

Package ‘ElectroGraph’

March 28, 2012

Type Package

Title Enhanced routines for plotting and analyzing valued relational data.

Version 0.9.3

Date 2012-03-26

Author Andrew C. Thomas

Maintainer Andrew C. Thomas <act@acthomas.ca>

Description This package contains routines and procedures for relational data, namely considerations for valued ties. In particular, relative distances may also be calculated using Ohmic methods.

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2012-03-28 21:23:54

R topics documented:

ElectroGraph-package	2
alphabet.class	3
animate.plot.series	3
betweenness.centralities	4
clustering.statistics.socio	5
dist.coords	6
electro.frats	7
electrograph	7
geodesic.mat	10
get.resistance	11
make.sociomatrix.from.edges	12
make.sociomatrix.from.lattice	13
network.projection	14

ohmic.properties	15
pair.sequence	16
plot.electrograph	17
sociomatrix.to.nby3.edges	19
summary.electrograph	19
wedding.cake.plot	21

Index	23
--------------	-----------

ElectroGraph-package *The ElectroGraph package: Analyzing and visualizing relational data*

Description

ElectroGraph takes as input an edge list or sociomatrix, either valued or binary, and computes various statistics for node and edge importance.

Details

Package:	ElectroGraph
Type:	Package
Version:	0.1
Date:	2009-05-15
License:	GPL, v2
LazyLoad:	yes

To create an ElectroGraph object, specify an edge list or sociomatrix.

Author(s)

Andrew C. Thomas

Maintainer: Andrew C. Thomas <act@acthomas.ca>

References

Andrew C. Thomas (2009) “Hierarchical Models for Relational Data”. Unpublished PhD thesis.

Examples

```
latt <- cbind(rep(1:5,5),sort(rep(1:5,5)))
latt.e <- electrograph (make.sociomatrix.from.lattice(latt)$sociomatrix)
plot(latt.e)
```

alphabet.class	<i>ElectroGraph Demo Data Set</i>
----------------	-----------------------------------

Description

A fictitious data set used in the ElectroGraph manual/article to demonstrate plots and tie fidelities.

Usage

```
alphabet.class
```

Format

A 16-by-3 data frame containing sources, sinks and fidelities for a series of ties.

animate.plot.series	<i>animate.plot.series</i>
---------------------	----------------------------

Description

Create a series of plots that animate the transitions between subsequent states of a relational data system.

Usage

```
animate.plot.series(plots.to.animate, intermediates=10, filebase="electro-animate",
                    plot.width=600, plot.height=600, verbose= 1, ...)
```

Arguments

plots.to.animate	A list of objects of class "electrograph.plot", the output from a plot(electrograph) command.
intermediates	The number of frames to intersperse between source plots.
filebase	The name of the directory into which the output files will be saved.
plot.width, plot.height	The height and width in pixels of the output PNG files.
verbose	Output additional (diagnostic) information.
...	Additional arguments to pass to "plot".

Value

Creates a series of plots in the PNG format in a directory named for "filebase". Also produces a shell script to produce an animated GIF of the sequence using the program ImageMagick. Function outputs the list of files produced in the animation.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
#uses data "newcomb" in this set.
data(electro.frats)
newcomb.electro <- list(NA)
for (kk in 1:dim(newcomb)[3]) {
  newcomb.electro[[kk]] <- electrograph(newcomb[, ,kk])
}

newcomb.plot <- list(NA)
newcomb.plot[[1]] <- plot(newcomb.electro[[1]],connectivity.mode="ohmic", just=TRUE)
for (kk in 2:length(newcomb.electro))
  newcomb.plot[[kk]] <- plot(newcomb.electro[[kk]],connectivity.mode="ohmic",
just=TRUE, previous.elec=newcomb.plot[[kk-1]])

animate.plot.series(newcomb.plot)
```

betweenness.centralities

Calculating Geodesic Betweenness Properties of a Network

Description

Functions that

Usage

```
single.source.betweenness (edgelist, sourcepoint=1, destpoint=NULL, node.ids=NULL)
betweenness.centralities (edgelist,
  path.weight=c("constant","closeness"), verbose=TRUE, node.ids=NULL)
```

```
recourse.betweenness.one (edgelist, sourcepoint=1, destpoint=NULL, penalty=20, node.ids=NULL)
recourse.betweenness.source (edgelist, sourcepoint=1, penalty=20, node.ids=NULL)
recourse.betweenness.full (edgelist, penalty=20, path.weight=c("constant","closeness"), node.ids=NULL)
```

Arguments

edgelist	A k-by-2 or k-by-3 matrix containing edge information. The first two columns represent the nodes in the arc; if included, the third refers to the strength of the tie (inverse distance).
sourcepoint	The number of the source node.
destpoint	The number of the destination node.

verbose	Display additional output if TRUE.
penalty	The additional cost to be considered in Recourse Betweenness if the route is blocked.
path.weight	Should the betweennesses of edges be calculated with all source-sink pairs equal, or weighted inversely by their geodesic distance?
node.ids	Labels for the nodes in the system.

Value

Objects output the betweenness measures for the edges in the system, given the sources and destinations as inputs.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
#Ring lattice.
edgelist <- cbind(c(1:5, 2:5, 1), c(2:5,1, 1:5))

geodesic.selected.from.edgelist (edgelist)
single.source.betweenness (edgelist)
betweenness.centralities (edgelist, path.weight="constant")

recourse.betweenness.full (edgelist)
```

```
clustering.statistics.socio
      clustering.statistics.socio
```

Description

Calculate the transitive counts and cycles in a sociomatrix.

Usage

```
clustering.statistics.socio(sociomatrix, verbose = 0)
```

Arguments

sociomatrix	An n-by-n sociomatrix.
verbose	Display additional information.

Value

A data frame containing the fraction of possible cycles and transitive triples for each node.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
# generate a 5-by-5 grid of points.
pts <- cbind(rep(1:5,5),sort(rep(1:5,5)))
socio <- make.sociomatrix.from.lattice(pts)
clustering.statistics.socio(socio$sociomatrix)
```

dist.coords

dist.coords

Description

Solve for the Euclidean distance matrix between two sets of points.

Usage

```
dist.coords(coords, coords2=NULL)
```

Arguments

coords, coords2

Coordinate matrices of dimension n1-by-k (and optionally n2-by-k).

Value

If one argument is entered, returns an n1-by-n1 matrix with the Euclidean distance between all pairs of points. If two, the matrix is the distances between all pairs of points across the two matrices.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
coords1 <- array(rnorm(20), c(10,2))
coords2 <- array(rnorm(20), c(10,2))

dist.coords(coords1)
dist.coords(coords1, coords2)
```

`electro.frats`*Two Fraternity Data Sets*

Description

Contains the time series from the Newcomb fraternity study, as well as the observational set from the Bernard-Killworth human interaction data sets.

Usage`electro.frats`**Format**

`newcomb`: an array, 17-by-17-by-15. Each of the 15 array slices represents a time point; each row represents the social preferences of each individual for the others at the time. Originally rank choices of 1-16, these are now fractions $(17 - (1-16))/17$ such that higher preferences have higher connection.

`bernard.killworth.b`: a 1934-by-3 array. Columns 1 and 2 ("P1","P2") represent the dyad under observation; column 3 ("count") is the number of times these individuals were seen to interact over the observational period.

Source

Originally obtained from the UCINET IV data source; included in R format for convenience and demonstration.

<http://vlado.fmf.uni-lj.si/pub/networks/data/Ucinet/UciData.htm>

References

Bernard H, Killworth P and Sailer L. (1980). Informant accuracy in social network data IV. *Social Networks*, 2, 191-218. Newcomb T. (1961). *The acquaintance process*. New York: Holt, Reinhard & Winston.

`electrograph`*electrograph object manipulation*

Description

Given a sociomatrix or an edge list, produce an electrograph object for display and analysis. Additional functions augment an electrograph object with new analyses.

Usage

```

electrograph(input, make.edgelist.symmetric=TRUE,
             shortest.paths.get=FALSE,
             betweenness.get=FALSE,
             recourse.betweenness.get=FALSE,
             ohmic.properties.get=FALSE,
             fidelities=NULL,
             verbose=FALSE, substitute.names=NULL,
             missing.numbers.imply.isolates=FALSE, ...)
add.shortest.paths(e.graph)
add.betweenness(e.graph)
add.recourse.betweenness(e.graph)

```

Arguments

<code>input</code>	A sociomatrix (n-by-n) or edge list (k-by-(2,3,4)).
<code>make.edgelist.symmetric</code>	Whether the input edge list should be taken as symmetric.
<code>shortest.paths.get</code>	If TRUE, calculates the geodesic distances between nodes.
<code>betweenness.get</code>	If TRUE, calculates the standard geodesic betweenness centrality for all edges.
<code>recourse.betweenness.get</code>	If TRUE, calculates the recourse betweenness centrality for all edges.
<code>ohmic.properties.get</code>	If TRUE, calculates the Ohmic properties of the network nodes and ties through <code>electrograph.exam</code> . This will be lengthy if the system has a large number of nodes.
<code>fidelities</code>	If set, allows for ties that are perceived to be less than “friends”. (Use with caution – results have not been peer reviewed!)
<code>verbose</code>	Additional stage-by-stage feedback is given if TRUE.
<code>substitute.names</code>	A k-by-2 matrix with one row containing original node designations and the other containing replacement names
<code>missing.numbers.imply.isolates</code>	If TRUE, interprets source-sink labels in the edge list as integer node IDs and adds the appropriate number of isolates.
<code>...</code>	Additional options to be passed to <code>ohmic.properties</code> .
<code>e.graph</code>	An electrograph object to be augmented with further analysis.

Details

The input object can have one of four forms: -A square sociomatrix. -An n-by-2 matrix containing social arcs; the individuals in column 1 consider those in column 2 to be friends with social strength 1. By default, this is symmetric. -An n-by-3 matrix containing social arcs; the individuals in column 1 consider those in column 2 to be friends with social strength as described in column 3. By

default, this is symmetric. -An n-by-4 matrix containing dyadic arcs. The connection (col1,col2) is described in column 3; the reciprocal connection (col2,col1) is described in column 4. This by nature overrides the option “symmetric” to be false unless the input is truly symmetric.

Value

A list, of class “electrograph” containing the following items:

<code>grand.sociomatrix</code>	The full sociomatrix of the system.
<code>grand.sociomatrices</code>	A list of sociomatrices of the components of the system.
<code>component.vector</code>	The component in which each node is located.
<code>geodesic</code>	The full geodesic path length matrix of the system.
<code>diameters</code>	The geodesic diameters of each component.
<code>global.pseudo.diameter</code>	A sum of all diameters in the set plus the psuedo-diameter bridges required to putatively connect all components.
<code>symmetric</code>	Is the resulting sociomatrix symmetric?

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Source

Andrew C. Thomas (2010). “Ohmic Circuit Interpretations of Network Distance and Centrality”. Unpublished: available at <http://www.acthomas.ca/>. Andrew C. Thomas and Stephen E. Fienberg (2011). “Exploring the Consequences of IED Deployment with a Generalized Linear Model Implementation of the Canadian Traveller Problem”. <http://arxiv.org/abs/1012.4185>

Examples

```
sources <- c(1,2,3,4)
sinks <- c(2,3,1,5)
socio <- electrograph(cbind(sources,sinks))
```

geodesic.mat *geodesic.mat*

Description

Given a k-by-2 or k-by-3 edge list, calculate the minimum geodesic distance between any two points using Dijkstra's Algorithm.

Usage

```
geodesic.matrix.from.edgelist(edgelist, node.ids=NULL)
geodesic.selected.from.edgelist(edgelist, sourcepoint=1, destpoint=NULL, node.ids=NULL)
```

Arguments

edgelist	A k-by-2 or k-by-3 matrix containing edge information. The first two columns represent the nodes in the arc; if included, the third refers to the strength of the tie (inverse distance).
sourcepoint	The number of the source node.
destpoint	The farthest node whose distance from the source should be calculated.
node.ids	Labels for the nodes in the system.

Value

geodesic.matrix.from.edgelist outputs an n-by-n matrix; cell (i,j) contains the shortest geodesic distance from point i to point j.

geodesic.selected.from.edgelist outputs a vector of distances from the source node, accurate as far as the distance for "destpoint".

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
latt <- cbind(rep(1:5,5),sort(rep(1:5,5)))
lattice.edges <- sociomatrix.to.nby3.edges(make.sociomatrix.from.lattice(latt)$sociomatrix)
latt.geo <- geodesic.matrix.from.edgelist(lattice.edges)

#Distances from the corner node.
corner.dist <- geodesic.selected.from.edgelist(lattice.edges, 1)
```

<code>get.resistance</code>	<i>Calculating Ohmic properties of a network</i>
-----------------------------	--

Description

Given a pair of nodes to act as source and sink, determine the resulting current flow.

Usage

```
compute.volts.symmetric.edges (edgelist, src, snk, fidelities = NULL)
compute.volts.asymmetric.edges (edgelist, src, snk, fidelities = NULL)
compute.volts.symmetric.socio (sociomatrix, src, snk, fidelities = NULL)
compute.volts.asymmetric.socio (sociomatrix, src, snk, fidelities = NULL)
get.resistance (e.graph, sources, sinks, verbose=FALSE)
```

Arguments

<code>sociomatrix</code>	An n-by-n sociomatrix.
<code>edgelist</code>	A k-by-{2,3,4} edge list.
<code>e.graph</code>	An electrograph object.
<code>src, snk</code>	Single values for the source and sink.
<code>sources, sinks</code>	Vectors of source and sink values.
<code>fidelities</code>	If set, allows for ties that are perceived to be less than “friends”. (Use with caution – results have not been peer reviewed!)
<code>verbose</code>	Enable additional (diagnostic) output.

Value

Returns a list containing the following items:

<code>resist.eq</code>	The equivalent resistance between the source and sink.
<code>voltages</code>	Each column contains the electric potential at each node given a source/sink pair.
<code>current.out</code>	The total current flowing into and out of each node given a source/sink pair.
<code>reference.mean</code>	As a check, computed mean of the sociomatrix.

Note

The “compute” functions are subsumed within `get.resistance()` and are given for diagnostic purposes. In general, the full execution of “`get.resistance`” is more reliable and faster; this is called in general form by “`electrograph.exam`”.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
latt <- cbind(rep(1:5,5),sort(rep(1:5,5)))
latt.mat <- make.sociomatrix.from.lattice(latt)$sociomatrix

print(compute.volts.symmetric.socio(latt.mat,1,25))
latt.mat[24,25] <- 0
print(compute.volts.asymmetric.socio(latt.mat,1,25))
print(compute.volts.asymmetric.socio(latt.mat,25,1))

latt.e <- electrograph (latt.mat)
print(get.resistance(latt.e,c(1,2,3),c(23,24,25)))
```

```
make.sociomatrix.from.edges
      make.sociomatrix.from.edges
```

Description

Given an edge list, with or without values, produce the corresponding sociomatrix.

Usage

```
make.sociomatrix.from.edges(inputmat, size=NULL, symmetric=FALSE, fidelities=NULL)
```

Arguments

<code>inputmat</code>	A matrix with dimension n-by-(2,3,4); see below for details.
<code>size</code>	If non-null, specifies the number of nodes that should be in the resulting sociomatrix and implies that existing node labels are integers from 1 to size.
<code>symmetric</code>	If TRUE, creates a symmetric sociomatrix with undirected edges.
<code>fidelities</code>	If set, allows for ties that are perceived to be less than “friends”. (Use with caution – results have not been peer reviewed!)

Details

The first two columns of the matrix are the source and target of each directed edge (undirected if symmetry flag is set to TRUE). If `inputmat` has 3 columns, the third column represents the value of each tie. If `inputmat` has 4 columns, the third and fourth column represent the edge weights in each direction, as in (source->target), (target->source).

Value

A sociomatrix, where the number of rows and columns is equal to the count of unique nodes specified; row and column names are equal to the node names given.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
src <- 1:10
tgt <- 2:11
socio <- make.sociomatrix.from.edges(cbind(src,tgt), symmetric=TRUE)

values <- rgamma(10,3,3)
socio.value <- make.sociomatrix.from.edges(cbind(src,tgt,values), symmetric=TRUE)

back.values <- rgamma(10,3,12)
socio.value <- make.sociomatrix.from.edges(cbind(src,tgt,values,back.values))
```

```
make.sociomatrix.from.lattice
      make.sociomatrix.from.lattice
```

Description

Given a set of points in two dimensions, produce a sociomatrix where a link exists if their distance is exactly 1.

Usage

```
make.sociomatrix.from.lattice(pts.nby2)
```

Arguments

pts.nby2 An n-by-2 matrix whose rows are (x,y) coordinates of points.

Value

A sociomatrix whose entries correspond to indicators of whether the points are exactly 1 unit apart.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
# generate a 5-by-5 grid of points.
pts <- cbind(rep(1:5,5),sort(rep(1:5,5)))
socio <- make.sociomatrix.from.lattice(pts)$sociomatrix
```

network.projection *network.projection*

Description

Given a table of ideal distances and a connectivity matrix, calculate the projected coordinates under force-directed placement.

Usage

```
network.projection(connectivity,
                   force.mode=c("fruchterman.reingold", "kamada.kawai"),
                   layout.dimension=2,
                   ego.focus=NULL,
                   initial.coordinates = NULL,
                   verbose=1,
                   projection.iterations=5000)
```

Arguments

connectivity	An n-by-n matrix containing the values of the connective strengths among the n individuals (inverse distance).
force.mode	Either of the above two standard methods for force-directed placement.
layout.dimension	Of 1, 2 or 3, the dimension of the output coordinates.
ego.focus	A vector of node indices for which priority should be made highest for their relative distances.
initial.coordinates	If specified, the algorithm will begin at the specified coordinates rather than a randomly generated set.
verbose	If TRUE, display additional information while running.
projection.iterations	The maximum number of cycles that the projection algorithm will run.

Value

An n-by-(layout.dimension) table of coordinates.

Note

The algorithm can technically handle any dimensionality in the data, and is written for expandability. However, in its current form, as only two-dimensional plotting is supported in the package (and three-dimensional visualization the maximum supported by the human brain) the layout dimension is restricted in this routine.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
latt <- cbind(rep(1:5,5),sort(rep(1:5,5)))
latt.e <- electrograph (make.sociomatrix.from.lattice(latt)$sociomatrix,
shortest.paths=TRUE)
latt.coords <- network.projection(1/latt.e$geodesic)

print(latt.coords)
```

ohmic.properties	<i>ohmic.properties</i>
------------------	-------------------------

Description

Compute a full analysis of Ohmic properties of the system.

Usage

```
ohmic.properties(e.graph, sample.edges=FALSE, sample.per=NULL, verbose=FALSE)
```

Arguments

e.graph	An object initialized by the electrograph() function.
sample.edges	If TRUE, samples of edges are taken as opposed to the full set.
sample.per	Sets the number of samples taken if sample.edges is TRUE. Defaults to the square root of the number of edges.
verbose	Enable additional (diagnostic) output.

Value

The original electrograph object, augmented with Ohmic-social quantities:

ohmic.distance.mat	The equivalent electro-social resistances for each pair of nodes in matrix form; if sample.edges is TRUE, edge importance is approximated.
conductances	A vector of Ohmic-social conductances for each specified pair of nodes.
voltages	A matrix with each column representing the voltages at each node.
currents.node	A matrix with each column representing the current through each node.
source.sink	The source-sink vectors used in the examination.
current.matrix	The average current through each directed edge.

Note

This is an option when the constructor “electrograph” is called, but can be executed manually as well.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
latt <- cbind(rep(1:5,5),sort(rep(1:5,5)))
latt.e <- electrograph (make.sociomatrix.from.lattice(latt)$sociomatrix, ohmic=FALSE)
latt.exam <- ohmic.properties(latt.e)
```

`pair.sequence`

pair.sequence

Description

Given an input integer, produces a sequence of all n-choose-2 pairs in the data.

Usage

```
pair.sequence(nn)
```

Arguments

nn Maximum number of nodes.

Value

An (n-choose-2)-by-2 matrix whose rows take values (1,2), (1,3), ... (n-1,n).

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
pair.sequence(5)
```

plot.electrograph *plot.electrograph*

Description

Given an electrograph object, display a two-dimensional representation.

Usage

```
## S3 method for class 'electrograph'
plot(x, connectivity.mode = c("visible.ties", "shortest.path", "ohmic", "ohmic.on.ties"),
     force.mode = c("fruchterman.reingold", "kamada.kawai"),
     ego.focus = NULL, manual.coords = NULL, redo.coordinates = FALSE,
     just.coordinates = FALSE, node.colors = NULL,
     label.colors = NULL, node.cex = 1.5, label.cex = 1.5,
     component.border.col = 8,
     edge.thickness = c("standard",
                       "geodesic.betweenness", "ohmic.betweenness"),
     max.thick = 2, source.sink.pair = NULL,
     previous.electrograph.plot.object = NULL,
     tick.marks=FALSE,
     bound.size=1,
     width.height.factor=1,
     edge.colors.specified = NULL,
                               edge.color.fr.ne.em = c("black", "blue", "red"),
     verbose = 0,
     ...)
```

Arguments

<code>x</code>	An object of class "electrograph".
<code>connectivity.mode</code>	Should the projected distances be calculated using the shortest path, or the equivalent conductance of the electro-social approach?
<code>force.mode</code>	Which force-directed placement method should be used to find node coordinates?
<code>ego.focus</code>	A vector of nodes whose importance should be considered highest in the plot.
<code>manual.coords</code>	If desired, the manual input of coordinates for each node.
<code>redo.coordinates</code>	If true, recalculate the true distances between points using the desired distance method.
<code>just.coordinates</code>	If true, do not output a plot.
<code>node.colors</code>	The colors of the nodes to be plotted.
<code>label.colors</code>	The colors of node labels.

<code>node.cex</code>	Point sizes.
<code>label.cex</code>	Label sizes.
<code>component.border.col</code>	The color assigned to the boxes separating graph components.
<code>edge.thickness</code>	Determines whether the sociomatrix or electro-social betweenness will determine the thickness of edges in the plot.
<code>max.thick</code>	The maximum thickness of an edge in plotting.
<code>source.sink.pair</code>	A vector of length 2 that specifies an electro-social current path to be highlighted.
<code>previous.electrograph.plot.object</code>	A previously produced graph object, for the sake of lining up a new plot with a previous one.
<code>tick.marks</code>	If TRUE, place tick marks at equal distances on each axis.
<code>bound.size</code>	Denotes the degree extra space to surround each component in the plot.
<code>width.height.factor</code>	The desired ratio of width to height in the resulting plot.
<code>verbose</code>	If TRUE, display additional information while running.
<code>edge.colors.specified</code>	Set this to an (nodes)-by-(nodes) matrix to have each tie be the desired RGB color.
<code>edge.color.fr.ne.em</code>	A three-item vector indicating the colors of “friend”, “neutral”, or “enemy” ties when fidelities are specified.
<code>...</code>	Additional options to pass to “plot”.

Value

Returns an object of class “electrograph.plot” which can be replotted or used in later analysis.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
sources <- c(1,2,3,4)
sinks <- c(2,3,1,5)
socio <- electrograph(cbind(sources,sinks))

plot(socio)
```

```
sociomatrix.to.nby3.edges
      sociomatrix.to.nby3.edges
```

Description

Given a sociomatrix, produce an edge/arc list whose columns are the source, destination and value of each arc.

Usage

```
sociomatrix.to.nby3.edges(sociomatrix, keep.labels=FALSE)
```

Arguments

sociomatrix	An n-by-n matrix whose elements are the strengths of arcs corresponding to the row and column indices.
keep.labels	If true, maintains the row names as the node labels; if false, assigns node IDs as 1:nrow(sociomatrix).

Value

A k-by-3 matrix, whose columns are the source, destination and value of each of k arcs.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
latt <- cbind(rep(1:5,5),sort(rep(1:5,5)))
lattice.socio <- make.sociomatrix.from.lattice(latt)$sociomatrix
lattice.edges <- sociomatrix.to.nby3.edges(lattice.socio)
```

```
summary.electrograph  summary.electrograph
```

Description

Given an electrograph object, summarize the qualities of its nodes.

Usage

```
## S3 method for class 'electrograph'
summary(object, verbose = 0, ...)
## S3 method for class 'electrograph'
print(x, max.count = NULL, ...)
```

Arguments

x, object	An object of class "electrograph".
verbose	Display additional information.
max.count	The maximum number of components to be displayed in output.
...	(no further options currently available)

Value

"print" returns an abbreviated list of contents of the object. "summary" Returns a list containing several measures:

in.deg, out.deg	The sums of the inbound and outbound weights of each tie from each node.
cent.geodesic	Geodesic centrality: the sums of the inverses of the minimum distances to each other node in the system.
cent.electro	Electro-social centrality: the sums of the effective conductances from a node to each other node in the system. Defined only if electrograph.exam has been run on the object.
current.cent	Electro-betweenness measure: the average current through each node, given each node pair for which the effective conductance was measured.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
sources <- c(1,2,3,4)
sinks <- c(2,3,1,5)
socio <- electrograph(cbind(sources,sinks))

print(socio)
summary(socio)
```

wedding.cake.plot *wedding.cake.plot*

Description

Given an electrograph object for a valued relational data set, create a series of plots that illustrate the graph at various threshold levels for tie existence.

Usage

```
wedding.cake.plot(x, connectivity.mode="visible.ties", force.mode="fruchterman.reingold",
                 ego.focus=NULL, filebase="electrograph-cake", #type="png",
                 lower.bound=NULL, upper.bound=NULL, main.title=TRUE,
                 plot.width = 600, plot.height = 600, verbose = 1, ...)
```

Arguments

<code>x</code>	An object of class “electrograph”.
<code>connectivity.mode</code>	Should the projected distances be calculated using the shortest path, or the equivalent conductance of the electro-social approach?
<code>force.mode</code>	Which force-directed placement method should be used to find node coordinates?
<code>ego.focus</code>	A vector of nodes whose importance should be considered highest in the plot.
<code>filebase</code>	An identifier for the series of images produced in the approach.
<code>lower.bound</code> , <code>upper.bound</code>	The minimum and maximum values for an edge to be plotted.
<code>main.title</code>	If true, print the bound values in the title of the plot.
<code>plot.width</code> , <code>plot.height</code>	The height and width in pixels of the output PNG files.
<code>verbose</code>	Additional output provided.
<code>...</code>	Additional options to pass to “plot”.

Value

Creates a series of plots in the PNG format corresponding to the choices of bounds in a directory named for “filebase”.

Author(s)

Andrew C. Thomas <act@acthomas.ca>

Examples

```
#A sample Erdos-Renyi type graph for wedding caking.  
pts <- pair.sequence(10)  
edges <- exp(rnorm(dim(pts)[1],0,1))  
socio <- electrograph(cbind(pts,edges))  
  
wedding.cake.plot(socio)
```

Index

*Topic **datasets**

alphabet.class, 3
electro.frats, 7

*Topic **package**

ElectroGraph-package, 2

add.betweenness (electrograph), 7

add.recourse.betweenness
(electrograph), 7

add.shortest.paths (electrograph), 7

alphabet.class, 3

alphabetclass (alphabet.class), 3

animate.plot.series, 3

bernard.killworth.b (electro.frats), 7

betweenness.centralities, 4

clustering.statistics.socio, 5

compute.volts.asymmetric.edges
(get.resistance), 11

compute.volts.asymmetric.socio
(get.resistance), 11

compute.volts.symmetric.edges
(get.resistance), 11

compute.volts.symmetric.socio
(get.resistance), 11

dist.coords, 6

electro.frats, 7

ElectroGraph (ElectroGraph-package), 2

electrograph, 7

ElectroGraph-package, 2

geodesic.mat, 10

geodesic.matrix.from.edgelist
(geodesic.mat), 10

geodesic.selected.from.edgelist
(geodesic.mat), 10

get.resistance, 11

make.sociomatrix.from.edges, 12

make.sociomatrix.from.lattice, 13

network.projection, 14

newcomb (electro.frats), 7

ohmic.properties, 15

pair.sequence, 16

plot.electrograph, 17

print.electrograph
(summary.electrograph), 19

recourse.betweenness.full
(betweenness.centralities), 4

recourse.betweenness.one
(betweenness.centralities), 4

recourse.betweenness.source
(betweenness.centralities), 4

single.source.betweenness
(betweenness.centralities), 4

sociomatrix.to.nby3.edges, 19

summary.electrograph, 19

wedding.cake.plot, 21