gogrid")
autographr(mpn_ryanair, "gogrid")
```

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

Description

A multimodal (3) network containing individuals affiliations to civic organizations in Bristol and their participation in major protest and civic events between 1990-2002, and the involvement of the organizations in these events.

Usage

```
data(mpn_bristol)
```

Format

A `tbl_graph` object with 264 rows and columns. Node IDs are prefaced with a type identifier:

1_ 150 Individuals, anonymised

2_ 97 Bristol Civic Organizations

3_ 17 Major Protest and Civic Events in Bristol, 1990-2002

The network is weighted, named, and directed.

Source

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

References

Diani, Mario, and Ivano Bison. 2004. "Organizations, Coalitions, and Movements." Theory and Society 33(3–4):281–309.

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

Description

This data contains the full network of 35 members of the Mexican power elite. The undirected lines connecting pairs of men represent any formal, informal, or organizational relation between a dyad; for example, "common belonging (school, sports, business, political participation), or a common interest (political power)" (Mendieta et al. 1997: 37). Additional nodal attributes include their full name, place of birth, state, and region (1=North, 2=Centre, 3=South, original coding added by Frank Heber), as well as their year of entry into politics and whether they are civilian (0) or affiliated with the military (1). An additional variable "in_mpn" can be used to subset this network to a network of 11 core members of the 1990s Mexican power elite (Knoke 2017), three of which were successively elected presidents of Mexico: José López Portillo (1976-82), Miguel de la Madrid (1982-88), and Carlos Salinas de Gortari (1988-94, who was also the son of another core member, Raúl Salinas Lozano).

Usage

```
data(mpn_elite_mex)
```

Format

`tbl_graph` network object. The network is simple, undirected, and named. The full network contains 35 nodes and 117 edges, and the subsetted network contains 11 nodes and 44 edges.

Source

Knoke, David. 1990. *Political Networks*.

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

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

Description

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

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

Usage

```
data(mpn_elite_usa_advice)

data(mpn_elite_usa_money)
```

Format

`mpn_elite_usa_advice` is a two-mode, named, and unweighted `tbl_graph` with 32 nodes and 46 edges.

`mpn_elite_usa_advice` is a two-mode, named, and unweighted `tbl_graph` with 38 nodes and 103 edges.

References

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

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

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

---

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

Description

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

Usage

```
data(mpn_IT_1990)

data(mpn_IT_2008)

data(mpn_DE_1990)

data(mpn_DE_2008)

data(mpn_UK_1990)

data(mpn_UK_2008)
```

Format

tbl_graph object based on an association matrix with 14 columns:

Welfare 1 if individual associated

Religious 1 if individual associated

Education.culture 1 if individual associated

Unions 1 if individual associated

Parties 1 if individual associated

Local.political.groups 1 if individual associated

Human.rights 1 if individual associated

Environmental.animal 1 if individual associated

Professional 1 if individual associated

Youth 1 if individual associated

Sports 1 if individual associated

Women 1 if individual associated

Peace 1 if individual associated

Health 1 if individual associated

An object of class tbl_graph (inherits from igraph) of length 10.

An object of class tbl_graph (inherits from igraph) of length 10.

An object of class tbl_graph (inherits from igraph) of length 10.

An object of class tbl_graph (inherits from igraph) of length 10.

An object of class tbl_graph (inherits from igraph) of length 10.

Source

Knoke, Diani, Hollway, and Christopoulos. 2021.

References

EVS (2020). European Values Study Longitudinal Data File 1981-2008 (EVS 1981-2008). GESIS Data Archive, Cologne. ZA4804 Data file Version 3.1.0, https://doi.org/10.4232/1.13486. *Multimodal Political Networks*. Cambridge University Press: Cambridge.

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

Description

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

Usage

```
data(mpn_ryanair)
```

Format

`tbl_graph` network object. The network is simple, directed, named and weighted. It contains 20 nodes and 177 edges.

Source

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

Knoke, Diani, Hollway, and Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press: Cambridge.

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

Description

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

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

Lastly, we obtained data on campaign contributions made by 92 PACs from the Open Secrets Website. We recorded all contributions made during the 2008, 2010, and 2012 election campaigns to

the 102 persons who were Senators in the 112th Congress. The vast majority of PAC contributions to a candidate during a campaign was for $10,000 (the legal maximum is $5,000 each for a primary and the general election). We aggregated the contributions across all three electoral cycles, then dichotomized the sums into no contribution (0) and any contribution (1).

Usage

```
data(mpn_DemSxP)

data(mpn_RepSxP)

data(mpn_OverSxP)
```

Format

`tbl_graph` network object. It is a bimodal, directed, named, weighted graph of 51 Senators (`type = FALSE`) and 63 PACS (`type = TRUE`) and 2791 edges.

`tbl_graph` network object. It is a bimodal, directed, named, weighted graph of 62 Senators (`type = FALSE`) and 72 PACS (`type = TRUE`) and 3675 edges.

`tbl_graph` network object. It is a bimodal, directed, named, weighted graph of 20 Senators (`type = FALSE`) and 32 PACS (`type = TRUE`) and 614 edges.

Source

Knoke, Diani, Hollway, and Christopoulos. 2021. *Multimodal Political Networks*. Cambridge University Press: Cambridge.

---

node_constraint *Constraint for one- and two-mode networks*

Description

This function measures constraint for both one-mode and two-mode networks. For one-mode networks, the function wraps the implementation of Ron Burt's measure in `{igraph}`. For two-mode networks, the function employs the extension outlined in Hollway et al. (2020).

Usage

```
node_constraint(object, nodes = V(object), weights = NULL)
```

Arguments

`object`	A matrix, igraph graph, or tidygraph object.
`nodes`	The vertices for which the constraint will be calculated. Defaults to all vertices.
`weights`	The weights of the edges. If this is NULL and there is a weight edge attribute this is used. If there is no such edge attribute all edges will have the same weight.

Value

A named vector (one-mode) or a list of two named vectors ($nodes1, $nodes2).

References

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.

See Also

Other one-mode measures: `cohesion()`

Other two-mode measures: `cohesion()`, `graph_smallworld()`

Examples

```
node_constraint(ison_southern_women)
```

---

read *Reading from/writing to external formats*

Description

Researchers regularly need to work with a variety of external data formats. The following functions offer ways to import from some common external file formats into objects that `{migraph}` and other graph/network packages in R can work with. Note that these functions are not as actively maintained as others in the package, so please let us know if any are not currently working for you by raising an issue on Github.

Usage

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

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

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

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

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

write_pajek(object, filename, ...)

read_ucinet(file = file.choose())

write_ucinet(object, filename, name)
```

Arguments

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

Details

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

- UCINET data
- Pajek data

See also:

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

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

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

Value

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

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

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

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

Functions

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

Source

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

See Also

coercion

Examples

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

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

## End(Not run)
```

`regression` *Linear and logistic regression for network data*

Description

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

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

Usage

```
network_reg(
  formula,
  object,
  method = c("qap", "qapy"),
  times = 1000,
  strategy = "sequential",
  verbose = FALSE
)
```

Arguments

`formula` A formula describing the relationship being tested. Several additional terms are available to assist users investigate the effects they are interested in. These include:

- `ego()` constructs a matrix where the cells reflect the value of a named nodal attribute for an edge's sending node
- `alter()` constructs a matrix where the cells reflect the value of a named nodal attribute for an edge's receiving node
- `same()` constructs a matrix where a 1 reflects if two nodes' attribute values are the same
- `dist()` constructs a matrix where the cells reflect the absolute difference between the attribute's values for the sending and receiving nodes

- `sim()` constructs a matrix where the cells reflect the proportional similarity between the attribute's values for the sending and receiving nodes
- dyadic covariates (other networks) can just be named

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

- matrix, from base R
- edgelist, a data frame from base R or tibble from tibble
- igraph, from the igraph package
- network, from the network package
- tbl_graph, from the tidygraph package

`method` A method for establishing the null hypothesis. Note that "qap" uses Dekker et al's (2007) double semi-partialling technique, whereas "qapy" permutes only the y variable. "qap" is the default.

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

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

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

References

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

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

See Also

```
vignette("p7linearmodel")
```

Examples

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

`tests` *Conditional uniform graph and permutation tests*

Description

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

Usage

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

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

Arguments

`object`	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
`FUN`	A graph-level statistic function to test.
`...`	Additional arguments to be passed on to FUN, e.g. the name of the attribute.
`times`	Integer indicating the number of draws to use for quantile estimation. (Relevant to the null hypothesis test only - the analysis itself is unaffected by this parameter.) Note that, as for all Monte Carlo procedures, convergence is slower for more extreme quantiles. By default, times=1000. 1,000 - 10,000 repetitions recommended for publication-ready results.

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

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

Examples

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

---

to *Tools for reformatting networks, graphs, and matrices*

Description

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

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

to_	edgelists	matrices	igraph	tidygraph	network
unweighted	X	X	X	X	X
undirected		X	X	X	X
unsigned	X	X	X	X	
uniplex			X	X	
unnamed	X	X	X	X	X
named	X	X	X	X	X
simplex		X	X	X	
main_component			X	X	X
onemode			X	X	
multilevel		X	X	X	
mode1		X	X	X	
mode2		X	X	X	

Usage

```
to_uniplex(object, edge)

to_main_component(object)

to_undirected(object)

to_unweighted(object, threshold = 1)

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

to_unnamed(object)

to_named(object, names = NULL)

to_simplex(object)

to_mode1(object)

to_mode2(object)

to_onemode(object)

to_multilevel(object)

to_edges(object)

to_subgraph(object, ...)
```

Arguments

object	An object of a migraph-consistent class: • matrix, from base R • edgelist, a data frame from base R or tibble from tibble • igraph, from the igraph package • network, from the network package • tbl_graph, from the tidygraph package
edge	Character string naming an edge attribute to retain from a graph.
threshold	For a matrix, the threshold to binarise/dichotomise at.
keep	In the case of a signed network, whether to retain the "positive" or "negative" ties.
names	Character vector of the node names. NULL by default.
...	Arguments passed on to dplyr::filter

Value

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

Functions

- to_uniplex: Returns an object that includes only a single type of tie
- to_main_component: Returns an object that includes only the main component without any smaller components or isolates
- to_undirected: Returns an object that has any edge direction removed, so that any pair of nodes with at least one directed edge will be connected by an undirected edge in the new network. This is equivalent to the "collapse" mode in {igraph}.
- to_unweighted: Returns an object that has all edge weights removed
- to_unsigned: Returns a network with either just the "positive" ties or just the "negative" ties
- to_unnamed: Returns an object with all vertex names removed
- to_named: Returns an object that has random vertex names added
- to_simplex: Returns an object that has all loops or self-ties removed
- to_mode1: Results in a weighted one-mode object that retains the row nodes from a two-mode object, and weights the ties between them on the basis of their joint ties to nodes in the second mode (columns)
- to_mode2: Results in a weighted one-mode object that retains the column nodes from a two-mode object, and weights the ties between them on the basis of their joint ties to nodes in the first mode (rows).
- to_onemode: Returns an object that has any type/mode attributes removed, but otherwise includes all the same nodes and ties. Note that this is not the same as to_mode1() or to_mode2(), which return only some of the nodes and new ties established by coincidence.
- to_multilevel: Returns a network that is not divided into two mode types but embeds two or more modes into a multimodal network structure.
- to_edges: Returns a matrix (named if possible) where the edges are the nodes
- to_subgraph: Returns a network subgraph filtered on the basis of some node-related logical statement.

See Also

Other manipulation: add, coercion, is()

Examples

```
autographr(ison_algebra)
a <- to_uniplex(ison_algebra, "friend_tie")
autographr(a)
a <- to_main_component(a)
autographr(a)
a <- to_undirected(a)
```

```
autographr(a)
a <- to_unweighted(a)
autographr(a)
a <- to_named(a)
autographr(a)
autographr(to_unsigned(ison_marvel_relationships, "positive")) /
autographr(to_unsigned(ison_marvel_relationships, "negative"))
autographr(ison_southern_women) /
(autographr(to_mode1(ison_southern_women)) |
autographr(to_mode2(ison_southern_women)))
autographr(ison_adolescents) +
autographr(to_edges(ison_adolescents))
```

Index

∗ **creation**
create, 18
generate, 20
∗ **datasets**
ison_adolescents, 30
ison_algebra, 30
ison_brandes, 31
ison_karateka, 31
ison_marvel, 32
ison_networkers, 33
ison_projection, 34
ison_southern_women, 34
mpn_bristol, 36
mpn_elite_mex, 36
mpn_elite_usa, 37
mpn_evs, 38
mpn_ryanair, 40
mpn_senate112, 40
∗ **manipulation**
add, 3
coercion, 13
is, 28
to, 48
∗ **measures**
centrality, 9
∗ **node-level measures**
node_constraint, 41
∗ **one-mode measures**
cohesion, 14
node_constraint, 41
∗ **three-mode measures**
cohesion, 14
∗ **two-mode measures**
cohesion, 14
graph_smallworld, 27
node_constraint, 41

add, 3, *14*, *29*, *50*
add_edge_attributes *(*add*)*, 3
add_node_attributes *(*add*)*, 3
as_edgelist *(*coercion*)*, 13
as_igraph *(*coercion*)*, 13
as_matrix *(*coercion*)*, 13
as_network *(*coercion*)*, 13
as_tidygraph *(*coercion*)*, 13
autographr, 4

blockmodel, 5
blockmodel_concor *(*blockmodel*)*, 5
blockmodel_vis, 7

census, 8
centrality, 9
cluster, 12
cluster_regular_equivalence *(*cluster*)*, 12
cluster_structural_equivalence *(*cluster*)*, 12
coercion, *3*, 13, *29*, *44*, *50*
cohesion, 14, *27*, *42*
connectedness, 16
copy_node_attributes *(*add*)*, 3
create, 18, *21*
create_complete *(*create*)*, 18
create_components *(*create*)*, 18
create_empty *(*create*)*, 18
create_lattice *(*create*)*, 18
create_ring *(*create*)*, 18
create_star *(*create*)*, 18
create_tree *(*create*)*, 18

diversity, 19

edge_attribute *(*grab*)*, 24
edge_betweenness *(*centrality*)*, 9
edge_bridges *(*connectedness*)*, 16
edge_closeness *(*centrality*)*, 9
edge_degree *(*centrality*)*, 9
edge_eigenvector *(*centrality*)*, 9
edge_loop *(*cohesion*)*, 14

edge_multiple (cohesion), 14
edge_reciprocal (cohesion), 14
edge_signs (grab), 24
edge_weights (grab), 24

generate, *19*, 20
generate_permutation (generate), 20
generate_random (generate), 20
generate_scalefree (generate), 20
generate_smallworld (generate), 20
ggatyear, 22
ggevolution, 22
ggidentify_clusters (blockmodel_vis), 7
gglineage, 23
ggtree (blockmodel_vis), 7
grab, 24
graph_adhesion (connectedness), 16
graph_assortativity (diversity), 19
graph_balance, 25
graph_betweenness (centrality), 9
graph_blau_index (diversity), 19
graph_census, 26
graph_closeness (centrality), 9
graph_cohesion (connectedness), 16
graph_components (connectedness), 16
graph_congruency (cohesion), 14
graph_degree (centrality), 9
graph_density (cohesion), 14
graph_diameter (connectedness), 16
graph_dims (grab), 24
graph_dyad_census (graph_census), 26
graph_edge_attributes (grab), 24
graph_edges (grab), 24
graph_ei_index (diversity), 19
graph_eigenvector (centrality), 9
graph_equivalency, *27*
graph_equivalency (cohesion), 14
graph_length (connectedness), 16
graph_mixed_census (graph_census), 26
graph_node_attributes (grab), 24
graph_nodes (grab), 24
graph_reciprocity (cohesion), 14
graph_smallworld, *16*, 27, *42*
graph_transitivity, *27*
graph_transitivity (cohesion), 14
graph_triad_census (graph_census), 26
group_tie_census (census), 8
group_triad_census (census), 8

is, *3*, *14*, 28, *50*
is_acyclic (is), 28
is_complex (is), 28
is_connected (is), 28
is_directed (is), 28
is_edgelist (is), 28
is_graph (is), 28
is_labelled (is), 28
is_migraph (is), 28
is_multiplex (is), 28
is_signed (is), 28
is_twomode (is), 28
is_uniplex (is), 28
is_weighted (is), 28
ison_adolescents, 30
ison_algebra, 30
ison_bb (ison_projection), 34
ison_bm (ison_projection), 34
ison_brandes, 31
ison_karateka, 31
ison_marvel, 32
ison_marvel_relationships
        (ison_marvel), 32
ison_marvel_teams (ison_marvel), 32
ison_mb (ison_projection), 34
ison_mm (ison_projection), 34
ison_networkers, 33
ison_projection, 34
ison_southern_women, 34

join_edges (add), 3

layout_tbl_graph_frgrid (layouts), 35
layout_tbl_graph_gogrid (layouts), 35
layout_tbl_graph_kkgrid (layouts), 35
layouts, 35

mpn_bristol, 36
mpn_DE_1990 (mpn_evs), 38
mpn_DE_2008 (mpn_evs), 38
mpn_DemSxP (mpn_senate112), 40
mpn_elite_mex, 36
mpn_elite_usa, 37
mpn_elite_usa_advice (mpn_elite_usa), 37
mpn_elite_usa_money (mpn_elite_usa), 37
mpn_evs, 38
mpn_IT_1990 (mpn_evs), 38
mpn_IT_2008 (mpn_evs), 38
mpn_OverSxP (mpn_senate112), 40

mpn_RepSxP (mpn_senate112), 40
mpn_ryanair, 40
mpn_senate112, 40
mpn_UK_1990 (mpn_evs), 38
mpn_UK_2008 (mpn_evs), 38

network_reg (regression), 45
node_attribute (grab), 24
node_betweenness (centrality), 9
node_closeness (centrality), 9
node_components (connectedness), 16
node_constraint, *16*, *27*, 41
node_cuts (connectedness), 16
node_degree (centrality), 9
node_eigenvector (centrality), 9
node_mode (grab), 24
node_names (grab), 24
node_quad_census (census), 8
node_tie_census (census), 8
node_triad_census (census), 8

plot.block_model (blockmodel_vis), 7
print.block_model (blockmodel), 5

read, 42
read_edgelist (read), 42
read_nodelist (read), 42
read_pajek (read), 42
read_ucinet (read), 42
reduce_graph (blockmodel), 5
regression, 45

summarise_statistics (blockmodel), 5

test_permutation (tests), 47
test_random (tests), 47
tests, 47
to, *3*, *14*, *29*, 48
to_edges (to), 48
to_main_component (to), 48
to_mode1 (to), 48
to_mode2 (to), 48
to_multilevel (to), 48
to_named (to), 48
to_onemode (to), 48
to_simplex (to), 48
to_subgraph (to), 48
to_undirected (to), 48
to_undirected(), *9*, *11*
to_uniplex (to), 48
to_unnamed (to), 48
to_unsigned (to), 48
to_unweighted (to), 48
to_unweighted(), *11*

write_edgelist (read), 42
write_nodelist (read), 42
write_pajek (read), 42
write_ucinet (read), 42