Package ‘migraph’

October 30, 2021

Title Multimodal and Multilevel Network Analysis

Version 0.8.5

Date 2021-10-29

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 dplyr, ggdendro, ggraph, ggplot2, gridExtra, igraph, magrittr, network, purrr, RColorBrewer, readxl, rlang, sna, stringr, tibble, tidygraph, tidyr, concaveman, ggforce

Suggests knitr, testthat, rmarkdown, roxygen2, covr

VignetteBuilder knitr

NeedsCompilation no

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

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


Repository CRAN
Date/Publication 2021-10-30 09:50:06 UTC

R topics documented:

autographr ................................................................. 3
blockmodel ................................................................. 4
blockmodel_vis ............................................................ 5
brandes ....................................................................... 6
census ....................................................................... 6
centrality ................................................................. 7
centralization ............................................................ 9
cluster ...................................................................... 10
coection ................................................................. 11
cohesion ................................................................. 12
create ................................................................. 13
generate ............................................................... 15
ggatyear ................................................................. 16
ggevolution ............................................................. 16
gglineage ................................................................. 17
ggraphgrid ............................................................... 18
ggttools ................................................................. 18

graph_balance .......................................................... 19
graph_census ............................................................ 20
graph_components ....................................................... 20
is ................................................................. 21
ison_coleman ............................................................... 22
ison_community ........................................................... 22
ison_marvel ............................................................... 23
ison_projection ............................................................ 24
layouts ................................................................. 24
m182 ................................................................. 25
mpn_bristol ............................................................... 26
mpn_elite_mex ............................................................. 26
mpn_elite_usa_advice .................................................... 27
mpn_elite_usa_money ..................................................... 27
mpn_evs ................................................................. 28
mpn_ryanair ............................................................... 29
mpn_senate112 ............................................................ 30
netlm ................................................................. 31
node_components .......................................................... 32
node_constraint ........................................................... 32
node_smallworld .......................................................... 33
project ................................................................. 34
read ................................................................. 34
southern_women ............................................................. 36
to ................................................................. 36
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.

Usage

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

Arguments

- **object**: migraph-consistent object
- **algorithm**: an igraph layout algorithm, currently defaults to 'stress' but Fruchterman-Reingold and Kamada-Kawai also available
- **labels**: logical, whether to print node names as labels if present
- **node_size**: an override in case this needs to be manually set
- **node_color**: node variable in quotation marks that should be used for colouring the nodes
- **node_group**: node variable in quotation marks that should 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.
- **...**: extra arguments

Examples

```
autographr(ison_coleman)
autographr(ison_karateka)
```
**blockmodel**

---

**Description**

Blockmodelling

**Usage**

```r
blockmodel(object, clusters)

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

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

reduce_graph(blockmodel, block_labels = NULL)
```

**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 "blockmodel"
- ... 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

**Source**

[https://github.com/aslez/concoR](https://github.com/aslez/concoR)
References


Examples

```r
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)
```

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

```r
## S3 method for class 'blockmodel'
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

```r
usa_concor <- blockmodel_concor(mpn_elite_usa_advice)
plot(usa_concor)
res <- cluster_regular_equivalence(mpn_elite_mex)
ggtree(res, 4)
ggidentify_clusters(res, node_triad_census(mpn_elite_mex))
```

---

### brandes

**One-mode centrality demonstration structure**

**Description**

One-mode centrality demonstration structure

**Usage**

```r
data(brandes)
```

**Format**

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

---

### 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**

```r
node_tie_census(object)

node_triad_census(object)

group_tie_census(object, clusters, decimals = 2)

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

**Arguments**

- `object`: a migraph-consistent object
- `clusters`: a vector of cluster assignment
- `decimals`: number of decimal points to round to
Examples

```r
task_eg <- to_named(to_uniplex(ison_m182, "task_tie"))
tie_cen <- node_tie_census(task_eg)
(triad_cen <- node_triad_census(task_eg))
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 for one- and two-mode networks*

**Description**

These functions calculate common centrality measures for both one- and two-mode networks. They accept as objects matrices and igraph graphs, and can be used within a tidygraph workflow. Importantly, these functions also offer correct normalization for two-mode networks.

**Usage**

```r
node_degree(
  object,
  weights = NULL,
  mode = "out",
  loops = TRUE,
  normalized = FALSE
)
```

```r
node_closeness(
  object,
  weights = NULL,
  mode = "out",
  normalized = FALSE,
  cutoff = NULL
)
```

```r
node_betweenness(
  object,
  weights = NULL,
  directed = TRUE,
  cutoff = NULL,
  nobigint = TRUE,
  normalized = FALSE
)
```

```r
node_eigenvector(
  object,
  weights = NULL,
  directed = FALSE,
```
options = igraph::arpack_defaults,
  scale = FALSE,
  normalized = FALSE
)

Arguments

  object    Either an igraph graph object or a matrix.
  weights   The weight of the edges to use for the calculation. Will be evaluated in the
            context of the edge data.
  mode      How should edges be followed. Ignored for undirected graphs
  loops     Should loops be included in the calculation
  normalized For one-mode networks, should Borgatti and Everett normalization be applied?
  cutoff    maximum path length to use during calculations
  directed  Should direction of edges be used for the calculations
  nobigint  Should big integers be avoided during calculations
  options   Settings passed on to igraph::arpack()
  scale     Should the scores be scaled to range between 0 and 1?

Value

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.

References


See Also

  Other two-mode measures: centralization, cohesion, node_constraint(), node_smallworld()
  Other node-level measures: node_constraint(), node_smallworld()

Examples

  node_degree(mpn_elite_mex)
  node_degree(southern_women)
  node_closeness(mpn_elite_mex)
  node_closeness(southern_women)
  node_betweenness(mpn_elite_mex)
  node_betweenness(southern_women)
  node_eigenvector(mpn_elite_mex)
  node_eigenvector(southern_women)
centralization

Centralization for one- and two-mode networks

Description

These functions measure the overall centralization for a network.

Usage

```r
graph_degree(
  object,
  directed = c("all", "out", "in", "total"),
  normalized = TRUE,
  digits = 2
)
```

```r
graph_closeness(
  object,
  directed = c("all", "out", "in", "total"),
  normalized = TRUE,
  digits = 2
)
```

```r
graph_betweenness(
  object,
  directed = c("all", "out", "in", "total"),
  normalized = TRUE,
  digits = 2
)
```

```r
graph_eigenvector(object, digits = 2)
```

Arguments

- **object**: A matrix, igraph graph, or tidygraph object.
- **directed**: Character string, "out" for out-degree, "in" for in-degree, and "all" or "total" for 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 nodeset.
- **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.
- **digits**: whether to round the resulting score, by default 2. Add FALSE to turn all rounding off.
Value

A single centralization score if the object was one-mode, and two centralization scores if the object was two-mode. In the case of a two-mode network, to return just the score for the first nodeset (rows), append $nodes1 to the end of the function call or returned object. To return just the score for the second nodeset (cols), append $nodes2 to the end of the function call or returned object.

References


See Also

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

Examples

graph_degree(southern_women, directed = "in")
graph_closeness(southern_women, directed = "in")
graph_betweenness(southern_women, directed = "in")
graph_eigenvector(mpn_elite_mex)

corresponding 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

ggplot(cluster_structural_equivalence(mpn_elite_mex))
ggplot(cluster_regular_equivalence(mpn_elite_mex))
Coercion between graph/network object classes

Description

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

- adjacency and incidence matrices
- edgelists (as data frames)
- `{igraph}` graph objects
- `{tidygraph}` tbl_graph objects
- `{network}` network objects

Usage

```r
as_edgelist(object, weight = FALSE)
as_matrix(object, weight = FALSE)
as_igraph(object, weight = FALSE, twomode = FALSE)
as_tidygraph(object, twomode = FALSE)
as_network(object)
```

Arguments

- `object` A data frame edgelist, matrix, igraph, tidygraph, or network object.
- `weight` An option to override the heuristics for distinguishing weighted networks. By default FALSE.
- `twomode` An option to override the heuristics for distinguishing incidence from adjacency matrices. By default FALSE.

Details

Behaviour is a little different depending on the data format.

If the data frame is a 2 column edgelist, the first column will become the rows and the second column will become the columns. If the data frame is a 3 column edgelist, then the third column will be used as the cell values or tie weights.

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:

<table>
<thead>
<tr>
<th>to/from</th>
<th>edgelists</th>
<th>matrices</th>
<th>igraph</th>
<th>tidygraph</th>
<th>network</th>
</tr>
</thead>
<tbody>
<tr>
<td>edgelists</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>matrices</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>igraph</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>tidygraph</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>network</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

See Also

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

Examples

```r
test <- data.frame(id1 = c("A","B","B","C","C"),
                   id2 = c("I","G","I","G","H"))

as_matrix(test)
as_igraph(test)
as_tidygraph(test)
as_network(test)
```

doctor 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

```r
graph_density(object)

graph_reciprocity(object, method = "default")

graph_transitivity(object)

graph_equivalency(object)

graph_congruency(object, object2)
```

Arguments

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

For one- 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.

References


See Also

Other one-mode measures: `node_constraint()`

Other two-mode measures: `centrality, centralization, node_constraint(), node_smallworld()`

Examples

```r
graph_density(mpn_elite_mex)
graph_density(mpn_elite_usa_advice)
graph_reciprocity(southern_women)
graph_transitivity(southern_women)
graph_equivalency(southern_women)
```

---

**create**  
*Create networks with particular structures*

**Description**

These functions create a variety of different network objects. Despite the common function names and syntax with existing packages, the common `n` argument can not only be passed a single integer to return a one-mode network, but also 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, directed = c("undirected", "in", "out"))
create_tree(n, directed = c("undirected", "in", "out"), branches = 2)
create_lattice(n, directed = c("undirected", "in", "out"))

Arguments

n Number of nodes. If a single integer is given, e.g. \( n = 10 \), the function will create a one-mode network. If a vector of two integers is given, e.g. \( n = c(5, 10) \), the function will create a two-mode network.

width The width or breadth of the ring. This is typically double the degree.
directed One of the following options: "in", "out", or "undirected" (DEFAULT).
...
Additional arguments passed on to igraph.
components Number of components to create.
branches How many branches at each level

details

create_empty() creates an empty graph of the given dimensions.
create_complete() creates a filled graph of the given dimensions.
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

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().

See Also

as_matrix as_tidygraph as_network

Other creation: generate
Examples

```r
g <- create_empty(c(8,6))
autographr(g)
g <- create_complete(c(8,6))
autographr(g)
g <- create_ring(c(8,6), width = 2)
autographr(g)
autographr(create_components(c(10, 12), components = 3))
autographr(create_star(c(12,1), "in"))
tr1 <- autographr(create_tree(12))
tr2 <- autographr(create_tree(12), "tree")
grid.arrange(tr1, tr2, ncol = 2)
cl1 <- autographr(create_lattice(5))
cl2 <- autographr(create_lattice(c(5,5)))
cl3 <- autographr(create_lattice(c(5,5,5)))
grid.arrange(cl1, cl2, cl3, ncol = 3)
```

---

**generate**

*Create networks from particular probabilities*

**Description**

Create networks from particular probabilities

**Usage**

```r
generate_random(n, p, m)
generate_smallworld(n, p = 0.05)
generate_scalefree(n, p = 1)
```

**Arguments**

- `n`: Integer of length 1 or 2.
- `p`: Number of edges in the network over the number of edges possible
- `m`: Number of edges in the network

**Details**

Creates a random network. If `length(n)==1`, then a one-mode network will be returned, equivalent to an Erdös-Rényi graph. If `length(n)==1`, then a two-mode network will be returned. The first number is the number of nodes in the first nodeset (rows), and the second number becomes the number of nodes in the second nodeset (columns).

**See Also**

Other creation: `create`
Examples

```r
er1 <- autographr(generate_random(12, 0.4))
er2 <- autographr(generate_random(c(6, 6), 0.4))
grid.arrange(er1, er2, ncol = 2)
sw1 <- autographr(generate_smallworld(12, 0.025))
sw2 <- autographr(generate_smallworld(12, 0.25))
grid.arrange(sw1, sw2, ncol = 2)
sf1 <- autographr(generate_scalefree(12, 0.25))
sf2 <- autographr(generate_scalefree(12, 1.25))
grid.arrange(sf1, sf2, ncol = 2)
```

---

**ggatyear**

*Plotting network at particular timepoint (year)*

**Description**

Plotting network at particular timepoint (year)

**Usage**

```r
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

**Examples**

```r
## 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**

```r
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_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")

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

ggligneage(object, labels = TRUE)

Arguments

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

Examples

glineage(cites)
**ggraphgrid**

*Plot graph to grid*

**Description**

For quick and easy graphing of networks to a grid plot.

**Usage**

```
  ggraphgrid(x, algorithm = c("kk", "fr"))
```

**Arguments**

- `x`: A migraph-consistent network/graph
- `algorithm`: An initial network layout, currently either Kamada-Kawai ("kk") or Fruchterman-Reingold ("fr")

**Details**

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

**References**


---

**ggtools**

Visualising graphs and identifying nodes with maximum values of the specified measure.

**Description**

Visualising graphs and identifying nodes with maximum values of the specified measure.

**Usage**

```
  ggidentify(object, node_measure, identify_function = max)
  ggdistrib(object, node_measure)
```
**Arguments**

- **object**: a migraph-consistent object
- **node_measure**: some arbitrary function that runs on the object and returns a numeric vector that can be used to scale the nodes
- **identify_function**: a function for the identification of a single node, e.g. max, min, mean, etc.

**Examples**

```r
ggidentify(brandes, node_degree)
ggidentify(brandes, node_betweenness)
ggidentify(brandes, node_closeness)
ggidentify(brandes, node_eigenvector)
gdistrib(brandes, node_degree)
gdistrib(brandes, node_betweenness)
gdistrib(brandes, node_closeness)
gdistrib(brandes, node_eigenvector)
```

---

**Description**

Structural balance

**Usage**

```r
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_components**

### Description

Censuses for the whole graph

### Usage

```r
graph_dyad_census(object)
graph_triad_census(object)
```

### Arguments

- **object**: a migraph-consistent object

### Examples

```r
graph_dyad_census(ison_coleman)
graph_triad_census(ison_coleman)
```

---

**graph_components**

### Description

Number of components in the network

### Usage

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

### Arguments

- **object**: a migraph-consistent object
- **method**: For directed networks, either weak if edge direction is irrelevant, or strong if edge direction is salient. Ignored if network undirected.
Description

Tests of network properties

Usage

is_twomode(object)

is_weighted(object)

is_directed(object)

is_labelled(object)

is_signed(object)

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

Arguments

object A migraph-consistent class object (matrix, edgelist, igraph, network, tidygraph)

method Whether to identify components if only "weak"ly connected or also "strong"ly connected.

Value

TRUE if object is a two-mode network, otherwise FALSE
TRUE if object is a weighted network, otherwise FALSE
TRUE if object is a directed network, otherwise FALSE
TRUE if object is a labelled network, otherwise FALSE
TRUE if object is a signed network, otherwise FALSE
TRUE if object is a connected network, otherwise FALSE

See Also

Other manipulation: coercion, project, to
Examples

```r
is_twomode(southern_women)
is_weighted(southern_women)
is_directed(southern_women)
is_labelled(southern_women)
is_signed(southern_women)
is_connected(southern_women)
```

ison_coleman  
One-mode subset of adolescent society dataset

Description

One-mode subset of adolescent society dataset

Usage

```r
data(ison_coleman)
```

Format

tidygraph graph object

References


ison_community  
Zachary's kareteka network

Description

Zachary's kareteka network

Usage

```r
data(ison_karateka)
```

Format

Undirected one-mode igraph with 34 nodes and 78 edges
Multilevel two-mode affiliation, signed one-mode networks of Marvel comic book characters

Description
Multilevel two-mode affiliation, signed one-mode networks of Marvel comic book characters

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
### layouts

#### ison_projection

**Two-mode projection examples**

**Description**

Two-mode projection examples

**Usage**

- `data(ison_mm)`
- `data(ison_bm)`
- `data(ison_mb)`
- `data(ison_bb)`

**Format**

- Directed two-mode igraph with 6 nodes and 6 edges
- Directed two-mode igraph with 8 nodes and 9 edges
- Directed two-mode igraph with 8 nodes and 9 edges
- Directed two-mode igraph with 10 nodes and 12 edges

---

#### layouts

**Layouts for one- and two-mode networks**

**Description**

Layouts for one- and two-mode networks

**Usage**

- `layout_tbl_graph_frgrid(object, circular = FALSE, maxiter = 1000)`
- `layout_tbl_graph_kkgrid(object, circular = FALSE, maxiter = 1000)`
- `layout_tbl_graph_gogrid(object, circular = FALSE, maxiter = 1000)`

**Arguments**

- **object**: A migraph-consistent network/graph
- **circular**: Should the layout be transformed into a radial representation. Only possible for some layouts. Defaults to FALSE
- **maxiter**: maximum number of iterations, where appropriate
Details

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

References


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")

m182: Multiplex igraph of friends, social, and task ties between 16 anonymous students

Description

M182 was an honors algebra class and friendship, social, and task ties were collected/observed.

Usage

data(ison_m182)

Format

Multiplex tidygraph of friends, social, and task ties between 16 anonymous 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.
**mpn_bristol**  
*Multimodal (3) Bristol protest events, 1990-2002*

**Description**

A multimodal (3) matrix 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**

```r
data(mpn_bristol)
```

**Format**

A matrix 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

**Source**


---

**mpn_elite_mex**  
*One-mode Mexican power elite database*

**Description**

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). 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).

**Usage**

```r
data(mpn_elite_mex)
```

**Format**

Matrix with 11 rows/columns
Source


---

**mpn_elite_usa_advice**  Two-mode American power elite database

**Description**

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.

**Usage**

```r
data(mpn_elite_usa_advice)
```

**Format**

Matrix with 14 rows and 20 columns

**References**


---

**mpn_elite_usa_money**  Three-mode American power elite database

**Description**

This data 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**

```r
data(mpn_elite_usa_money)
```
**Format**

Matrix with 26 rows and 6+6 columns

**References**


---

**mpn_evs**

*Two-mode European Values Survey, 1990 and 2008*

**Description**


**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**

Matrices 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

Usage
data(mpn_ryanair)

Format
Matrix with 20 rows/columns

Source
Two-mode 112th Congress Senate Voting

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 we analyzed is 102.

Usage

data(mpn_DemSxP)
data(mpn_RepSxP)
data(mpn_OverSxP)

Format

Matrix of 51 rows (Senators) and 63 columns (PACS)
Matrix of 62 rows (Senators) and 72 columns (PACS)
Matrix of 20 rows (Senators) and 32 columns (PACS)

Details

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).

Source

Description

This function extends the multiple regression quadratic assignment procedure (MRQAP) of network linear model to two mode networks.

Usage

netlm(formula, data, ...)

## S3 method for class 'netlm'
summary(object, reps = 1000, ...)

## S3 method for class 'summary.netlm'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  signif.stars = getOption("show.signif.stars"),
  ...
)

Arguments

formula A formula describing the relationship being tested.
data A named list of matrices, graphs, or a tidygraph object.
... Arguments passed on to lm().
object an object of class "netlm", usually as a result of a call to netlm().
reps 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, reps=1000.
x an object of class "summary.netlm", usually, a result of a call to summary.netlm().
digits the number of significant digits to use when printing.
signif.stars logical. If TRUE, ‘significance stars’ are printed for each coefficient.

Examples

mat1 <- matrix(c(0,1,0,0,1,1,1),4,2)
mat2 <- matrix(c(0,1,0,1,0,1,0),4,2)
mat3 <- matrix(c(0,0,1,1,0,0,1),4,2)
lmat <- list(mat1 = mat1, mat2 = mat2, mat3 = mat3)
model1 <- netlm(mat1 ~ mat2 + mat3, lmat)
summary(model1)
### node_components

**Description**

Identifying nodes’ component membership

**Usage**

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

**Arguments**

- **object**: A migraph-consistent object
- **method**: For directed networks, either `weak` if edge direction is irrelevant, or `strong` if edge direction is salient. Ignored if network undirected.

### node_constraint

**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**

See Also

Other one-mode measures: cohesion
Other two-mode measures: centrality, centralization, cohesion, node_smallworld()
Other node-level measures: centrality, node_smallworld()

Examples

node_constraint(southern_women)

node_smallworld(southern_women)

Description

Calculates small-world metrics for two-mode networks

Usage

node_smallworld(object, niter = 100)

Arguments

object A matrix, igraph graph, or tidygraph object
niter 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 latter. 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 two_mode_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.

See Also

graph_transitivity and graph_equivalency for how clustering is calculated
Other two-mode measures: centrality, centralization, cohesion, node_constraint()
Other node-level measures: centrality, node_constraint()

Examples

node_smallworld(southern_women)
Projecting two-mode objects into one-mode objects

Description

These functions 'project' or convert a two-mode object in any format – tidygraph, igraph, or matrix – into a corresponding one-mode object.

Usage

project_rows(object)

project_cols(object)

Arguments

object A matrix, igraph graph or tidygraph tbl_graph object.

Details

project_rows() 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).

project_cols() 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).

See Also

Other manipulation: coercion, is(), to

Examples

project_rows(southern_women)
project_cols(southern_women)

Description

These functions import from and export to UCINET network files.
Usage

read_edgelist(file)

read_ucinet(header_file)

write_ucinet(
    object,
    filename = deparse(substitute(object)),
    name = deparse(substitute(object))
)

Arguments

file, header_file  A character string giving the path to the header (.##h) file. If the function is called without a header_file specified, an OS-specific file picker is opened to help users select it.

object  A migraph-consistent object to be exported.

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

These 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

By default, read_ucinet() and read_edgelist() will import into an igraph format, but can be easily coerced from there into other formats.

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

Author(s)

Christian Steglich, 18 June 2015

See Also

convert
## Examples

```r
# 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)
```

---

### southern_women

*Two-mode southern women dataset*

---

**Description**

Two-mode network dataset collected by Davis, Gardner and Gardner (1941) about women and social events.

**Usage**

```r
data(southern_women)
```

**Format**

igraph graph object

**References**


---

**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).
Usage

to_unweighted(object, threshold = 1)
to_unnamed(object)
to_undirected(object)
to_onemode(object)
to_main_component(object)
to_uniplex(object, edge)
to_unsigned(object, keep = c("positive", "negative"))
to_simplex(object)
to_named(object)
to_multilevel(object)

Arguments

object A matrix, \{igraph\} graph, \{tidygraph\} tbl_graph, or \{network\} object.

threshold For a matrix, the threshold to binarise/dichotomise at.

edge the name of an edge attribute to retain from a graph

keep in the case of a signed network, whether to retain the "positive" or "negative" ties

Details

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

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

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

- to_unweighted() returns an object that has all edge weights removed
- to_unnamed() returns an object that has all vertex names removed
- to_named() returns an object that has random vertex names added
- to_undirected() returns an object that has any edge direction removed
- to_onemode() returns an object that has any type/mode attributes removed, but otherwise includes all the same nodes and ties. Note that this is not the same as project_rows() or project_cols(), which return only some of the nodes and new ties established by coincidence.
- to_main_component() returns an object that includes only the main component and not any smaller components or isolates
- to_uniplex() returns an object that includes only a single type of tie
- to_simplex() returns an object that has all loops or self-ties removed
- to_unsigned() returns an object that has

See Also

Other manipulation: coercion, is(), project

Examples

to_unweighted(project_rows(southern_women))
to_unnamed(project_rows(southern_women))
to_undirected(ison_coleman)
to_onemode(ison_marvel_teams)
to_uniplex(ison_m182, "friend_tie")
to_unsigned(ison_marvel_relations, "positive")
to_unsigned(ison_marvel_relations, "negative")
to_simplex(ison_m182)
to_named(ison_m182)
to_multilevel(mpn_elite_usa_advice)
Index

* creation
  create, 13
  generate, 15
* datasets
  brandes, 6
  ison_coleman, 22
  ison_community, 22
  ison_marvel, 23
  ison_projection, 24
  m182, 25
  mpn_bristol, 26
  mpn_elite_mex, 26
  mpn_elite_usa_advice, 27
  mpn_elite_usa_money, 27
  mpn_evs, 28
  mpn_ryanair, 29
  mpn_senate112, 30
  southern_women, 36
* graph-level measures
  centralization, 9
* manipulation
  coercion, 11
  is, 21
  project, 34
  to, 36
* node-level measures
  centrality, 7
  node_constraint, 32
  node_smallworld, 33
* one-mode measures
  cohesion, 12
  node_constraint, 32
* three-mode measures
  cohesion, 12
* two-mode measures
  centrality, 7
  centralization, 9
  cohesion, 12
  node_constraint, 32

node_smallworld, 33
as_edgelist (coercion), 11
as_igraph (coercion), 11
as_matrix (coercion), 11
as_network (coercion), 11
as_tidygraph (coercion), 11
autographr, 3
blockmodel, 4
blockmodel_concor (blockmodel), 4
blockmodel_vis, 5
brandes, 6
census, 6
centrality, 7, 10, 13, 33
centralization, 8, 9, 13, 33
cluster, 10
cluster_regular_equivalence (cluster), 10
cluster_structural_equivalence (cluster), 10
coercion, 11, 21, 34, 38
cohesion, 8, 10, 12, 33
convert, 35
create, 13, 15
create_complete (create), 13
create_components (create), 13
create_empty (create), 13
create_lattice (create), 13
create_ring (create), 13
create_star (create), 13
create_tree (create), 13
generate, 14, 15
generate_random (generate), 15
generate_scalefree (generate), 15
generate_smallworld (generate), 15
ggayear, 16
ggdistrib (ggtools), 18
INDEX

```{r}
ggevolution, 16
ggidentify (ggtools), 18
ggidentify_clusters (blockmodel_vis), 5
ggraphkit, 17
ggraphgrid, 18
ggtree (blockmodel_vis), 5
graph_balance, 19
graph_betweenness (centralization), 9
graph_census, 20
graph_closeness (centralization), 9
graph_components, 20
graph_congruency (cohesion), 12
graph_degree (centralization), 9
graph_density (cohesion), 12
graph_dyad_census (graph_census), 20
graph_eigenvector (centralization), 9
graph_equivalency, 33
graph_equivalency (cohesion), 12
graph_reciprocity (cohesion), 12
graph_transitivity, 33
graph_transitivity (cohesion), 12
graph_triad_census (graph_census), 20
group_tie_census (census), 6
group_triad_census (census), 6
```

```{r}
layout_tbl_graph_kkgrid (layouts), 24
layout_tbl_graph_kggrid (layouts), 24
layout_tbl_graph_kggrid (layouts), 24
layout_tbl_graph_kxgrid (layouts), 24
layout_tbl_graph_mggrid (layouts), 24
layout_tbl_graph_nxgrid (layouts), 24
layout_tbl_graph_pggrid (layouts), 24
layout_tbl_graph_qxgrid (layouts), 24
layout_tbl_graph_rkgrid (layouts), 24
layout_tbl_graph_scgrid (layouts), 24
layout_tbl_graph_sggrid (layouts), 24
layout_tbl_graph_spgrid (layouts), 24
layout_tbl_graph_sqgrid (layouts), 24
layout_tbl_graph_sxgrid (layouts), 24
layout_tbl_graph_wkgrid (layouts), 24
layout_tbl_graph_wkgrid (layouts), 24
```
INDEX

*to_main_component*(to), 36
*to_multilevel*(to), 36
*to_named*(to), 36
*to_onemode*(to), 36
*to_simplex*(to), 36
*to_undirected*(to), 36
*to_uniplex*(to), 36
*to_unnamed*(to), 36
*to_unsigned*(to), 36
*to_unweighted*(to), 36

*write_ucinet*(read), 34