Package ‘PairViz’

August 12, 2022

Version 1.3.6

Date 2022-08-12

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

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

Title Visualization using Graph Traversal

Description Improving graphics by ameliorating order effects, using Eulerian tours and Hamiltonian decompositions of graphs. References for the methods presented here are C.B. Hurley and R.W. Oldford (2010) <doi:10.1198/jcgs.2010.09136> and C.B. Hurley and R.W. Oldford (2011) <doi:10.1007/s00180-011-0229-5>.

Depends TSP,gtools, graph, methods

Suggests knitr, rmarkdown, Sleuth3, colorspace,multcomp,igraph, scales,RColorBrewer,alr4,scagnostics

License GPL-2

VignetteBuilder knitr

RoxygenNote 7.2.1

URL https://cbhurley.github.io/PairViz/

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-12 15:00:05 UTC

R topics documented:

`best_orientation` *Re-orients a path to be weight-decreasing*

Description

Re-orients 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`	Sepcifies 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

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.

`desaturate_color` *Desaturates colors*

Description

Desaturates colors

Usage

```
desaturate_color(cols, frac = 0.8)
```

Arguments

`cols`	Colors
`frac`	Fraction to desaturate by.

Value

Desaturated version of orginal colors

`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 Kn where `n` is odd

Details

The algorithm used for eseq builds up a path on 1..n by appending extra edges on to the path on nodes 1..(n-2).

The function eseqa constructs paths on 1..n using an alternative algorithm. For odd n, the tour starts at 1, then takes steps of size 1,2,..m repeatedly, where m is (n-1)/2, For even n, the path constructed is formed as eseqa(n+1), followed by dropping node n+1.

The function kntour_drop removes instances of n from the 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 a tour on nodes 1, ..n. 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 an 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

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

Details

The supplied graph should satisfyis_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. The returned tour is typically a closed path. However, if the last edge is a duplicated edge added to make the graph even, this edge is omitted and the result is an open path.

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

eulerian(6) # The eulerian wrapper looks after making even graph,
#also returns numbers rather than nodes

# 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) # Equivalently, use eulerian wrapper
```

```
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(g) # Equivalently, use eulerian wrapper
```

`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

`self`	– see below
`start`	– see below
`weighted`	– see below

Value

A vector representing the eulerian- a character vector of node names for a graph, otherwise a numeric vector. If the graph is not connected, the result is a list of eulerians for each connected component.

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 verion. If self is not connected, (approximate) eulerians are formed for each connected component, which are returned as a list.

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). Uber die Moglichkeit, 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) # In this case, starts at city 1 and ends at city 8
```

`even_graph` *Class of graphs where all nodes have even degree*

Description

This class is an extension of `graphNEL-class`. 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 `graphNEL-class` 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 ith row (or element) has weights for pair indexed by pair in row i of edge.index.
`path`	a function used to construct the index path.
`combine`	A function that combines the row of weights for an edge into a single numeric value.
`edge.index`	A 2-column matrix with each row giving indices for corresponding weight in `edgew`.
`...`	passed to path construction function.

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=pcp,
    pcp.col = 1,lwd=0.5, panel.colors=NULL, pcp.mar=c(1.5,2,2,2), pcp.scale=TRUE,
    bar.col=1:9,bar.axes=FALSE, bar.mar=NULL,bar.ylim=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, `find_path(edgew,path,...)` 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 the`pcpfn`
pcp.mar	Controls PCP margin size.
pcp.scale	If TRUE, the variables will be scaled to 0-1 range, otherwise the data is not scaled.
bar.col	Bar colors.
bar.axes	Draw barplot axes, if TRUE.
bar.mar	Controls barplot margin size.
bar.ylim	Vertical limits of bar plot.
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

pcp,catpcp

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,lwd=2,
       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 *Hamiltonian paths on the complete graph on 1..n, using Lucas-Walecki constructions.*

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.
...	Ignored.

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 Matematiques, 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 row duplicates edges
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 duplicates edges

hpaths(6,cycle=TRUE)

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

knn_graph *Functions to construct graphs.*

Description

Functions to construct graphs- see details below.

Usage

```
knn_graph(g, k = 2)
dn_graph(g, d = 1, test=`<=`)
mk_binary_graph(n,sep="",delta=1,test=`==`)
mk_hypercube_graph(n,sep="")
mk_line_graph(g,sep="-")
kspace_graph(n,m, link=NULL,sep="-")
graph_product(g,h, type="cartesian",sep="-")
graph_compose(g,h,sep="-")
graph_sum(g,h,combineWeight=`+`)

bipartite_graph(n1,n2)
iterated_line_graph(g,sep="-")
```

Arguments

g	a graph
h	a graph
n	a positive integer, or a character vector.
k	a positive integer
d	an edge weight
test	used to select edges.
sep	used to form node names of new graph.
m	subsets of size m are nodes of kneser graph.
link	A positive number or NULL. If NULL, the returned graph is complete. Otherwise edges for subsets sharing link elements.
type	the type of graph product, one of "cartesian", "strong" or "tensor"
n1	a character vector.
n2	a character vector.
delta	used to select edges.
combineWeight	used to combine weights.

Details

knn_graph- returns a symmetric k nearest neighbour graph

dn_graph- returns a graph formed from g where edges of satisfy test(weight, d). The default retains edges whose weight are 1 are less. Nodes with no edges are also removed.

mk_hypercube_graph- returns a hypercube graph with 2^n nodes

mk_binary_graph(n,sep="",delta=1,test='==') - returns a graph with 2^n nodes. Undirected edges join nodes A and B whose binary vectors satisfy $a_i \leq b_i, i = 1, \ldots, n$ and test($\sum(b_i - a_i)$, delta) is true.

mk_line_graph- returns the line graph of g

kspace_graph- returns a graph where nodes are subsets of size m from n. Edges are connect nodes whose subsets share link elements. The standard kneser graph has link=0. When link is NULL, returned graph is complete.

graph_product(g,h, type="cartesian",sep="-")- returns the graph product of g and h.

graph_compose(g,h,sep="-")- returns the graph composition of g and h.

bipartite_graph(n1,n2)- returns the complete bipartite graph with node sets n1 and n2.

graph_sum(g,h,combineWeight='+')- returns a graph whose nodes and edges are the union of those in g and h. Weights of common edges are combined using combineWeight.

iterated_line_graph- returns the iterated line graph of g, with compression of nodes as described in the reference Hurley and Oldford(2008) given below.

Author(s)

C.B. Hurley and R.W. Oldford

References

See any Graph Theory text. Also C.B. Hurley and R.W. Oldford, Graphs as navigational infrastructure for high dimensional data spaces. 2008 submitted.

Examples

```
# See the demo file nav.R
```

---

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 = c(0.9, 0.95, 0.99), varwidth = TRUE, frame.plot = FALSE,
boxwex = 0.3, cex=0.75, zoom=NULL,ci.yusr=NULL,ci.pos=FALSE,...)
```

Arguments

`data`	A list of vectors, such as that returned by `split`.
`fit`	Either an `aov` fit, or else a matrix with columns estimate, followed by confidence intervals. If fit is not an `aov` fit, the path argument should be a vector.
`path`	an index vector or a function. If a vector, groups are plotted in order `data[path]`. By default, it is the function `eulerian`, 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.
`varwidth`	Passed to `boxplot`.
`frame.plot`	Passed to `boxplot`.
`boxwex`	Passed to `boxplot`.
`cex`	Passed to `boxplot`.
`zoom`	If provided, a numeric vector specifying a subsequence of path to display.
`ci.yusr`	Specifies the vertical `par(usr)` for the confidence intervals. Defaults to max and min.
`ci.pos`	If TRUE, all CIs are mu(max) - mu(min), otherwise mu(right) - mu(left).
`...`	Optional arguments, passed to `boxplot` and `overlayCI`.

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 = 1.0, cex.lab=1)
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 character vector of nodes names, 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 *Constructs an even graph*

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 folowing. 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 *Uses brute-force enumeration to find the best hamiltonian on the complete graph on 1..n.*

Description

Returns the best hamiltonian

Usage

```
order_best(d, maxexact=9,nsamples=50000,path_weight=sum,
cycle=FALSE,path_dir = path_cor,...)
```

Arguments

d
: A dist, used to provide edge weights.

maxexact
: If the sequence length is <= maxexact, find the overall best hamiltonian, otherwise compares nsamples randomly generated permutations.

nsamples
: If the sequence length is <= 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.

...
: Ignored.

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=FALSE,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 = 2, ci.ocol = "grey40",
p.col = "grey40", pch = 1, sig.col = "red", sig.lwd = 1, yusr = NULL,
ci.label="Differences",ci.cex=0.5,arrow.length=0.1,...)
```

Arguments

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

pch	Symbol used for CI centre.
sig.col	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.
ci.cex	Controls size of CI mean point symbol.
arrow.length	Controls size arrow at right hand axis.
...	Ignored

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`, `catpcp` and `guided_pcp` and a barchart/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. Journal of Computational and Graphical Statistics. 19(10), pp. 861–886, 2010.

C.B. Hurley and R.W. Oldford, Eulerian tour algorithms for data visualization and the PairViz package. Computational Statistics, 26(4), pp 613–633, 2011.

`path_cor` *Measures the tendency of edge weights to increase.*

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 `cor`

`path_weights` *Utility functions to manipulate pairwise information.*

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),ci.pos=FALSE)
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`.
`ci.pos`	If TRUE, all CIs are mu(max) - mu(min), otherwise mu(right) - mu(left).
`d`	A `dist` or matrx of distances.
`order`	If "low.order.first" or "scagnostics", lists lowest index pairs first, otherwise lists pairs starting with 1, then 2 etc.
`x`	An edgew matrix or vector, or a positive integer.
`...`	Ignored

Details

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

path_cis - Returns matrix of path confidence intervals so that the ith 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 "scagnostics", 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))
```

pcp *Enhanced parallel coordinate plots.*

Description

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

catpcp draws a parallel coordinate plot variant for categorical data.

Usage

```
pcp(data, order = NULL, panel.colors = NULL, col = 1, lty = 1,
horizontal = TRUE, mar = NULL, scale = TRUE, axis.width = 0,
axis.grid.col="grey70",connect=TRUE, ...)

catpcp(data, order = NULL, pcpbars, barvars = 1:ncol(data),
pcpbars.border = "black", pcpbars.col = NULL, pcpbars.labels = FALSE,
pcpbars.axis.at = NULL, pcpbars.axis.labels = NULL,
axis.width = 0.2,connect=TRUE, ...)
```

Arguments

data	A data frame or matrix.
order	an index vector specifying variable order. If NULL, all variables are used.
panel.colors	either a vector or a matrix of panel colors. If a vector is supplied, the ith color is used for the ith panel. If a matrix, dimensions should match those of the variables. Diagonal entries are ignored.
col	a vector of colours, recycled as necessary for each observation.
lty	a vector of line types, recycled as necessary for each observation.
horizontal	If TRUE, orientation is horizontal.
mar	margin parameters, passed to par.
scale	If TRUE, the variables are scaled to the unit interval.
axis.width	Width of each of the parallel axes.
axis.grid.col	Color of variable axes. Use NULL for no axes.
connect	If FALSE, line segments are not connected. Points are drawn if axis.width=0.
pcpbars	A list, with one component per barvar. Component i is a matrix with the bottom and top of the bars for that variable.
barvars	Categorical variables where overlayed bars show the level frequency.
pcpbars.border	Border colour of the bars.
pcpbars.col	Colour of the bars.

pcpbars.labels Labels for the bars.

pcpbars.axis.at
: Axis label positions for the bars.

pcpbars.axis.labels
: Axis label text for the bars.

... other parameters, passed to pcp by catpcp

Examples

```
require(PairViz)
y <- as.data.frame(as.table(HairEyeColor))

colvar <- 3 # any of 1:3 will do
y <- y[order(y[,colvar]),] # ensures that cases are ordered by colour within each factor level
ylong <- apply(y[,-4],2, function(x) rep(x,times=y[,4]))

cols <- desaturate_color(rainbow(4,alpha=0.3),.5)
cols <- cols[as.numeric(as.factor(ylong[,colvar]))]



ds <- factor_spreadout(ylong)



dev.new(width=5,height=2.5)
par(mar=c(2,1,2,1))
par( cex.axis=.8,cex=.8)


catpcp(ds$data,col=cols,lwd=2,pcpbars=ds$bars,pcpbars.labels=TRUE,main="Hair Eye data")
```

---

spreadout *Functions to prepare for categorical parallel coordinates, drawn by catpcp.*

Description

factor_spreadout spreads out the data at each factor level. rater_spreadout spreads out the data at each rating level. The rater version is appropriate when the variables (factors) have all the same levels.

Usage

```
factor_spreadout(d)
rater_spreadout(d, levs, minspace = NULL,scale=FALSE)
```

Arguments

`d`	A data frame where each variable can be interpreted as a factor.
`levs`	The rating levels. Specifying this controls the order of rating levels on each axis.
`minspace`	The minimum amount of space between the bars.
`scale`	If scale=FALSE, the ith rater values are spreadout about the value i. If scale=TRUE, all values are scaled to 0-1.

Details

factor_spreadout spreads out the data at each factor level. It returns a list with two components. The first is data, containing the spreadout data, scaled to 0-1. The second is bars, which is a list whose ith component gives the bottom and top of the bars for the ith variable of d.

rater_spreadout spreads out the data at each rater level. It returns a list with two components. The first is data, containing the spreadout data. If scale=FALSE, the ith rater values are spreadout about the value i. If scale=TRUE, all values are scaled to 0-1. The second component is bars, which is a list whose ith component gives the bottom and top of the bars for the ith variable of d.

`table_plot` *Plots rectangles on a grid*

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, spacey = 0.03, xjust = "center", yjust = "center",
xruler = NULL, yruler = NULL, color.ruler = "grey32",
pch0=1,xlab=NULL,ylab=NULL, plot=TRUE,...)
```

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.
pch0	Symbol for zero cell size. May be NULL.
xlab	X label
ylab	Y label
plot	If TRUE, draw tge plot. Otherwise returns a matrix where each row is the coordinates of a the calculated rectangle.
...	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(tab),sqrt(tab))
# this table plot has cells with widths and heights proportional to the square root of cell counts.

tabp <- prop.table(tab,2)

table_plot(apply(tab,2,sum),tabp) # make cell widths proportional to
```

```
#margin totals, heights to conditional prob

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

# The result is similar to the mosaic, without the mosaic effect of equalizing gaps.
#In the table version the rectangles line up across rows,
#so comparing heights, ie. conditional probs is easier.

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 can easily see that
#p(black|blue eyes)> p(black|green eyes)


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=sum, path_dir = path_cor,...)
```

Arguments

d	A dist, used to provide edge weights.
path1	A vector giving a hamiltonian. This will be the first path of the returned hamiltonian. The default is obtained from order_tsp.
paths	A matrix where each row is a hamiltonian. Default comes from hpaths.
matrix	if TRUE, returns a matrix where each row is a hamiltonian path, otherwise concatenates the rows into a vector. For odd n, the starting node is appended to close the eulerian.

cycle	If TRUE, the weighted_hpaths algorithm evaluates path_weight on hamiltonian cycles, if FALSE, on open hamiltonian paths. Default is TRUE for odd n and FALSE for even n.
path_weight	A function used combine path weights into a single value. Default function is path_cor.
path_dir	A function used to evaluate a path start and orientation.
...	passed to path_weight

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