

Package ‘HiveR’

February 14, 2012

Type Package

Title 2D and 3D Hive Plots for R

Version 0.2-1

Date 2011-12-12

Author Bryan A. Hanson DePauw University, Greencastle Indiana USA

Maintainer Bryan A. Hanson <hanson@depauw.edu>

Description HiveR is an R package for creating and plotting 2D and 3D hive plots. Hive plots are a unique method of displaying networks of many types in which node properties are mapped to axes using meaningful properties rather than being arbitrarily positioned. The hive plot concept was invented by Martin Krzywinski at the Genome Science Center (www.hiveplot.net/).
Keywords: networks, food webs, linnnet, systems biology, bioinformatics.

License GPL-3

Imports grid, RColorBrewer, rgl, xtable, bipartite, RFOC, sna

Suggests mvbutils, FuncMap, lattice, reshape

URL <http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

ByteCompile TRUE

Repository CRAN

Date/Publication 2011-12-13 08:36:08

R topics documented:

HiveR-package	2
adj2HPD	3
Arroyo	5
cart2sph	6

chkHPD	7
dot2HPD	8
drawHiveSpline	9
HivePlotData	10
manipAxis	11
mineHPD	12
plotHive	13
ranHiveData	16
rcsr	18
sumHPD	21

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

HiveR-package	<i>2D and 3D Hive Plots for R</i>
---------------	-----------------------------------

Description

HiveR is an R package for creating and plotting 2D and 3D hive plots. Hive plots are a unique method of displaying networks of many types in which node properties are mapped to axes using meaningful properties rather than being arbitrarily positioned. The hive plot concept was invented by Martin Kryzwinski at the Genome Science Center (www.hiveplot.net/). Keywords: networks, food webs, linnet.

Details

Package: HiveR
 Type: Package
 Version: 0.2-1
 Date: 2011-12-12
 License: GPL-3

Author(s)

Bryan A. Hanson, DePauw University, Greencastle Indiana USA

Maintainer: Bryan A. Hanson <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

adj2HPD

*Process an adjacency graph into a HivePlotData object***Description**

This function will take an adjacency graph and convert it into a basic `HivePlotData` object. Further manipulation will almost certainly be required before the data can be plotted.

Usage

```
adj2HPD(M = NULL, axis.cols = NULL, type = "2D", desc = NULL, ...)
```

Arguments

<code>M</code>	A matrix with named dimensions. The names should be the node names.
<code>axis.cols</code>	A character vector giving the colors desired for the axes.
<code>type</code>	One of <code>c("2D", "3D")</code> . If 2D, a <code>HivePlotData</code> object suitable for use with <code>plotHive</code> will be created and the eventual hive plot will be static and 2D. If 3D, the <code>HivePlotData</code> object will be suitable for a 3D interactive plot using <code>plot3dHive</code> .
<code>desc</code>	Character. A description of the data set.
<code>...</code>	Other parameters to be passed downstream.

Details

This function produces a "bare bones" `HivePlotData` object. The names of the dimensions of `M` are used as the node names. All nodes are given size 1, an id number (1:number of nodes), are colored black and are assigned to axis 1. The edges are all gray, and the weight is `M[i,j]`. The user will likely have to manually make some changes to the resulting `HivePlotData` object before plotting. Alternatively, `mineHPD` may be able to extract some information buried in the data, but even then, the user will probably need to make some adjustments. See the examples.

Value

A `HivePlotData` object which is capable of more sophisticated and flexible processing of a `.dot` file if the file contains tags.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

See Also[dot2HPD](#)**Examples**

```

### Note: this example has questionable scientific value!
### The purpose is to show how to troubleshoot and
### manipulate a HivePlotData object.

set.seed(31)
require(bipartite)
data(Safariland)

# You may wish to do ?Safariland or ?Safari for background

hive1 <- adj2HPD(Safariland, desc = "Safariland data set from bipartite")
sumHPD(hive1)

# Note that all nodes are one axis with radius 1. Process further:

hive2 <- mineHPD(hive1, option = "rad <- tot.edge.count")
sumHPD(hive2)

# All nodes still on 1 axis but degree has been used to set radius

# Process further:

hive3 <- mineHPD(hive2, option = "axis <- source.man.sink")
sumHPD(hive3, chk.all = TRUE)

# Note that mineHPD is generating some warnings, telling us
# that the first 9 nodes were not assigned to an axis. Direct
# inspection of the data shows that these nodes are insects
# that did not visit any of the flowers in this particular study.

# Pretty up a few things, then plot:

hive3$edges$weight <- sqrt(hive3$edges$weight)*0.5
hive3$nodes$size <- 0.5
plotHive(hive3)

# This is a one-sided hive plot of 2 axes, which results
# from the curvature of the splines. We can manually fix
# this by reversing the ends of edges as follows:

for (n in seq(1, length(hive3$edges$id1), by = 2)) {
  a <- hive3$edges$id1[n]
  b <- hive3$edges$id2[n]
  hive3$edges$id1[n] <- b
  hive3$edges$id2[n] <- a
}

```

```
plotHive(hive3)
```

 Arroyo

Plant-pollinator data sets in hive plot data format

Description

Plant-pollinator data sets which were derived ultimately from Vasquez and Simberloff, 2003. These are two-trophic level systems that have almost exactly the same plants and pollinators. Safari is from an undisturbed area, while Arroyo is from a nearby location grazed by cattle. In the original publication, the data sets are called Safariland and Arroyo Goye. See Details for how the original data was converted.

Usage

```
data(Arroyo)
data(Safari)
```

Format

The format is:

List of 7

\$ nodes :'data.frame': 39 obs. of 6 variables:

..\$ id : int [1:39] 1 2 3 4 5 6 7 8 9 10 ...

..\$ lab : chr [1:39] "Aristotelia chilensis" "Alstroemeria aurea" "Schinus patagonicus" "Berberis darwinii" ...

..\$ radius: num [1:39] 0.404 0.775 0.507 0.79 0.55 ...

..\$ axis : int [1:39] 1 1 1 1 1 1 1 1 1 1 ...

..\$ size : num [1:39] 1 1 1 1 1 1 1 1 1 1 ...

..\$ color : chr [1:39] "black" "black" "black" "black" ...

\$ edges :'data.frame': 43 obs. of 4 variables:

..\$ id1 : int [1:43] 2 21 2 2 33 2 38 39 16 6 ...

..\$ id2 : int [1:43] 13 2 27 30 2 37 2 3 6 20 ...

..\$ weight: num [1:43] 0.985 0.985 0.985 0.985 0.985 ...

..\$ color : chr [1:43] "white" "white" "white" "white" ...

\$ type : chr "2D"

\$ desc : chr "Modified Arroyoland Data Set"

\$ axis.cols : chr [1:2] "gray" "gray"

- attr(*, "class")= chr "HivePlotData"

Details

These data sets are [HivePlotData](#) objects. They were created from the datasets Safariland and vazarr in the package bipartite. The process was the same for each: 1. Plants were placed on one axis, pollinators on the other. 2. A radius was assigned by calculating d' using function `dfun` in package bipartite. d' is an index of specialization; higher values mean the plant or pollinator is

more specialized. 3. Edge weights were assigned proportional to the square root of the normalized number of visits of a pollinator to a plant. Thus the width of the edge drawn is an indication of the visitation rate. 4. The number of visits were divided manually into 4 groups and used to assign edge colors ranging from white to red. The redder colors represent greater numbers of visits, and the color-coding is comparable for each data set. Thus both the edge color and the edge weight encode the same information. It would of course be possible to encode an additional variable by changing either edge color or weight. Please e-mail if you would like the script used to convert the data sets.

Source

D. P. Vasquez and D. Simberloff "Changes in interaction biodiversity induced by an introduced ungulate" Ecology Letters, vol. 6 pgs 1077-1083 (2003).

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
data(Safari)
sumHPD(Safari)
plotHive(Safari)
```

cart2sph

Convert spherical to Cartesian coordinates or vice versa

Description

These functions interconvert coordinate systems.

Usage

```
cart2sph(df)
sph2cart(df)
```

Arguments

`df` For `cart2sph`, a data frame with columns named `x`, `y` and `z` containing the Cartesian coordinates to be converted. For `sph2cart`, a data frame with columns named `r`, `theta` and `phi` with the radius and angles (in spherical coordinates) to be converted to Cartesian coordinates.

Value

A data frame with named columns containing the converted coordinates.

Note

Cobbled together from similar functions in other packages.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
df.cart1 <- data.frame(x = rnorm(10), y = rnorm(10), z = rnorm(10))
df.sph <- cart2sph(df.cart1)
str(df.sph)
df.cart2 <- sph2cart(df.sph)
all.equal(df.cart1, df.cart2)
```

chkHPD

Verify the integrity of a hive plot data object

Description

This function inspects the classes of each part of a [HPD](#) as a means of verifying its integrity. A few other characteristics are checked as well.

Usage

```
chkHPD(HPD, confirm = FALSE)
```

Arguments

HPD	An object of S3 class <code>HivePlotData</code> .
confirm	Logical; if TRUE then a favorable result is affirmed in the console (problems are always reported).

Value

A logical value; TRUE if there is a problem, otherwise FALSE.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
test4 <- ranHiveData(nx = 4)
good <- chkHPD(test4, confirm = TRUE)
# mess it up and do again
# next test is not run as it halts execution
## Not run:
test4$nodes$color <- as.factor(test4$nodes$color)
bad <- chkHPD(test4)

## End(Not run)
```

dot2HPD

Process a .dot graph file into a HivePlotData object

Description

This function will read a .dot file containing a graph specification in the DOT language, and using two other files, convert the information into a [HivePlotData](#) object.

Usage

```
dot2HPD(file = NULL, node.inst = NULL, edge.inst = NULL, axis.cols = NULL,
type = "2D", desc = NULL, ...)
```

Arguments

file	The path to the .dot file to be processed.
node.inst	The path to a .csv file containing instructions about how to map node tags in the .dot file to parameters in the HivePlotData object.
edge.inst	The path to a .csv file containing instructions about how to map edge tags in the .dot file to parameters in the HivePlotData object.
axis.cols	A character vector giving the colors desired for the axes.
type	One of c("2D", "3D"). If 2D, a HivePlotData object suitable for use with plotHive will be created and the eventual hive plot will be static and 2D. If 3D, the HivePlotData object will be suitable for a 3D interactive plot using plot3dHive .
desc	Character. A description of the data set.
...	Other parameters to be passed downstream.

Details

This function is currently agnostic with respect to whether or not the .dot graph is directed or not. Either type will be processed, but if the graph is directed, this will only be indirectly stored in the HivePlotData object (in that the first node of an edge in the .dot file will be in `HPD$nodes$id1` and the second node of an edge will be in `HPD$nodes$id2`. This fact can be used; see the vignette and [mineHPD](#).

Value

A [HivePlotData](#) object.

warning

This function currently handles only an extremely limited portion of the .dot standard. Don't expect any particular .dot file to be processed correctly. This will be improved in future versions.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

See Also

See the vignette for an example of using this function. Use `> vignette("HiveR")` to produce the vignette.

[adj2HPD](#) for a means of importing adjacency matrices.

drawHiveSpline	<i>Draw a 3D spline as part of a 3D hive plot</i>
----------------	---

Description

This function analyzes the edges of a `HivePlotData` object in order to draw 3D splines representing those edges. Each pair of nodes at the ends of an edge is identified, and a control point is computed. This information is passed to [rcsr](#) to work out the details.

Usage

```
drawHiveSpline(HPD, L_A = FALSE, ...)
```

Arguments

HPD	An object of S3 class <code>HivePlotData</code> .
L_A	Logical: should splines be drawn with <code>line_antialias = TRUE</code> ?
...	Parameters to be passed downstream.

Value

None. A spline is added to the 3D hive plot in progress.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

See Also

[plot3dHive](#) which calls this function and is the user interface.

HivePlotData

Hive Plot Data Objects

Description

In package HiveR, hive plot data sets are stored as an S3 class called HivePlotData, detailed below.

Structure

The structure of a HivePlotData object is a list of 6 elements, some of which are data frames, and an attribute, as follows:

<i>element</i>	<i>(element)</i>	<i>type</i>	<i>description</i>
\$nodes		data frame	Data frame of node properties
	\$id	int	Node identifier
	\$lab	chr	Node label
	\$axis	int	Axis to which node is assigned
	\$radius	num	Radius (position) of node along the axis
	\$size	num	Node size in pixels
	\$color	chr	Node color
\$edges		data frame	Data frame of edge properties
	\$id1	int	Starting node id
	\$id2	int	Ending node id
	\$weight	num	Width of edge in pixels
	\$color	chr	Edge color
\$type		chr	Type of hive. See Note.
\$desc		chr	Description of data
\$axis.cols		chr	Colors for axes
- attr		chr "HivePlotData"	The S3 class designation.

Note

While \$edges\$id1 and \$edges\$id2 are defined as the starting and ending nodes of a particular edge, hive plots as currently implemented are not directed graphs (agnostic might be a better word).

HPD\$type indicates the type of hive data: If 2D, then the data is intended to be plotted with `hivePlot` which is a 2D plot with axes radially oriented, and (hopefully) no edges that cross axes. If 3D, then the data is intended to be plotted with `plot3dHive` which gives an interactive 3D plot, with axes oriented in 3D.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

See Also

`sumHPD` to summarize a `HivePlotData` object.
`chkHPD` to verify the integrity of a `HivePlotData` object.
`ranHiveData` to generate random `HivePlotData` objects for testing and demonstration.

Examples

```
test4 <- ranHiveData(nx = 4)
str(test4)
sumHPD(test4)
plotHive(test4)
```

manipAxis

Modify the display of axes and nodes in a hive plot

Description

This function modifies the node positions (radii) along each axis in a `HivePlotData` object. A typical use is to convert the radii from the native/absolute values in the original object to either a normalized value (0...1) or to a ranked value. Other modifications can be added relatively easily.

Usage

```
manipAxis(HPD, method, action = NULL)
```

Arguments

HPD	An object of S3 class <code>HivePlotData</code> .
method	One of <code>c("rank", "norm", "scale", "invert", "ranknorm")</code> giving the type of modification to be made.
action	A numeric vector of the same length as the number of axes. Used if <code>method = c("scale", "invert")</code> . For <code>scale</code> node radii will be multiplied by the corresponding value in this argument. For <code>invert</code> , if <code>action = -1</code> , the given axis will be inverted.

Details

The rank method uses `ties.method = "first"` so that each node gets a unique radius.

Value

A modified `HivePlotData` object.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
set.seed(55)
test3 <- ranHiveData(nx = 3)
plotHive(test3)
plotHive(test3, method = "rank")
plotHive(test3, method = "norm")
```

mineHPD

Examine (mine) a HivePlotData object and extract information contained within it

Description

A `HivePlotData` object, especially one created fresh using `dot2HPD`, generally contains a lot of hidden information about the network described. This function can extract this hidden information. This function has options which are quite specific as to what they do. The user can easily write new options and incorporate them (or send them to me for incorporation). This function can be called multiple times using different options to gradually modify the `HivePlotData` object.

Usage

```
mineHPD(HPD, option = "rad <- tot.edge.count")
```

Arguments

HPD A `HivePlotData` object.

option A character string giving the option desired. See Details for current options.

Details

option = "rad <- tot.edge.count" This option looks through the HivePlotData object and determines how many edges start or end on each node (the "degree"). This value is then assigned to the radius for that node.

option = "axis <- source.man.sink" This option examines the nodes and corresponding edges in a HivePlotData object to determine if the node is a source, manager or sink. A source node only has outgoing edges. A sink node only has incoming edges. A manager has both. Hence, this option treats the HivePlotData object as if it were directed in that the first node of an edge will be in `HPD$nodes$id1` and the second node of an edge will be in `HPD$nodes$id2`. As a result, this option produces a hive plot with 3 axes. This concept is similar to the idea of [FuncMap](#) but the internals are quite different. See also [dot2HPD](#) for some details about processing .dot files in an agnostic fashion.

option = "remove orphans" removes nodes that have degree zero (no incoming or outgoing edges).

Value

A modified HivePlotData object.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

See Also

See the vignette for an example of using this function. Use `> vignette("HiveR")` to produce the vignette.

plotHive

Create a hive plot (2D or 3D)

Description

These functions plot a HivePlotData object in either 2D or 3D, depending upon which function is called.

Usage

```
plotHive(HPD, ch = 1, method = "abs", dr.nodes = TRUE, bkgnd = "black",
axLabs = NULL, axLab.pos = NULL, axLab.gpar = NULL,
anNodes = NULL, anNode.gpar = NULL,
arrow = NULL, np = TRUE, ...)
```

```
plot3dHive(HPD, ch = 1, dr.nodes = TRUE, method = "abs",
axLabs = NULL, axLab.pos = NULL, LA = FALSE, ...)
```

Arguments

HPD	An object of S3 class HivePlotData .
ch	Numeric; the size of the central hole in the hive plot.
method	Character. Passed to manipAxis (see there for allowed values).
dr.nodes	Logical; if TRUE nodes will be drawn.
bkgnd	Any valid color specification. Used for the background color for plotHive.
axLabs	A vector of character strings for the axis labels.
axLab.pos	Numeric; An offset from the end of the axis for label placement. Either a single value or a vector of values. If a single value, all labels are offset the same amount. If a vector of values, there should be a value for each axis. This allows flexibility with long axis names. The units depend upon the method employed.
axLab.gpar	A list of name - value pairs acceptable to gpar for use with plotHive only. These control the label and arrow displays. See the examples. Note that if you use this argument the default color for text and arrows becomes black, so if bkgnd = "black" be sure to specify col = "white" (or some other non-black color) or you won't be able to see your labels.
anNodes	The path to a csv file containing information for labeling nodes. If present, a line segment will be drawn from the node to the specified text. The text is positioned near the end of a radial vector. The columns in the csv file must be named as follows (description and use in parentheses): node.lab (node label from HPD\$nodes\$lab), node.text (the text to be drawn on the plot), angle (polar coordinates: angle at which to draw the text), radius (polar coordinates: radius at which to draw the text), offset (additional distance along radius vector to offset text), hjust, vjust (horizontal and vertical justification; nominally 0 or 1 but fractional and negative values also work).
anNode.gpar	A list of name - value pairs acceptable to gpar for use with plotHive only. These control both the text used to annotate the nodes and the line segments connecting that text to the node. See the examples. Note that if you use this argument the default color for text and arrows becomes black, so if bkgnd = "black" be sure to specify col = "white" (or some other non-black color) or you won't be able to see your labels.
np	Logical; should a new device (page) be opened when drawing the hive plot? If you are making multiple plots within some sort of grid manipulations (e.g. a hive plot per the vignette) then this should be set to FALSE.

arrow	A vector of 5 or 6 values: a character string to label the arrow, and 4 numeric values giving the angle of the arrow, the radius at which to start the arrow, the radius at which to end the arrow, and a value to offset the arrow label from the end of the arrow. A 5th numeric value (the 6th argument overall) can specify an offset in the y direction for the arrow when <code>nx = 2</code> . See the examples.
LA	Logical: should splines be drawn with <code>line_antialias = TRUE</code> ? Only applies to <code>type = 3D</code> .
...	Additional parameters to be passed downstream.

Details

plotHive uses grid graphics to produce a 2D hive plot in a style similar to the original concept. For a 2D plot, axis number 1 is vertical except in the case of 2 axes. plot3dHive produces a 3D hive plot using [rgl](#) graphics.

For most work with plot3dHive, use `LA = FALSE` for speed of drawing. `LA = TRUE` is over 20 times slower, and is more appropriate for high quality hive plots. These are probably better made with `R CMD BATCH script.R` use rather than interactive use.

Using `anNodes` gives a lot of flexibility to positioning annotation text, but it may take quite a few iterations if you are perfectionists. Prepare to be patient if you plan to label quite a few nodes. I suggest beginning with `offset`, `hjust` and `vjust` set to 0. There may be some changes to this in the near future.

Value

None. Side effect is a plot.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
test2 <- ranHiveData(nx = 2)
test3 <- ranHiveData(nx = 3)
test4 <- ranHiveData(nx = 4)
#
# plot one with defaults:
plotHive(test3)
#
# Add axis labels & options to nx = 3 example. Note that rot is not part of gpar
require("grid")
plotHive(test3, ch = 5, labs = c("axis 1", "axis 2", "axis 3"),
lab.pos = c(10, 15, 15), lab.gpar = gpar(col = "orange", fontsize = 14),
rot = c(0, 30, -30))
```

```

#
# Now the nx = 2 case.
# Note that gpar contains parameters that apply to both the
# axis labels and arrow. A 6th value in arrow offsets the arrow vertically:
plotHive(test2, ch = 5, labs = c("axis 1", "axis 2"), rot = c(-90, 90),
lab.pos = c(20, 20), lab.gpar = gpar(col = "pink", fontsize = 14, lwd = 2),
arrow = c("radius units", 0, 20, 60, 25, 40))

# The following must be run interactively:
## Not run:
plot3dHive(test4)

## End(Not run)

```

ranHiveData

Generate random hive plot data

Description

This function generates random data sets which can be used to make a hive plot.

Usage

```

ranHiveData(type = "2D", nx = 4, nn = nx * 15, ne = nx * 15, rad = 1:100,
ns = c(0.5, 1.0, 1.5), ew = 1:3, nc = brewer.pal(5, "Set1"),
ec = brewer.pal(5, "Set1"), axis.cols = brewer.pal(nx, "Set1"),
desc = NULL, allow.same = FALSE, verbose = FALSE)

```

Arguments

type	The type of hive plot to be generated. One of c("2D", "3D").
nx	An integer giving the number of axes to be created (2 ≤ nx ≤ 6).
nn	An integer giving the number of nodes to be created. This is an initial number which may be reduced during clean up. See Details.
ne	An integer giving the number of edges to be created. This is an initial number which may be reduced during clean up. See Details.
rad	Numeric; a range of values that will be used as node radius values (the position of the node along the axis).
ns	Numeric; a range of values that will be used as the node sizes.
ew	Numeric; a range of values that will be used as the edge weights.
nc	A vector of valid color names giving the node colors.
ec	A vector of valid color names giving the edge colors.
axis.cols	A vector of valid color names to be used to color the axes; length(axis.cols) must = nx.
desc	Character; a description of the data set.

allow.same	Logical; indicates if edges may begin and end on the same axis. Only applies to type = 2D.
verbose	Logical; If TRUE, the generation, processing and final result is reported to the console.

Details

For type = "2D", after the function creates an initial set of random nodes, these are randomly chosen and connected between adjacent axes, so that no edge crosses an axis.

For type = "3D", after the function creates an initial set of random nodes and edges, these are cleaned up by removing the following cases (which the rest of HiveR is not intended to handle at this time): duplicated nodes, nodes that are not part of any edge, edges that begin and end on the same point, edges that begin and end on the same axis, and finally, for nx = 5 or 6, edges that begin and end on colinear axes. Most of these don't cause an error, but produce some ugly results.

For the arguments rad, ns, ew, nc and ec, the values given are sampled randomly (with replacement) and assigned to particular nodes or edges.

Value

An object of S3 class [HivePlotData](#).

warning

If you create a very small data set with few nodes, there may be no nodes assigned to some axes which will give an error when you try to plot the data. It's up to the user to check for this possibility (you can use `sumHPD`).

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
test4 <- ranHiveData(nx = 4)
str(test4)
sumHPD(test4)
```

`rcsr`*Compute the details of a 3D spline for a hive plot edge*

Description

This is a wild bit of trigonometry! Three points in 3D space, two ends and an control point, are rotated into 2D space. Then a spline curve is computed. This is necessary because spline curves are only defined in R as 2D objects. The new collection of points, which is the complete spline curve and when drawn will be the edge of a hive plot, is rotated back into the original 3D space. `rcsr` stands for rotate, compute spline, rotate back.

Usage

```
rcsr(p0, cp, p1)
```

Arguments

<code>p0</code>	A triple representing one end of the final curve (x, y, z).
<code>cp</code>	A triple representing the control point used to compute the final curve (x, y, z).
<code>p1</code>	A triple representing the other end of the final curve (x, y, z).

Details

See the code for exactly how the function works. Based upon the process described at http://www.fundza.com/mel/axis_to_vector/align_axis_to_vector.html Timing tests show this function is fast and scales linearly (i.e. 10x more splines to draw takes 10x more time). Roughly 3 seconds were required to draw 1,000 spline curves in my testing.

Value

A 3 column matrix with the x, y and z coordinates to be plotted to create a hive plot edge.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
# This is a lengthy example to prove it works.
# Read it and then copy the whole thing to a blank script.
# Parts of it require rgl and are interactive.
# So none of the below is run during package build/check.

### First, a helper function
```

```

## Not run:

drawUnitCoord <- function() {

# Simple function to draw a unit 3D coordinate system

# Draw a Coordinate System

r <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1) # in polar coordinates
theta <- c(0, 0, 0, 90, 0, 180, 0, 270, 0, 0, 0, 0) # start, end, start, end
phi <- c(0, 90, 0, 90, 0, 90, 0, 90, 0, 0, 0, 180)
cs <- data.frame(radius = r, theta, phi)
ax.coord <- sph2cart(cs)

segments3d(ax.coord, col = "gray", line_antialias = TRUE)
points3d(x = 0, y = 0, z = 0, color = "black", size = 4,
point_antialias = TRUE) # plot origin

# Label the axes

r <- c(1.1, 1.1, 1.1, 1.1, 1.1, 1.1) # in polar coordinates
theta <- c(0, 90, 180, 270, 0, 0)
phi <- c(90, 90, 90, 90, 0, 180)
l <- data.frame(radius = r, theta, phi)
lab.coord <- sph2cart(l)
text3d(lab.coord, texts = c("+x", "+y", "-x", "-y", "+z", "-z"))

}

### Now, draw a reference coordinate system and demo the function in it.

drawUnitCoord()

### Draw a bounding box

box <- data.frame(
x = c(1, -1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, -1),
y = c(1, 1, 1, 1, 1, -1, 1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, 1),
z = c(1, 1, 1, -1, 1, 1, -1, -1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, 1, 1, 1, -1))

segments3d(box$x, box$y, box$z, line_antialias = TRUE, col = "red")

### Draw the midlines defining planes

mid <- data.frame(
x = c(0, 0, 0, 0, 0, 0, 0, 0, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1),
y = c(-1, -1, -1, 1, 1, 1, 1, -1, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, 1, 1, 1, 1, -1),
z = c(-1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, 0, 0, 0, 0, 0, 0, 0, 0))

segments3d(mid$x, mid$y, mid$z, line_antialias = TRUE, col = "blue")

### Generate two random points

```

```

p <- runif(6, -1, 1)

# Special case where p1 is on z axis
# Uncomment line below to demo
#p[4:5] <- 0

p0 <- c(p[1], p[2], p[3])
p1 <- c(p[4], p[5], p[6])

### Draw the pts, label them, draw vectors to those pts from origin

segments3d(x = c(0, p[1], 0, p[4]),
y = c(0, p[2], 0, p[5]),
z = c(0, p[3], 0, p[6]),
line_antialias = TRUE, col = "black", lwd = 3)

points3d(x = c(p[1], p[4]),
y = c(p[2], p[5]),
z = c(p[3], p[6]),
point_antialias = TRUE, col = "black", size = 8)

text3d(x = c(p[1], p[4]),
y = c(p[2], p[5]),
z = c(p[3], p[6]),
col = "black", texts = c("p0", "p1"), adj = c(1,1))

### Locate control point
### Compute and draw net vector from origin thru cp
### Connect p0 and p1

s <- p0 + p1
segments3d(x = c(0, s[1]), y = c(0, s[2]), z = c(0, s[3]),
line_antialias = TRUE, col = "grey", lwd = 3)

segments3d(x = c(p[1], p[4]), # connect p0 & p1
y = c(p[2], p[5]),
z = c(p[3], p[6]),
line_antialias = TRUE, col = "grey", lwd = 3)

cp <- 0.6*s # Now for the control point

points3d(x = cp[1], # Plot the control point
y = cp[2],
z = cp[3],
point_antialias = TRUE, col = "black", size = 8)

text3d(x = cp[1], # Label the control point
y = cp[2],
z = cp[3],
texts = c("cp"), col = "black", adj = c(1,1))

### Now ready to work on the spline curve

```

```

n2 <- rcsr(p0, cp, p1) # Compute the spline

lines3d(x = n2[,1], y = n2[,2], z = n2[,3],
line_antialias = TRUE, col = "blue", lwd = 3)

### Ta-Da!!!!

## End(Not run)

```

sumHPD

Summarize a hive plot data object and optionally run some checks

Description

This function summarizes a [HivePlotData](#) object in a convenient form. Optionally, it can run some checks for certain conditions that may be of interest. It can also output a data frame of edges to be drawn, either as a data frame or in a LaTeX ready form, or a data frame of orphaned nodes.

Usage

```

sumHPD(HPD, chk.all = FALSE, chk.sm.pt = FALSE, chk.ax.jump = FALSE,
chk.sm.ax = FALSE, chk.orphan.node = FALSE,
plot.list = FALSE, tex = FALSE, orphan.list = FALSE)

```

Arguments

HPD	An object of S3 class <code>HivePlotData</code> .
chk.all	Logical; should all the checks below be run? See Details.
chk.sm.pt	Logical; should the edges be checked to see if any of them start and end on the same axis with the same radius? See Details.
chk.ax.jump	Logical; should the edges be checked to see if any of them start and end on the same axis with the same radius? See Details.
chk.sm.ax	Logical; should the edges be checked to see if any of them start and end on the same axis?
chk.orphan.node	Logical; should orphan nodes be identified? Orphan nodes have degree 0 (no incoming or outgoing edges).
plot.list	Logical; should a data frame of edges to be drawn be returned?
tex	Logical; should the <code>plot.list</code> be formatted for LaTeX?
orphan.list	Logical; should a data frame of orphaned nodes be returned?

Details

Argument `chk.sm.pt` applies only to hive plots of `type = 2D` and only when edges exist that start and end on the same axis. It checks to see if any of the edges start and end at the same radius on the same axis, which causes an error in `plotHive` since you are trying to draw an edge of length zero (the actual error message is `Error in calcCurveGrob(x, x$debug) : End points must not be identical`). Some data sets may have such cases intrinsically or due to data entry error, or the condition may arise during processing. Either way, one needs to be able to detect such cases for removal or modification. This argument will tell you which nodes cause the problem.

Argument `chk.ax.jump` applies only to hive plots of `type = 2D`. It checks to see if any of the edges jump an axis. This argument will tell you which nodes are at either end of the jumping edge. This should be avoided in hive plots as it makes the plot aesthetically unpleasing. However, depending upon how you process the data, this condition may arise and hence it is useful to be able to locate jumps.

Value

A summary of the `HivePlotData` object's key characteristics is printed at the console, followed by the results of any checks set to `TRUE`. The format of these results is identical to that of `plot.list` described just below, except for the orphan node check. This is formatted the same as `HPD$nodes`; see `?HPD` for details.

If `plot.list = TRUE`, a data frame containing a list of the edges to be drawn in a format suitable for troubleshooting a plot. If `tex = TRUE` as well, the data frame will be in a format suitable for pasting into a LaTeX document. The data frame will contain rows describing each edge to be drawn with the following columns: node 1 id, node 1 axis, node 1 label, node 1 radius, then the same info for node 2, then the edge weight and the edge color.

If `orphan.list = TRUE` a data frame giving the orphan nodes is returned. If you want both `plot.list` and `orphan.list` you have to call this function twice.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

<http://academic.depauw.edu/~hanson/HiveR/HiveR.html>

Examples

```
set.seed(55)
test <- ranHiveData(nx = 4, ne = 5, desc = "Tiny 4D data set")
out <- sumHPD(test, chk.all = TRUE, plot.list = TRUE)
print(out)
```

Index

- *Topic **classes**
 - HivePlotData, 10
 - *Topic **datagen**
 - ranHiveData, 16
 - *Topic **datasets**
 - Arroyo, 5
 - *Topic **dplot**
 - rcsr, 18
 - *Topic **dynamic**
 - plotHive, 13
 - *Topic **hplot**
 - drawHiveSpline, 9
 - plotHive, 13
 - *Topic **package**
 - HiveR-package, 2
 - *Topic **plot**
 - drawHiveSpline, 9
 - plotHive, 13
 - *Topic **utilities**
 - cart2sph, 6
 - chkHPD, 7
 - dot2HPD, 8
 - manipAxis, 11
 - mineHPD, 12
 - rcsr, 18
 - sumHPD, 21
 - *Topic **utility**
 - adj2HPD, 3
- HivePlotData (HivePlotData), 10
- adj2HPD, 3, 9
- Arroyo, 5
- cart2sph, 6
- chkHPD, 7, 11
- dfun, 5
- dot2HPD, 4, 8, 12, 13
- drawHiveSpline, 9
- FuncMap, 13
- gpar, 14
- HivePlotData, 3, 5, 8, 9, 10, 12, 14, 17, 21
- HiveR (HiveR-package), 2
- HiveR-package, 2
- HPD, 7
- HPD (HivePlotData), 10
- manipAxis, 11, 14
- mineHPD, 3, 8, 12
- plot3dHive, 3, 8, 10
- plot3dHive (plotHive), 13
- plotHive, 3, 8, 13
- ranHiveData, 11, 16
- rcsr, 9, 18
- rgl, 15
- Safari (Arroyo), 5
- sph2cart (cart2sph), 6
- sumHPD, 11, 21