

Package ‘PairViz’

April 17, 2009

Version 1.0

Author C.B. Hurley and R.W. Oldford

Maintainer Catherine Hurley <catherine.hurley@nuim.ie>

Title Visualization using Eulerian tours and Hamiltonian decompositions

Description Eulerian tours and Hamiltonian decompositions of complete graphs are used to ameliorate order effects in statistical graphics.

Depends TSP, gtools, graph, methods

Suggests

License GPL-2

Repository CRAN

Date/Publication 2008-10-28 20:07:25

R topics documented:

best_orientation	2
cancer	3
eseq	4
etour	5
eulerian	6
even_graph	7
find_path	8
guided_pcp	9
hpaths	11
mc_plot	13
mk_complete_graph	14
mk_even_graph	15
order_best	16
order_tsp	17
overlayCI	18

overview	19
path_cor	20
path_weights	20
pv_pcp	22
table_plot	23
weighted_hpaths	24

Index	26
--------------	-----------

best_orientation *Re-oriens a path to be weight-decreasing*

Description

Re-oriens a path/cycle, preserving adjacencies so that weights tend to decrease. From specifies the starting point, for cycles only.

Usage

```
best_orientation(path, d, cycle=FALSE, path_dir= path_cor, from=NULL)
```

Arguments

path	A vector giving a hamiltonian.
d	A dist, used to provide edge weights.
cycle	If TRUE, the path is interpreted as a closed path.
path_dir	A function used to evaluate a path start and orientation
from	Specifies the starting point, for cycles only.

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

[hpaths](#), [eulerian](#).

Examples

```
require(PairViz)

rdist <- function(n) {
  d <- matrix(0,n,n)
  d[lower.tri(d)] <- runif(n*(n-1)/2)
  return(as.dist(d))
}
r <- rdist(7)
best_orientation(1:7,r)
best_orientation(1:7,r,cycle=TRUE)
```

cancer

Cancer Survival data

Description

Patients with advanced cancers of the stomach, bronchus, colon, ovary or breast were treated with ascorbate. The purpose of the study was to determine if the survival times differ with respect to the organ affected by the cancer.

Usage

```
data(cancer)
```

Format

This data frame contains the following columns:

Survival time in days

Organ Organ affected by the cancer

Source

Data obtained from <http://lib.stat.cmu.edu/DASL/Stories/CancerSurvival.html>

References

Cameron, E. and Pauling, L. (1978) Supplemental ascorbate in the supportive treatment of cancer: re-evaluation of prolongation of survival times in terminal human cancer. Proceedings of the National Academy of Science USA, 75, 4538-4542.

Also found in: Manly, B.F.J. (1986) Multivariate Statistical Methods: A Primer, New York: Chapman and Hall, 11. Also found in: Hand, D.J., et al. (1994) A Handbook of Small Data Sets, London: Chapman and Hall, 255.

`eseq`*Construct eulerian paths on the complete graph where nodes are integers 1..n.*

Description

Constructs an eulerian on the complete graph where nodes are integers 1..n. The result in an euler tour for odd n. For even n the result is not exactly an euler tour or path because $(n-2)/2$ edges must be visited twice.

Usage

```
eseq(n)
eseqa(n)
kntour_drop(e)
kntour_add(e)
```

Arguments

n	a positive integer.
e	an euler tour on K_n where n is odd.

Details

The algorithm used for `eseq` builds up an euler tour on 1..n by appending extra edges on to the tour on 1..(n-2).

The function `eseqa` constructs tours on 1..n using an alternative algorithm.

The function `kntour_drop` removes instances of n from the euler tour, creating an open approximately eulerian path on the complete graph with n-1 nodes.

The function `kntour_add` inserts an extra node n+1 into an euler tour on the complete graph on n nodes. It adds a detour to the tour visiting all edges joining nodes 1..n to n+1. The result is an open approximately eulerian path on the complete graph with n+1 nodes.

Value

a numeric vector.

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

[hpaths](#), [eulerian](#).

Examples

```
require(PairViz)
eseq(5)
eseq(6)
```

etour

Constructs eulerian tours on a graph.

Description

etour- Constructs a closed eulerian tour on a graph using Hierholzer's algorithm. Returns a vector of node labels. If `weighted` is `TRUE` constructs a weight-decreasing eulerian using the modified Hierholzer's algorithm. Usually `etour` is not called directly, rather the generic function `eulerian` is used.

Usage

```
etour(g, start=NULL, weighted=TRUE)
```

Arguments

<code>g</code>	a graph satisfying <code>is_even_graph</code>
<code>start</code>	an optional starting node for the tour.
<code>weighted</code>	whether tour uses weights

Details

The supplied graph should satisfy `is_even_graph`. If `weighted` is `TRUE` the lowest weight edge is found, and the tour starts at the one of its nodes, picking the node with the bigger second-smallest edge weight. After that the tour follows weight-increasing edges. If `weighted` is `FALSE` weights are ignored.

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

Examples

```

require(PairViz)

g <- mk_even_graph(5)

etour(g)
g <- mk_even_graph(6) # adds 3 extra edges to g, so all nodes are even
etour(g)
etour(g, start= "4") # modifies the starting node

# On a general graph.
v <- LETTERS[1:4]
g <- new("graphNEL", nodes=v)
g <- addEdge(v[1], v[3:4], g, 1:2)
g <- addEdge(v[2], v[3:4], g, 3:4)
etour(g)

eulerian(g) # The eulerian wrapper looks after this

n <- LETTERS[1:5]
g <- new("graphNEL", nodes=n)
g <- addEdge(n[1], n[2:3], g)
g <- addEdge(n[2], n[3:5], g)
g <- addEdge(n[4], n[3], g)
is_even_graph(g)
etour(mk_even_graph(g))

```

eulerian

~~ Methods for Function eulerian ~~

Description

A generic function that returns an eulerian (or nearly eulerian) path based on `self`.

Usage

```
eulerian(self, start=NULL, weighted=TRUE)
```

Arguments

<code>self</code>	– see below
<code>start</code>	– see below
<code>weighted</code>	– see below

Value

A vector representing the eulerian- a character vector of node names for a graph, otherwise a numeric vector.

Methods

self = "even_graph" Uses `etour` to construct the eulerian. If `weighted` is `TRUE` a weighted eulerian is constructed, otherwise weights are ignored. A non-null `start` is the eulerian starting point.

self = "graphNEL" Augments the graph using `mk_euler_graph`, then invokes `eulerian` again on the augmented version. If `self` is not connected, (approximate) eulerians are formed for each connected component, which are joined by NAs.

self = "matrix" Builds a graph using `mk_euler_graph`, then invokes `eulerian` again on the result.

self = "numeric" Builds a graph with `self` nodes using `mk_euler_graph`, then invokes `eulerian` again on the result.

self = "ANY" Builds a graph using `mk_euler_graph`, then invokes `eulerian` again on the result.

Author(s)

C.B. Hurley and R.W. Oldford

References

C. Hierholzer (1873). *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren*. Math. Annalen VI, pp. 30-32.

Also, see [overview](#)

Examples

```
require(PairViz)

d <- as.matrix(eurodist)[1:8,1:8] # pick the first 8 cities

eulerian(d)
eulerian(d, weighted=FALSE)
```

even_graph

Class of graphs where all nodes have even degree

Description

This class is an extension of `graph_NEL`. For graphs of this class, euler tours may always be constructed. Objects of this class should be created by `mk_even_graph`

Slots

This class has all slots from "[graph_NEL](#)" plus:

dummy_node: Object of class "character"

extra_edges: Object of class "character"

weighted: Object of class "logical"

Extends

Class [graphNEL-class](#), directly. Class [graph-class](#), by class "graphNEL", distance 2.

Methods

is_even_graph signature (g = "graphNEL"): checks whether a graph has all nodes of even degree.

is_even_graph signature (g = "even_graph"): always TRUE.

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

Examples

```
showClass("even_graph")
```

find_path

Constructs a path from a matrix of edge weights.

Description

Returns a path, constructed by applying the function in `path` to the edge weights. If each edge has many weights, i.e if `edgew` is a matrix, these weights are first reduced by the function `combine` applied to the rows. If `path` is `NULL`, the returned path defaults to `1..nnodes(edgew)`

Usage

```
find_path(edgew, path=NULL, combine=sum, edge.index=edge_index(edgew), ...)
```

Arguments

edgew	Matrix (or vector) whose <i>i</i> th row (or element) has weights for pair indexed by pair in row <i>i</i> of edge.index.
path	a function used to construct the index path.
edge.index	A 2-column matrix with each row giving indices for corresponding weight in edgew.
combine	A function that combines the row of weights for an edge into a single numeric value.
...	

Author(s)

C.B. Hurley and R.W. Oldford

guided_pcp

Guided parallel coordinate plot.

Description

Draws a parallel coordinate plot, with an accompanying barchart showing an index (eg correlation, scagnostics) levels for each panel. An index legend is optional.

Usage

```
guided_pcp(data, edgew=NULL, path = NULL, pathw=NULL, zoom=NULL, pcpfn=pv_pcp,
  pcp.col = 1, lwd=0.5, panel.colors=NULL, pc.mar=c(1.5, 2, 2, 2),
  bar.col=1:9, bar.axes=FALSE, bar.mar=NULL, reorder.weights=TRUE,
  layout.heights=NULL, layout.widths=c(10, 1),
  main=NULL, legend=FALSE, cex.legend = 1, legend.mar=c(1, 4, 1, 1), ...)
```

Arguments

data	A data frame or matrix.
edgew	Matrix (or vector) whose rows give index values for each pair of variables.
path	an index vector specifying variable order, or a function. If a function, <code>find_path(edgew, path, ...)</code> constructs the index vector.
pathw	Matrix (or vector) whose rows give index values for each adjacent pair of variables in path. Usually this argument is NULL and pathw is computed from the path and edgew.
zoom	If provided, a numeric vector specifying a subsequence of path to display.
pcpfn	Function to draw the parallel coordinates.
pcp.col	Line colors.
lwd	Line widths.

```

panel.colors Background panel colors, passed to thepcpfn
pc.mar       Controls PCP margin size.
bar.col      Bar colors.
bar.axes     Draw barplot axes, if TRUE.
bar.mar      Controls barplot margin size.
reorder.weights
              If TRUE, reorder barplot indices so large values are drawn at the bottom.
layout.heights
              Controls the layout.
layout.widths
              Controls the layout.
main         Main title for PCP.
legend       If TRUE, draws the barplot index legend.
cex.legend   Controls legend text size.
legend.mar   Legend margin size.
...          Optional arguments

```

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

[pv_pcp](#)

Examples

```

require(PairViz)

data <- mtcars[,c(1,3:6)]
cols <- c("red","green")[mtcars[,9]+1 ] # transmission type, red=automatic

# add a correlation guide and find "better" hamiltonians...

# add a correlation guide...

corw <- dist2edge(as.dist(cor(data)))
edgew <- cbind(corw*(corw>0), corw*(corw<0))

# add a correlation guide to a PCP, positive cors shown in blue, negative in purple...

## Not run:

```

```

dev.new(width=3,height=3)

par(cex.axis=.65)

guided_pcp(data,edgew, pcp.col=cols,
            main="Correlation guided PCP",bar.col = c("blue","purple"))

dev.new(width=7,height=3)
par(cex.axis=.65)

guided_pcp(data,edgew, path=eulerian, pcp.col=cols,
            main="Correlation guided Eulerian PCP",bar.col = c("blue","purple"),bar.axes=TRUE)

## End(Not run)

# Scagnostic guides are useful here- see the demos for more examples.

```

hpaths	<i>Hamiltonian paths on the complete graph on 1..n, using Lucas-Walecki constructions.</i>
--------	--

Description

zigzag - Constructs hamiltonian paths where each pair (i,j) appears in at least one of the hamiltonians.

hpaths - Returns a hamiltonian decomposition on the complete graph with n nodes. See Details.

permute_hpaths - Returns a modified version of paths, where vertices are re-labelled so that the first hamiltonian is path1.

Usage

```

zigzag(n)
hpaths(n, matrix=TRUE,cycle=NULL,...)
permute_hpaths(path1,paths= hpaths(length(path1)), matrix=TRUE,...)

```

Arguments

n	a positive integer. For hpaths, n may also be a vector specifying the first hamiltonian.
matrix	if TRUE, returns a matrix where each row is a hamiltonian path, otherwise concatenates the rows into a vector.
cycle	If TRUE, returns hamiltonian cycles, i.e. every hamiltonian starts at the same node. If FALSE, returned paths are open. Defaults to TRUE for odd n,FALSE for even n.
path1	A vector- This will be the first hamiltonian of the returned hamiltonian decomposition.
paths	A matrix where each row is a hamiltonian.
...	

Details

`hpaths` - From graph theory we know that for odd n , the complete graph decomposes into $(n-1)/2$ edge distinct hamiltonian cycles, while for even n the graph decomposes into $n/2$ edge distinct hamiltonian paths. The default behaviour of the function `hpaths` is to produce the cycle decomposition for odd n and the path decomposition for even n .

However, if a `TRUE` value is supplied as argument `cycle`, the returned paths are cycles, and the result is a true decomposition for odd n , but for even n the last hamiltonian has some duplicate edges. If a `FALSE` value is supplied as argument `cycle`, the returned paths are open, and the result is a true decomposition for even n , but for odd n the last hamiltonian has some duplicate edges.

Value

A numeric matrix where each row contains a permutation of $1..n$, or these rows concatenated into a vector if `matrix=FALSE`.

Author(s)

C.B. Hurley and R.W. Oldford

References

D.E. Lucas (1892), *Recreations Mathematiques*, Vol II. Gauthier Villars, Paris.

Also see [overview](#)

See Also

[weighted_hpaths](#), [eseq](#).

Examples

```
require(PairViz)

zigzag(7)
hpaths(7) # the rows form a decomp. into hamiltonian cycles

# Now concatenate the rows and close the path
hpaths(7,matrix=FALSE)

# Form a decomposition into hamiltonian cycles- this decomposition is not exact, as the last
hpaths(7,cycle=FALSE)

# For even n, the default is a decomposition into hamiltonian paths, not cycles.
hpaths(6)

# If cycles are required for even n, the decomposition will not be exact and the last row du
hpaths(6,cycle=TRUE)

# If you want to specify the first hamiltonian of the decomposition, use
hpaths(1:7)
```

 mc_plot

Multiple comparison plot.

Description

For grouped data. Draws boxplots for each group and overlays with confidence intervals for pairwise comparison of means.

Usage

```
mc_plot(data, fit, path = eulerian, col = rainbow(length(data), s = 0.4), levels =
cifunction = function(a, level) TukeyHSD(a, conf.level = level)[[1]],
varwidth = TRUE, frame.plot = FALSE, boxwex = 0.3, cex=0.75, zoom=NULL, ci.yusr=NULL)
```

Arguments

data	A list of vectors, such as that returned by <code>split</code> .
fit	A fitted model object, usually an <code>ao</code> fit.
path	an index vector or a function. If a vector, groups are plotted in order <code>data[path]</code> . By default, it is the function <code>eulerian</code> , and produces an ordering where each pair of groups appears adjacently, with p-values roughly increasing as the sequence progresses.
col	A vector of colours, one per group.
levels	Vector of increasing confidence levels.
cifunction	Function that returns confidence intervals. Default uses <code>TukeyHSD</code> .
varwidth	Passed to <code>boxplot</code> .
frame.plot	Passed to <code>boxplot</code> .
boxwex	Passed to <code>boxplot</code> .
cex	Passed to <code>boxplot</code> .
zoom	If provided, a numeric vector specifying a subsequence of <code>path</code> to display.
ci.yusr	Specifies the vertical <code>par(usr)</code> for the confidence intervals. Defaults to <code>max</code> and <code>min</code> .
...	Optional arguments, passed to <code>path</code> and <code>overlayCI</code> .

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

See also [overlayCI](#)

Examples

```
require(PairViz)

data(cancer)
bx <- with(cancer, split(sqrt(Survival), Organ))

a <- aov(sqrt(Survival) ~ Organ, data=cancer)
## Not run:
dev.new(height=4.5, width=9.5)
op <- par(no.readonly = TRUE)

par(cex.axis=.75, cex.main = 1.6, cex.lab=1.4)
par(mar=c(3, 5, 3, 5))

mc_plot(bx, a, main="Pairwise comparisons of cancer types", ylab="Sqrt Survival")

par(op)
## End(Not run)
```

mk_complete_graph *Constructs a complete graph.*

Description

Constructs a complete graph, actually an instance of graph-NEL

Usage

```
mk_complete_graph(d)
```

Arguments

d an integer vector of length 1 which specified the number of nodes, a `dist`, or a symmetric matrix, either of which specify the nodes and edge weights.

Value

- a graph-NEL

Author(s)

C.B. Hurley and R.W. Oldford

Examples

```
require(PairViz)
d <- dist(rnorm(5))
g <- mk_complete_graph(d)
```

mk_even_graph	<i>Constructs an even graph</i>
---------------	---------------------------------

Description

~~ Methods for function `mk_even_graph`. Each of these return an instance of `even_graph`, where all nodes are of even degree. The result satisfies `is_even_graph`. The resulting graph yields an euler tour.

Methods

self = "graphNEL",use_weights=TRUE,add_edges=TRUE This is the workhorse method. If `self` does not satisfy `is_even_graph`, the graph is forced to be even by one of the following. If `add_edges` is `TRUE`, the odd nodes are paired off and a new edge added between each pair, possibly duplicating an existing edge. If `add_edges` is a vector of the odd nodes, they are paired off in this order. If `add_edges` is `FALSE` a new dummy node is added with edges going to all odd nodes.

self = "matrix",use_weights=TRUE,add_edges=TRUE first constructs a complete graph using `mk_complete_graph`, which is then augmented to be even.

self = "numeric",use_weights=FALSE,add_edges=TRUE first constructs a complete graph using `mk_complete_graph`, which is then augmented to be even.

self = "ANY",use_weights=TRUE,add_edges=TRUE first constructs a complete graph using `mk_complete_graph`, which is then augmented to be even.

self = "even_graph",add_edges=TRUE returns `self`.

References

see [overview](#)

See Also

[mk_complete_graph](#), [is_even_graph](#)

order_best	<i>Uses brute-force enumeration to find the best hamiltonian on the complete graph on 1..n.</i>
------------	---

Description

Returns the best hamiltonian

Usage

```
order_best(d, maxexact=9, nsamples=50000, path_weight=sum, cycle=FALSE, path_dir = path
```

Arguments

d	A dist, used to provide edge weights.
maxexact	If $\geq n$, finds the overall best hamiltonian, otherwise compares nsamples randomly generated permutations.
nsamples	If $n > \maxexact$, finds the best of nsamples randomly generated permutations .
cycle	If TRUE, finds the shortest cycle, otherwise the shortest open path.
path_weight	Combines edge weights into a single path/cycle weight.
path_dir	If a function is provided, used to re-orient the cycle/path. Default function is path_cor .
...	

Details

Requires package gtools. Currently it is possible to find the best hamiltonian by complete enumeration for up to 10 nodes. When path_dir is non NULL, the returned hamiltonian is also optimally oriented using best_orientation, which compares orientations via path_dir.

Value

A vector containing a permutation of 1..n

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

[order_tsp](#).

Examples

```
require(PairViz)
order_best(eurodist)
```

order_tsp

Uses tsp to find the best hamiltonian on the complete graph on 1..n

Description

Returns shortest cycle or path via tsp solver from package TSP

Usage

```
order_tsp(d, method = "nearest", cycle=TRUE, improve=FALSE, path_dir = path_cor, ...)
```

Arguments

d	A dist, used to provide edge weights.
method	Options are nearest_insertion, farthest_insertion, cheapest_insertion, arbitrary_insertion, nn, repetitive_nn, 2-opt and if concorde package is loaded, concorde. See solve_TSP for details.
improve	if TRUE, attempts to improve the solution using "2-opt".
cycle	If TRUE, finds the shortest cycle, otherwise the shortest open path.
path_dir	If a function is provided, used to re-orient the cycle/path. Default function is path_cor .
...	passed to solve_tsp

Details

Requires package TSP. When `path_dir` is non NULL, the returned hamiltonian is also optimally oriented using `best_orientation`, which compares orientations via `path_dir`.

Value

A vector containing a permutation of 1..n

Author(s)

C.B. Hurley and R.W. Oldford

References

See package TSP.

See Also

[order_best](#), [solve_TSP](#) in **TSP**.

Examples

```
require(PairViz)

rdist <- function(n) {
  d <- matrix(0,n,n)
  d[lower.tri(d)] <- runif(n*(n-1)/2)
  return(as.dist(d))
}

order_tsp(rdist(7))

edist <- as.dist(as.matrix(eurodist))
order_tsp(edist)
```

 overlayCI

Function to overlay confidence intervals on the current plot.

Description

Overlays confidence intervals on the current plot. Also draws a right hand axis, a horizontal broken line at zero, and marks the significant comparisons with an arrow, i.e. the CIs that do not intersect zero.

Usage

```
overlayCI(cis, xpos=NULL, ci.cols = NULL, ci.ex = 3, ci.ocol = "steelblue", p.col =
```

Arguments

<code>cis</code>	A matrix containing the confidence intervals. Each row corresponds to a different comparison, the first column is the estimated mean, and successive pairs of columns give the lower and upper limits for different confidence levels.
<code>ci.cols</code>	A vector of colours, one colour for each confidence level. Defaults to shades of grey.
<code>ci.ex</code>	Controls confidence interval line width.
<code>xpos</code>	Horizontal positions where CIs are drawn. Defaults to 1.5,2.5,3.5,..
<code>ci.ocol</code>	Colour of zero line.
<code>p.col</code>	Colour of point used for CI centre.
<code>pch</code>	Symbol used for CI centre.
<code>sig.col</code>	Colour of arrow marking significant comparisons.

`sig.lwd` Width of arrow marking significant comparisons.
`yusr` Specifies the vertical `par(usr)` .Defaults to max and min.
`ci.label` Label drawn on right margin.
...

Note

This function is called by `mc_plot`

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

See Also as [mc_plot](#)

overview

Overview of PairViz package

Description

Implements methods described in Hurley and Oldford paper.

There are functions for constructing eulerian paths on complete graphs- see [eseq](#), [hpaths](#), and [weighted_hpaths](#), and eulerians on general graphs- see [etour](#) and [eulerian](#).

There are also functions for new types of graphics, [mc_plot](#) and [guided_pcp](#) and a bar-chart/mosaic variant [table_plot](#).

Author(s)

C.B. Hurley and R.W. Oldford

References

C.B. Hurley and R.W. Oldford, Pairwise display of high dimensional information via Eulerian tours and Hamiltonian decompositions. 2008 submitted.

C.B. Hurley and R.W. Oldford, Eulerian tour algorithms for dta visualization and the PairViz package. 2008 submitted.

path_cor	<i>Measures the tendency of edge weights to increase.</i>
----------	---

Description

Returns the (Kendalls tau) correlation of the edge weights with the vector 1.. (number of weights).

Usage

```
path_cor(edgew, method = "kendall")
```

Arguments

edgew	A vector of edge weights.
method	passed to <code>cor</code>

path_weights	<i>Utility functions to manipulate pairwise information.</i>
--------------	--

Description

These functions perform calculations on edge matrices containing pairwise information.

Usage

```
path_weights(edgew, path, symmetric = TRUE, edge.index=edge_index(edgew), ...)
path_cis(edgew, path, edge.index=edge_index(edgew))
edge2dist(edgew, edge.index=edge_index(edgew))
dist2edge(d)
edge_index(x, order="default")
```

Arguments

edgew	A Matrix (or vector) whose ith row (or element) has weights for pair indexed by pair in row i of edge.index. For edge2dist, edgew should be a vector.
path	Vector of indices into rows of edgew.
symmetric	If TRUE edge weights are interpreted as symmetric.
edge.index	A 2-column matrix with each row giving indices for corresponding weight in edgew.
d	A dist or matrix of distances.
order	If "low.order.first" or "scagdf", lists lowest index pairs first, otherwise lists pairs starting with 1, then 2 etc.
x	An edgew matrix or vector, or a positive integer.
...	

Details

`path_weights` - Returns matrix of path weights so that the *i*th row of result contains weights for indices `path[i]`, `path[i+1]`

`path_cis` - Returns matrix of path confidence intervals so that the *i*th row of result contains intervals for `mean-path[i]` - `mean-path[i+1]`

`edge2dist` - Returns a `dist`, containing elements of `edgew`.

`dist2edge` - Returns a vector of edge weights.

`edge_index` -A generic function. Returns a 2-column matrix with one row for each edge. Each row contains an index pair *i,j*. If `order` is "low.order.first" or "scagdf", lists lowest index pairs first - this is the default ordering for class `scagdf`, otherwise lists pairs starting with 1, then 2 etc

`nnodes` - Here `edgew` contains edge weights for a complete graph; returns the number of nodes in this complete graph.

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

Examples

```
require(PairViz)

s <- matrix(1:40,nrow=10,ncol=4)

edge2dist(s[,1])

path_weights(s,1:4)
path_weights(s,eseq(5))

fm1 <- aov(breaks ~ wool + tension, data = warpbreaks)
tuk <- TukeyHSD(fm1, "tension")[[1]]

# Here the first argument (weight matrix) can have number of columns

path_weights(tuk,c(1:3,1))

# Here the first argument (weight matrix) should have an odd number of columns-
# the first is the mean difference, other column pairs are endpoints of CIs

path_cis(tuk[,-4],c(1:3,1))
```

`pv_pcp`*Enhanced parallel coordinate plot*

Description

This function draws a parallel coordinate plot of data. Variables may be reordered and panels colored in the display. It is a modified version of `parcoord` {MASS}.

Usage

```
pv_pcp(data, order = NULL, panel.colors = NULL, col = 1, lty = 1, horizontal = TRUE)
```

Arguments

<code>data</code>	a numeric matrix
<code>order</code>	the order of variables. Default is the order in data.
<code>panel.colors</code>	either a vector or a matrix of panel colors. If a vector is supplied, the <i>i</i> th color is used for the <i>i</i> th panel. If a matrix, dimensions should match those of the variables. Diagonal entries are ignored.
<code>col</code>	a vector of colours, recycled as necessary for each observation.
<code>lty</code>	a vector of line types, recycled as necessary for each observation.
<code>horizontal</code>	If TRUE, orientation is horizontal.
<code>mar</code>	margin parameters, passed to <code>par</code> .
<code>scale</code>	If TRUE each column of data is scaled to unit interval prior to plotting.
<code>axis.width</code>	A vector of widths (recycled as needed) used to stretch the axes.
<code>...</code>	graphics parameters which are passed to <code>matplot</code> .

Details

If `panel.colors` is a matrix and `order` is supplied, `panel.colors` is reordered.

Author(s)

Catherine B. Hurley

See Also

[guided_pcp](#)

table_plot	<i>Plots rectangles on a grid</i>
------------	-----------------------------------

Description

Plots rectangles on a grid- a barchart/mosaic variant which facilitates pairwise comparisons.

Usage

```
table_plot(rectw, recth, col="grey50", gapx = NULL, gapy = NULL, spacex = 0.03, spa
```

Arguments

rectw	An n*m matrix of rectangle widths, or a vector of m column widths.
recth	An n*m matrix of rectangle heights, or a vector of n row heights.
col	Rectangle fill colours.
gapx	Gaps in the x-direction. If provided should be a vector of length m-1.
gapy	Gaps in the x-direction. If provided should be a vector of length n-1.
spacex	A single value- extra space between columns as a fraction of maximum row total of rectw .
spacey	A single value- extra space between rows as a fraction of maximum column total of recth .
xjust	Horizontal justification of rectangles- "center", "left", or "right".
yjust	Vertical justification of rectangles- "center", "bottom", or "top".
xruler	Specifies position of rulers drawn parallel to x-axis. Values are a subset of ("top","center","bottom")
yruler	Specifies position of rulers drawn parallel to y-axis. Values are a subset of ("left","center","right")
color.ruler	Color for the rulers.
xlab	X label
ylab	Y label
...	Passed to plot.

Author(s)

Catherine Hurley

References

See [overview](#)

See Also

See also [barplot](#), [mosaicplot](#)

Examples

```
## Not run:
require(PairViz)

tab <- apply(HairEyeColor, c(1, 2), sum)

dev.new()
par(mar=c(3,3,1,1))
par(cex=.6,mgp=c(2, -.5, 0))
table_plot(sqrt(tabc),sqrt(tabc))
# this table plot has cells with widths and heights proportional to the square root of cell

tabp <- prop.table(tabc,2)

table_plot(apply(tab,2,sum),tabp) # make cell widths proportional to margin totals, heights

cols <- 2:5
table_plot(apply(tab,2,sum),tabp, yjust="bottom",col=cols,yruler=c("left","right")) # add co

# The result is similar to the mosaic, without the mosaic effect of equalizing gaps. In the

o <- hpaths(1:4)[2,]
table_plot(apply(tab,2,sum)[o],tabp[,o], yjust="bottom",col=cols,yruler=c("left","right"))
# Permutes the columns so all pairs of columns can be compared. In the second permutation ca

dev.new()
par(mar=c(3,3,1,1))
par(mgp=c(2, -.5, 0))
mosaicplot(t(tab)[,nrow(tab):1],col=rev(cols),main="")
# mosaic- good for seeing deviations from independence. hard to compare conditional probs,
# except for those in the bottom and top rows.
## End(Not run)
```

weighted_hpaths *Constructs weight decreasing hamiltonian paths*

Description

Returns a modified version of `paths`, where component paths/cycles are re-oriented so low weight edges occur first, and the component paths/cycles are then permuted so low-weight paths are first.

Usage

```
weighted_hpaths(d, path1 = NULL, paths=NULL, matrix=TRUE, cycle=NULL, path_weight=s
```

Arguments

<code>d</code>	A <code>dist</code> , used to provide edge weights.
<code>path1</code>	A vector giving a hamiltonian. This will be the first path of the returned hamiltonian. The default is obtained from <code>order_tsp</code> .
<code>paths</code>	A matrix where each row is a hamiltonian. Default comes from <code>hpaths</code> .
<code>matrix</code>	if <code>TRUE</code> , returns a matrix where each row is a hamiltonian path, otherwise concatenates the rows into a vector. For odd <code>n</code> , the starting node is appended to close the eulerian.
<code>cycle</code>	If <code>TRUE</code> , the <code>weighted_hpaths</code> algorithm evaluates <code>path_weight</code> on hamiltonian cycles, if <code>FALSE</code> , on open hamiltonian paths. Default is <code>TRUE</code> for odd <code>n</code> and <code>FALSE</code> for even <code>n</code> .
<code>path_weight</code>	A function used combine path weights into a single value. Default function is <code>path_cor</code> .
<code>path_dir</code>	A function used to evaluate a path start and orientation.
<code>...</code>	passed to <code>path_weight</code>

Details

If `path` is not provided, find the hamiltonian (path for even `n`, cycle for odd `n`) with the smallest total weight. Applying `path_dir` to edge weights, pick the starting and point orientation for `path1` giving the largest `path_dir` value. (For open paths, there are only two possible starts, for cycles there are `n`). Apply this node labelling to the hamiltonians in the rows of `paths`. Use criterion `path_dir` again to find the best orientation for each of rows 2... of `paths` and permute these rows in order of increasing `path_weight`.

Author(s)

C.B. Hurley and R.W. Oldford

References

see [overview](#)

See Also

[hpaths](#), [eulerian](#).

Examples

```
require(PairViz)

weighted_hpaths(dist(rnorm(6)))

weighted_hpaths(dist(rnorm(7)))
```

Index

- *Topic **aplot**
 - overlayCI, 18
- *Topic **classes**
 - even_graph, 7
- *Topic **color**
 - pv_pcp, 21
- *Topic **datasets**
 - cancer, 2
- *Topic **graphs**
 - best_orientation, 1
 - eseq, 3
 - etour, 4
 - even_graph, 7
 - find_path, 8
 - mk_complete_graph, 14
 - mk_even_graph, 14
 - order_best, 15
 - order_tsp, 16
 - overview, 19
 - path_weights, 20
 - weighted_hpaths, 24
- *Topic **hplot**
 - guided_pcp, 8
 - mc_plot, 12
 - overview, 19
 - pv_pcp, 21
 - table_plot, 22
- *Topic **methods**
 - eulerian, 6
 - mk_even_graph, 14
- *Topic **multivariate**
 - pv_pcp, 21
- *Topic **optimize**
 - best_orientation, 1
 - find_path, 8
 - order_best, 15
 - order_tsp, 16
 - overview, 19
 - weighted_hpaths, 24
- aov, 12
- barplot, 23
- best_orientation, 1
- boxplot, 13
- cancer, 2
- cor, 19
- dist2edge (*path_weights*), 20
- edge2dist (*path_weights*), 20
- edge_index (*path_weights*), 20
- eseq, 3, 11, 19
- eseqa (*eseq*), 3
- etour, 4, 19
- eulerian, 2, 4, 6, 12, 19, 25
- eulerian, ANY-method (*eulerian*), 6
- eulerian, even_graph-method (*eulerian*), 6
- eulerian, graphNEL-method (*eulerian*), 6
- eulerian, matrix-method (*eulerian*), 6
- eulerian, numeric-method (*eulerian*), 6
- eulerian-methods (*eulerian*), 6
- even_graph, 7
- even_graph-class (*even_graph*), 7
- find_path, 8
- graph-class, 7
- graph_NEL, 7
- graph_NEL-class (*even_graph*), 7
- graphNEL-class, 7
- guided_pcp, 8, 19, 22
- hpaths, 2, 4, 10, 19, 25
- is_even_graph, 15

`is_even_graph` (`even_graph`), 7
`is_even_graph`, `even_graph`-method
 (`even_graph`), 7
`is_even_graph`, `graphNEL`-method
 (`even_graph`), 7

`kntour_add` (`eseq`), 3
`kntour_drop` (`eseq`), 3

`mc_plot`, 12, 18, 19
`mk_complete_graph`, 14, 15
`mk_even_graph`, 14
`mk_even_graph`, ANY-method
 (`mk_even_graph`), 14
`mk_even_graph`, `even_graph`-method
 (`mk_even_graph`), 14
`mk_even_graph`, `graphNEL`-method
 (`mk_even_graph`), 14
`mk_even_graph`, `matrix`-method
 (`mk_even_graph`), 14
`mk_even_graph`, `numeric`-method
 (`mk_even_graph`), 14
`mk_even_graph`-methods
 (`mk_even_graph`), 14
`mosaicplot`, 23

`nnodes` (`path_weights`), 20

`order_best`, 15, 17
`order_tsp`, 16, 16
`overlayCI`, 13, 18
`overview`, 2, 4–7, 9, 11, 13, 15, 16, 18, 19,
 21, 23, 25

PairViz-package (`overview`), 19
`path_cis` (`path_weights`), 20
`path_cor`, 15, 17, 19, 25
`path_weights`, 20
`permute_hpaths` (`hpaths`), 10
`pv_pcp`, 9, 21

`solve_TSP`, 16, 17

`table_plot`, 19, 22

`weighted_hpaths`, 11, 19, 24

`zigzag` (`hpaths`), 10