Package 'RnavGraph'

October 21, 2014

Type Package

Title Using Graphs as a Navigational Infrastructure.

Version 0.1.7

Date 2014-10-21

Author Adrian R. Waddell and R. Wayne Oldford

Maintainer Adrian Waddell <adrian@waddell.ch>

URL http://www.navgraph.com

Description GUI to explore high dimensional data (including image data) using graphs as navigational infrastructure.

License GPL-2

Suggests PairViz, MASS, hexbin, RDRToolbox, vegan, RnavGraphImageData,png, Rgraphviz

Depends R (>= 2.10.0), methods, graphics, tcltk, graph

Imports scagnostics, rgl

LazyLoad yes

Collate 'AllGenerics.R' 'ng_data.R' 'ng_graph.R' 'ng_path.R' 'ng_image.R' 'SettingClasses.R' 'NavGraphHandler.R' 'GraphDisplay.R' 'Visualization.R' 'Visualization2D.R' 'Viz2D_Axis.R' 'Viz2D_tk2d.R' 'FunctionNavGraph.R' 'graphtools.R' 'navGraph.R' 'scagNav.R' 'zzz.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-21 13:34:43

R topics documented:

RnavGraph-package *Overview RnavGraph package*

Description

GUI to explore high dimensional data using graphs as navigational infrastructure.

This package implements some of the methods described in the Hurley and Oldford paper.

Visualization instructions can be linked to a bullet on the graph. The bullet can be moved along the graph.

With the `tk2d` graphics device, provided by the **RnavGraph** package, one can display images, glyphs, text and glyphs and modify their location smoothly.

Details

The package vignette provides a detailed overview of the **RnavGraph** package. Use `vignette("RnavGraph")`.

Author(s)

Adrian Waddell and R. Wayne Oldford

References

C. B. Hurley and R. W. Oldford. "Graphs as navigational infrastructure for high dimensional data spaces." Computational Statistics (2011).

See Also

`navGraph`

Examples

```
ng.iris <- ng_data(name = "iris", data = iris[,1:4],
shortnames = c('s.L', 's.W', 'p.L', 'p.W'),
group = iris$Species,
labels = substr(iris$Species,1,2))

nav <- navGraph(ng.iris)
```

`closeViz` *Method that gets called when switching away from a graph that is connected with the visualization instructions*

Description

See the vignette for more details.

Usage

```
closeViz(viz,ngEnv)
```

Arguments

`viz`	Object of class NG_Visualization
`ngEnv`	Environment of navGraph session.

Value

Object of NG_Visualization (viz from argument)

Author(s)

Adrian Waddell and Wayne Oldford

See Also

`initializeViz`, `updateViz`

Examples

```
## See vignette
```

`closeViz-methods` *Initialization of a new Display*

Description

If custom visualization class (NG_Visualization) has been defined. This is the method that gets called if navGraph switches away from a graph that links to the visualization class.

See package vignette for more detail.

Methods

```
signature(obj = "NG_Visualization")
```

completegraph *Create a complete graph of class graphAM*

Description

(From Wikipedia) A complete graph is a simple graph in which every pair of distinct vertices is connected by a unique edge.

Usage

```
completegraph(nodeNames)
```

Arguments

nodeNames	Numeric or character string vector.

Value

graphAM object.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, newgraph, linegraph

Examples

```
G1 <- completegraph(1:10)
G2 <- completegraph(LETTERS[1:7])
```

initializeViz *Method that gets called for initializing a display for some visualization instructions*

Description

See the vignette for more details.

Usage

```
initializeViz(viz,ngEnv)
```

Arguments

viz	Object of class NG_Visualization
ngEnv	Environment of navGraph session.

Value

Object of NG_Visualization

Author(s)

Adrian Waddell and Wayne Oldford

See Also

updateViz, closeViz

Examples

```
## See vignette
```

---

initializeViz-methods *Initialization of a new Display*

Description

If custom visualization class (NG_Visualization) has been defined. This is the method that gets called if navGraph switches to a graph that links to the visualization class.

See package vignette for more detail.

Methods

```
signature(obj = "NG_Visualization")
```

---

linegraph *Linegraph of a graph*

Description

(From Wikipedia) The line graph L(G) of an undirected graph G is another graph L(G) that represents the adjacencies between edges of G. By definition, each vertex of L(G) represents an edge of G, and two vertices of L(G) are adjacent if and only if their corresponding edges share a common endpoint ("are adjacent") in G.

Usage

```
linegraph(graph, sep = ":")
```

Arguments

graph	Undirected graph of class graph.
sep	Separates the node names of G in the node names of the new graph L(G).

Value

graphNEL object.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, completegraph, newgraph

Examples

```
G <- completegraph(LETTERS[1:4])
LG <- linegraph(G, sep = "xx")
```

---

linegraph-methods *Linegraph of a graph*

Description

(From Wikipedia) The line graph L(G) of an undirected graph G is another graph L(G) that represents the adjacencies between edges of G. By definition, each vertex of L(G) represents an edge of G, and two vertices of L(G) are adjacent if and only if their corresponding edges share a common endpoint ("are adjacent") in G.

Methods

```
signature(x = "NG_data")
```

---

names-methods *Show the variable names of a NG_data object*

Description

Show the variable names of a NG_data object.

Methods

```
signature(x = "NG_data")
```

names<--methods *Change the names of a NG_data object*

Description

Change the names of a NG_data object. This end effectively changes the names of the data.frame wrapped in the NG_data object.

Methods

```
signature(x = "NG_data")
```

navGraph *Start an navGraph session*

Description

navGraph is an interactive data visualization program that lets the user explore high dimensional data by using graphs as a "road map". That is, navGraph connects a bullet on a graph to some user specified visualization of some data.

The package is discussed in detail in our vignette, see `vignette("RnavGraph")`.

Usage

```
navGraph(data, graph = NULL, viz = NULL, settings = NULL)
```

Arguments

data	a single- or a list of objects generated by the ng_graph function. I.e. objects from the NG_data class. Two special cases are: - either a *navGrah handler* to reinitialize a navGraph session. - or the sting tclreset to delete *tk2d* related settings and data.
graph	a single- or a list of objects generated by the ng_graph function. I.e. objects from the NG_data class.
viz	a single- or a list of visualization instruction objects generated for example by ng_2d or ng_2d_myplot function. I.e. objects from the NG_visualization class.
settings	a list of pailists. See details.

Details

The **RnavGraph** package comes with an extensive package vignette which we encourage to read.

navGraph is capable to handle multiple graphs where each graph can be connected to multiple data displays.

navGraph needs to know the data to be explored, the graph whose node represent some low dimensional visualization of the data and some visualization instructions which connect the graph and data. Data, graph and visualization instruction have their own wrapper class provided by the **RnavGraph** package. See ng_data, ng_graph and for the visualization instructions ng_2d.

There are several shortcuts to start a navGraph session. All of them assume that the nodes of the graphs represent 2d scatterplots and the edges 3d rigid rotation or a 4d transition of one scatterplot into another. The easiest of them is to only pass by a data object created with the ng_data function. For the other ways consult the vignette.

The settings argument modifies the default appearance and interaction properties of the navGraph user interface. It must be a list of named (names are: color, interaction, display and tk2d) lists containing tag=value pairs. To get the possible tag=value options, study the class definitions of the ColorSettings, InteractionSettings, DisplaySettings and Tk2dDisplay classes with the getClassDef function. See the examples section below for an example.

Value

A navGraph handler which can be used to interface the navGraph session via the R prompt. Use ng_get-methods to get data from the function.

Use ng_update to update the navGraph handler with the current state of its associated navGraph session. If you have changed some attributes such as color or size in the tk2d display, you can get the new grouping with ng_get(ng_get(nav,"data"),"group") back, where *nav* is the navGraph handler.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

ng_data, ng_graph, ng_2d, ng_2d_myplot, scagNav, ng_get-methods, ng_set-methods, ng_update, ng_walk, ng_set_color<-, ng_get_color, ng_set_size<-, ng_get_size

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "iris", data = iris[,1:4],
shortnames = c('s.L', 's.W', 'p.L', 'p.W'),
group = iris$Species,
labels = substr(iris$Species,1,2))

## Start navGraph
nav1 <- navGraph(ng.iris)

## navGraph session, data linked to previous session
```

```
nav2 <- navGraph(ng.iris, settings =
   list(color=list(background="steelblue",bullet="blue"),
   interaction=list(NSteps=11)))

## navGraph session, data is not linked
nav3 <- navGraph(ng.iris, settings=list(tk2d=list(linked=FALSE))
```

---

newgraph *Create a graph object of class graph*

Description

Create a graph object of class graphNEL or graphAM. You might also use the graph creating facility provided by the **graph** package.

Usage

```
newgraph(nodeNames, mat, weights = NULL, directed = FALSE, isAdjacency = FALSE, ...)
```

Arguments

nodeNames	Numeric or character string vector.
mat	Either an adjacency matrix or a from to matrix.
weights	Numeric weights for edges. Either in the same order as the from to matrix or as a square matrix, depending what one have chosen for the mat argument.
directed	Logical value for defining a directed or undirected graph.
isAdjacency	If argument mat is adjacency matrix.
...	Currently not needed.

Value

graphNEL or grapAM object.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, completegraph, linegraph

Examples

```
## Using from to matrices
from <- c("A","A","C","C")
to   <- c("B","C","B","D")
ftEmat <- cbind(from,to)

## note how the E node is added
G <- newgraph(nodeNames = LETTERS[1:5], mat = ftEmat)

## say you would like to add weights to the graph
weights <- c(2,1,3,4)
G <- newgraph(nodeNames = LETTERS[1:5], mat = ftEmat, weights = weights)

## newgraph with adjacency matrix
V <- c('s.L', 's.W', 'p.L', 'p.W')
adjM <- matrix(c(0,1,1,0,1,0,1,1,1,1,0,0,0,1,0,0), ncol = 4)
all(adjM == t(adjM)) ## is symmetric (undirected)
G <- newgraph(nodeNames = V, mat= adjM, isAdjacency=TRUE)

## if you use adjacency matrices, you can add a matrix with weights
adjM <-     matrix(c(0,0,1,0,1,0,1,1,0,0,0,0,0,1,0,0), ncol = 4)
weightsM <- matrix(c(0,0,5,0,2,0,1,3,0,0,0,0,0,7,0,0), ncol = 4)
G <- newgraph(nodeNames = V, mat= adjM, weights = weightsM, directed = TRUE, isAdjacency=TRUE)
edgeData(G, attr = "weight")
```

---

ng_2d *Visualization instruction for 2d scatterplots on a tk2d display*

Description

Visualization instruction for a navGraph session that link the nodes of a navigation graph to 2d scatterplots and the edges to a 3d rigid rotation or a 4d transition.

The scatterplots will be displayed on a tk2d display, provided by the **RnavGraph** package. tk2d allows the user to display dots, glyphs, text and images at given x-y positions. Further interactivity such as selection, zooming, brushing, changing colors and changing sizes are provided by the tk2d display. tk2d displays can also link data between different navGraph sessions.

Usage

```
ng_2d(data, graph, images = NULL, glyphs = NULL)
```

Arguments

data	NG_data object.
graph	NG_graph object.
images	NG_image object. Order of the images must match the order of the data in the NG_data object.
glyphs	Vector of character strings matching either the names or shortnames of the NG_data object.

Details

The text labels in the tk2d display are taken from the labels slot in the NG_data object.

It the group slot of the NG_data object contains only colors from the `colors` function, objects get colored accordingly. Otherwise the group data gets mapped to some color key.

Value

NG_Viztk2d object that inherits from the NG_Visualization class.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, ng_data, ng_graph, ng_image_array_gray, ng_image_files, ng_get-methods, ng_set-methods, ng_2d_myplot

Examples

```
## NG_data
ng.iris <- ng_data(name = "iris", data = iris[,1:4],
shortnames = c('s.L', 's.W', 'p.L', 'p.W'),
group = iris$Species,
labels = substr(iris$Species,1,2))

## NG_graph
G <- completegraph(shortnames(ng.iris))
LG <- linegraph(G, sep = "++")
ng.lg <- ng_graph("3d transition", LG, "++", "fruchtermanReingold" )

## NG_image
## see the image demos: demo(package = "RnavGraph")

## Visualization instruction
viz1 <- ng_2d(ng.iris, ng.lg, glyphs = c("s.L","s.W","p.L","p.W"))
viz1
```

---

ng_2d_myplot *Visualization instruction for 2d scatterplots on a user defined display*

Description

Visualization instruction for a `navGraph` session that link the nodes of a navigation graph to 2d scatterplots and the edges to a 3d rigid rotation or a 4d transition.

`navGraph` will call at any bullet state change a function the user defines in the global environment.

The executed function should contain plot instruction and have any subset of these arguments

argument	description
x	x-coordinate
y	y-coordinate
group	group slot from NG_data object
labels	labels slot from NG_data object
order	order of points. In 3d rigid rotation, the order increases with the distance of a the point from the viewer.
from	node name the bullet moves from
to	node name the bullet moves to
percentage	in between percentage of bullet
data	data name of NG_data object

Usage

```
ng_2d_myplot(data, graph, fnName, device = "base", scaled = TRUE)
```

Arguments

`data`	NG_data object.
`graph`	NG_graph object.
`fnName`	Character sting of the function name defined in the global environment `.GlobalEnv`. If a function with the name `fnName.init` also exists, it gets called to initialize the plots.
`device`	One of the following choices: "base", "grid", "ggplot2", "lattice" or "rgl".
`scaled`	Logical. If TRUE, scaled x and y coordinates get passed to the function `fnName` which lie within the rectangle defined by (-1,-1) and (1,1).

Note

Note that the base graphic device (including **grid**, **ggplot2** and the **lattice** device) don't have double buffering implemented in Linux and OSX but in Windows they have. Hence, in Windows you will get smooth plots. But in OSX the plots only flush to the screen after a certain idle time from writing data to the device. In Linux, the device gets updated after every plot command but the user will experience a white flickering from redrawing the plot.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, ng_data, ng_graph, ng_get-methods, ng_set-methods, ng_2d

Examples

```
library(grid)

## NG_data
ng.iris <- ng_data(name = "iris", data = iris[,1:4],
shortnames = c('s.L', 's.W', 'p.L', 'p.W'),
```

```
group = iris$Species,
labels = substr(iris$Species,1,2))

## NG_graph
V <- shortnames(ng.iris)
G <- completegraph(V)
LG <- linegraph(G)
LGnot <- complement(LG)
ng.lg <- ng_graph(name = '3D Transition', graph = LG, layout = 'circle')
ng.lgnot <- ng_graph(name = '4D Transition', graph = LGnot, layout = 'circle')

## initialization plot
myPlot.init <- function(x,y,group,labels,order) {
pushViewport(plotViewport(c(5,4,2,2)))
pushViewport(dataViewport(c(-1,1),c(-1,1),name="plotRegion"))

grid.points(x,y, name = "dataSymbols")
grid.rect()
grid.xaxis()
grid.yaxis()
grid.edit("dataSymbols", pch = 19)
grid.edit("dataSymbols", gp = gpar(col = group))
}

## update plot
myPlot <- function(x,y,group,labels,order) {
  print(order)
grid.edit("dataSymbols",
           x = unit(x,"native"),
           y = unit(y,"native"))
      }

## Visualization instruction
viz1 <- ng_2d_myplot(ng.iris,ng.lg,fnName = "myPlot",
           device = "grid", scaled=TRUE)

## navGraph session
nav <- navGraph(ng.iris,ng.lg, viz1)
```

---

ng_data *Create an NG_data object to be used by a navGraph session*

Description

NG_data objects wrap the data to be explored and some meta data such as the data name, variable short names, the group identifier and text labels for each observation.

Usage

```
ng_data(name, data, shortnames = character(0), group = numeric(0), labels = character(0))
```

Arguments

name	Character string containing no spaces. Note that for a single navGraph session, the names of all the NG_data objects passed by to navGraph must be unique.
data	data.frame with only numeric variables.
shortnames	If the variable names of the data argument are too long or contain spaces, shortnames can be specified.
group	Vector of group identifiers for each observation. For the tk2d display, one can use color names (see colors) to map a certain color directly to the objects.
labels	Factor- or character sting vector with labels for each observation.

Value

NG_data object.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, shortnames-methods, ng_get-methods, ng_set-methods

Examples

```
## minimal example
ng.iris <- ng_data(name = "iris", data = iris[,1:4])
ng.iris ## see output


## full specification
ng.iris <- ng_data(name = "iris", data = iris[,1:4],
shortnames = c('s.L', 's.W', 'p.L', 'p.W'),
group = iris$Species,
labels = substr(iris$Species,1,2))

## see shortnames
shortnames(ng.iris)
## change shortnames
shortnames(ng.iris) <- c("a","b","c","d")
## see variable names
names(ng.iris)
## change variable names
names(ng.iris) <- LETTERS[1:4]

## ng_get
ng_get(ng.iris)
ng_get(ng.iris,"group")

## ng_set
```

```
ng_set(ng.iris)
ng_set(ng.iris,"group") <- iris$Species
```

---

ng_get *Extract data from a NG_data, NG_graph, NG_path or navgraph handler object.*

Description

Extract data from objects from some of the in RnavGraph specifically defined classes.

Usage

```
ng_get(obj, what = NULL, ...)
```

Arguments

obj	Either a: navgraph handler, NG_data, NG_path, NG_graph object.
what	String of what should be extracted from the object. You can get a list of possible accessible objects by not specifying the what argument.
...	Currently not used.

Value

Object that was requested with what.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_get-methods, ng_set, ng_set-methods, ng_set_color<-, ng_get_color, ng_set_size<-, ng_get_size

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "iris", data = iris[,1:4])

## start a navGraph session
nav <- navGraph(ng.iris)

## See what you can extract from the navGraph handler
ng_get(nav)

## get group
ng_get(ng_get(nav,"data"),"group")
```

ng_get-methods *Access data from a NG_... object*

Description

Access data in the wrapper classes in navGraph or the navGraph hanlder.

The usage is either ng_get(object) or ng_get(object,"what").

See in the examples of ng_update.

Methods

```
signature(obj = "NavGraph_handler")
signature(obj = "NG_data")
signature(obj = "NG_graph")
signature(obj = "NG_path")
```

ng_get_color *Extract point coloring from an active navGraph session for a particular data set.*

Description

If you use the brush and coloring tool in navGraph, you can retrieve your coloring information with this function as long the navGraph session is still active.

Usage

```
ng_get_color(obj, dataName)
```

Arguments

obj	A navgraph handler of an active navGraph session.
dataName	String of the data name. If not specified and only one data set is beeing used in the navGraph session, it will default to this data. Otherwise, if multiple data sets are being used in a navGraph session, the function will list the name of these data sets and ask you to specify one.

Value

Vector with color values for each point.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_set_color<-, ng_get_size, ng_set_size<-

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "IrisData", data = iris[,1:4])

## start a navGraph session
nav <- navGraph(ng.iris)

## get colors from active navGraph session
cols <- ng_get_color(nav, "IrisData")
```

`ng_get_size` *Extract point sizes from an active navGraph session for a particular data set.*

Description

If you use the brush and resisizing tool in navGraph, you can retrieve your point sizes information with this function as long the navGraph session is still active.

Usage

```
ng_get_size(obj, dataName)
```

Arguments

`obj`	A navgraph handler of an active navGraph session.
`dataName`	String of the data name. If not specified and only one data set is beeing used in the navGraph session, it will default to this data. Otherwise, if multiple data sets are being used in a navGraph session, the function will list the name of these data sets and ask you to specify one.

Value

Nummerical vector with size values for each point.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_set_size<-, ng_get_color, ng_set_color<-

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "IrisData", data = iris[,1:4])

## start a navGraph session
nav <- navGraph(ng.iris)

## get sizes from active navGraph session
sizes <- ng_get_size(nav, "IrisData")
```

---

ng_graph *Create an NG_data object to be used by a navGraph session*

Description

NG_graph objects wrap the transition graphs and additional information such as graph name and graph layout.

The node names of the graph split by the chosen sep character sting must match either the variable names- or the short names of the NG_data object.

Usage

```
ng_graph(name, graph, sep = ":", layout = "circle")
```

Arguments

name	Name of graph. This name will be displayed in the pull down menu within a navGraph session.
graph	Undirected graph objects of class graph. See the package **graph**.
sep	Node names represent a set of variables whose name are separated by the character string sep (containing no spaces).
layout	One of the following strings: "circle", "kamadaKawaiSpring", "fruchtermanReingold" or "random".

Details

The number of sep occurrences in each node name of the graph must be the same.

Value

NG_graph object.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, newgraph, completegraph, linegraph, complement, ng_get-methods, ng_set-methods

Examples

```
G <- completegraph(LETTERS[1:4])
LG <- linegraph(G, sep = "++")

ng.lg <- ng_graph("3d transition", LG, "++", "fruchtermanReingold" )
plot(ng.lg)

## If you have the Rgraphviz package working, plot graph object of class graph
## Not run: library(Rgraphviz)
## Not run: plot(LG)
```

---

ng_image_array_gray *Convert a matrix or data.frame of image data into a NG_image object.*

Description

NG_image objects are needed to plot images in the tk2d device.

ng_image_array_gray imports gray scale images that are saved in a matrix structure, that is every row or column contains the pixel data of an image.

See the "alpha_letter", "digits", "faces", "frey" and "umist_faces" demos (demo(package="RnavGraph")) for examples.

Usage

```
ng_image_array_gray(name, imageData,
                    width, height,
                    img_in_row = TRUE, invert = FALSE,
                    rotate = 0)
```

Arguments

name	Character string.
imageData	data.frame of matrix with image data. Image information must be stored as a gray scale value (0-255) within a single row or column.
width	Pixel width of image.
height	Pixel height of image.
img_in_row	Logical. Single image stored in a row or column in imageData.
invert	For invert=FALSE 0 = whit, for invert=TRUE 0=black.
rotate	Rotate the image: 0, 90, 180 or 270 degree.

Value

NG_image object

Author(s)

Adrian Waddell and R. Wayne oldford

See Also

navGraph, ng_image_files, ng_2d

Examples

```
Img <- matrix(c(0,0,0,255,255,255,
                0,0,0,255,255,255,
                0,0,0,255,255,255,
                255,255,255,0,0,0,
                255,255,255,0,0,0,
                255,255,255,0,0,0),
              byrow = TRUE, ncol=6)

ng.img <- ng_image_array_gray("Test",
                cbind(as.vector(Img),as.vector(Img)),6,6,FALSE)
ng.img

## See demos and vignette
```

---

ng_image_files *Import color image files (png, jpg, bmp, etc...) into a NG_image object*

Description

NG_image objects are needed to plot images in the tk2d device.

Supported file types are those supported by the tcl Img extension. If your R can not access the Img extension (currently under Windows) you can not import such files. However if you know tcl and R well enough, you might create your own NG_image object as shown in the images_iris demo.

See the "files_aloi" and "images_iris" demos (`demo(package="RnavGraph")`) for examples.

Usage

```
ng_image_files(name, paths)
```

Arguments

name	Character string.
paths	A vector of image paths.

Value

NG_image object.

Author(s)

Adrian Waddell and R. Wayne Oldford.

See Also

navGraph, ng_image_array_gray, ng_2d

Examples

```
## See demos and vignette
```

`ng_set` *Shows which data can be modified in a NG_data, NG_graph, NG_path or navgraph handler object.*

Description

Show what can be changed in objects from some of the in RnavGraph specifically defined classes.

Usage

```
ng_set(object)
```

Arguments

`object`	Either a: navgraph handler, NG_data, NG_path, NG_graph object.

Value

No return value. Only a string gets printed onto the comand prompt of all possible data that can be modified within the object.

Note

Using ng_set is risky because we don't guarantee that the objects gets updated correctly. We recommend to just re-create the object.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_set-methods, ng_get, ng_get-methods, ng_set_color<-, ng_get_color, ng_set_size<-, ng_get_size

ng_set-methods *Change data in NG_... objects*

Description

Care must be taken when using the ng_set method, as one might create inconsistency in some objects.

Its use is ng_set(object) to see what you can modify and ng_set(object,"what") <- .... See ng_set<--methods.

Methods

signature(object = "NG_data")

signature(object = "NG_graph")

ng_set<- *Modify data in a NG_data, NG_graph, NG_path or navgraph handler object.*

Description

Modifies data in some of the in RnavGraph specifically defined classes.

Usage

```
ng_set(object, what) <- value
```

Arguments

object	Either a: navgraph handler, NG_data, NG_path, NG_graph object.
what	String of what should be modified within the object.
value	Replacement value.

Details

Using ng_set is risky because we don't guarantee that the objects gets updated correctly. We recommend to just re-create the object.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_set-methods, ng_get, ng_get-methods , ng_set_color<-, ng_get_color, ng_set_size<-, ng_get_size

ng_set<--methods *~~ Methods for Function ng_set<- ~~*

Description

~~ Methods for function ng_set<- ~~

Methods

```
signature(object = "NavGraph_handler")
signature(object = "NG_data")
signature(object = "NG_graph")
```

ng_set_color<- *Change colors of data points in an active navGraph session*

Description

Specify new colors for each point for an active navGraph session.

Usage

```
ng_set_color(obj, dataName) <- value
```

Arguments

obj	A navgraph handler of an active navGraph session.
dataName	String of the data name. If not specified and only one data set is beeing used in the navGraph session, it will default to this data. Otherwise, if multiple data sets are being used in a navGraph session, the function will list the name of these data sets and ask you to specify one.
value	Replacement vector or single value specifing valid colors.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_get_color, ng_set_size<-, ng_get_size

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "IrisData", data = iris[,1:4])

## start a navGraph session
nav <- navGraph(ng.iris)

## set all point to red
ng_set_color(nav, "IrisData") <- 'red'
```

`ng_set_color<--methods` *Change colors of data points in an active navGraph session*

Description

Specify new colors for each point for an active navGraph session.

Methods

```
signature(obj = "NavGraph_handler")
```

`ng_set_size<-` *Change sizes of data points in an active navGraph session*

Description

Specify new sizes for each point for an active navGraph session.

Usage

```
ng_set_size(obj, dataName) <- value
```

Arguments

`obj`	A navgraph handler of an active navGraph session.
`dataName`	String of the data name. If not specified and only one data set is beeing used in the navGraph session, it will default to this data. Otherwise, if multiple data sets are being used in a navGraph session, the function will list the name of these data sets and ask you to specify one.
`value`	Replacement vector or single value specifing the sizes (>0) of points.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_get_size, ng_set_color<-, ng_get_color

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "IrisData", data = iris[,1:4])

## start a navGraph session
nav <- navGraph(ng.iris)

## set point sizes
ng_set_size(nav,'IrisData') <- sample(1:7, replace=TRUE, 150)
```

---

ng_set_size<--methods *Change sizes of data points in an active navGraph session*

Description

Specify new sizes for each point for an active navGraph session.

Methods

signature(obj = "NavGraph_handler")

---

ng_update *Synchronize a navGraph handler with a running navGraph sesson*

Description

The navGraph handler can be used to interact with a running navGraph session via the R prompt.

For the tk2d display one can change color and size of the data points. You can retrieve this information by updating the navGraph handler and reading the group attribute of the data object back. See the examples.

Usage

```
ng_update(nghandler)
```

Arguments

nghandler	navGraph handler of a running navGraph session.

Value

updated navGraph handler.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, ng_walk, ng_get-methods

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "iris", data = iris[,1:4])

## start a navGraph session
nav <- navGraph(ng.iris)

## modify colors of points

## update navGraph handler
nav <- ng_update(nav)
nav # show method

## get information form navGraph handler
ng_get(nav)

## get group attribute back
ng_get(ng_get(nav,"data"),"group")
```

NG_Visualization-class

Class "NG_Visualization"

Description

Used when custom visualization instructions are defined using initializeViz, updateViz and closeViz.

See package vignette for more details.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

graph: Graph name in slot name of NG_graph object.

data: Data name in slot name of NG_data object.

from: Start from node. Not used anymore.

to: Start to node. Not used anymore.

varList: Vector with all variable names that are used in graph.

Methods

show `signature(object = "NG_Visualization")`: ...

Note

Read more about the use of this class in the package vignette.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

`initializeViz-methods`

Examples

```
showClass("NG_Visualization")
```

---

ng_walk *Walk a path on the current graph in a navGraph session*

Description

If you create a path outside the navGraph session, you can let navGraph walk you path.

A path has to be a sequence of node names of adjoining nodes in the current shown graph.

If a path has been walked through, it gets added to the `activePath` in the path tool of the running navGraph session.

Usage

```
ng_walk(nghandler, path)
```

Arguments

`nghandler`	navGraph handler of a running navGraph session.
`path`	Vector of node names that are adjoining in the current graph. A single character string with the node names separated by a space also works.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

`navGraph`

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "iris", data = iris[,1:4],
shortnames = c('s.L', 's.W', 'p.L', 'p.W'))

## start a navGraph session
nav <- navGraph(ng.iris)

## Find linegraph
LG <- linegraph(completegraph(shortnames(ng.iris)))

## find a path
library(PairViz)
path = eulerian(LG)

## walk the path
ng_walk(nav,path)
```

`olive` *Fatty Acid Composition of Italian Olive Oils*

Description

This data set records the percentage composition of 8 fatty acids (palmitic, palmitoleic, stearic, oleic, linoleic, linolenic, arachidic, eicosenoic) found in the lipid fraction of 572 Italian olive oils. The oils are samples taken from three Italian regions varying number of areas within each region. The regions and their areas are recorded as shown in the following table:

Region	**Area**
North	North-Apulia, South-Apulia, Calabria, Sicily
South	East-Liguria, West-Liguria, Umbria
Sardinia	Coastal-Sardinia, Inland-Sardinia

Usage

```
olive
```

Format

A data frame containing 572 cases and 10 variates.

References

Forina, M., Armanino, C., Lanteri, S., and Tiscornia, E. (1983) "Classification of Olive Oils from their Fatty Acid Composition", in *Food Research and Data Analysis* (Martens, H., Russwurm, H., eds.), p. 189, Applied Science Publ., Barking.

plot-methods *Plot the graph saved in a NG_data object.*

Description

Once the graph is within a NG_graph object, it has a layout associated. Plot plots the graph with this layout.

Methods

```
signature(x = "NG_data")
```

scagEdgeWeights *Create a from-to edge matrix with the scagnostic weights*

Description

Create a from-to edge matrix with the scagnostic weights.

Usage

```
scagEdgeWeights(data,
                scags = c("Clumpy", "NotClumpy", "Monotonic", "NotMonotonic",
                    "Convex", "NotConvex", "Stringy", "NotStringy",
                    "Skinny", "NotSkinny", "Outlying","NotOutlying",
                    "Sparse", "NotSparse", "Striated", "NotStriated",
                    "Skewed", "NotSkewed"),
                combineFn = NULL)
```

Arguments

data	object to calculate scagnostics on: NG_data, a vector, a matrix or a data.frame.
scags	Single element or a subset of (with possible a "Not" preceding): "Outlying", "Skewed", "Clumpy", "Sparse", "Striated", "Convex", "Skinny", "Stringy", "Monotonic"
combineFn	Must be a function that takes in a vector of length scags and returns a single value. This return value comprises the new weights of the nodes get selected from.

Value

a named list with fromToEdgeMatrix being a matrix and nodeNames being a vector.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, scagNav, scagGraph

Examples

```
data(olive)
ng.olive <- ng_data(name = "Olive",
data = olive[,-c(1,2)],
shortnames = c("p1","p2","s","ol","l1","l2","a","e"),
group = as.numeric(olive$Area)+1
)

edgeWts <- scagEdgeWeights(data = ng.olive,
scags = c("Clumpy", "Skinny"),
combineFn = max)
edgeWts$fromToEdgeMatrix[1:3,]


edgeWts <- scagEdgeWeights(data = ng.olive,
scags = c("Clumpy", "Skinny"),
combineFn = function(x){
2*x[1]+3*x[2]
})
edgeWts$fromToEdgeMatrix[1:3,]
```

---

scagGraph *Create a list of graphs, given the scagnostics edge weights*

Description

scagGraph is useful to turn the output of scagEdgeWeights into a list of graphs.

Usage

```
scagGraph(edgeWeights, topFrac = 0.2)
```

Arguments

edgeWeights	List returned by the scagEdgeWeights function.
topFrac	Keep the nodes with the topFrac fraction of the scagnostic weights.

Value

graph object or a list of graph objects.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, scagNav, scagEdgeWeights

Examples

```
data(olive)
ng.olive <- ng_data(name = "Olive",
data = olive[,-c(1,2)],
shortnames = c("p1","p2","s","ol","l1","l2","a","e"),
group = as.numeric(olive$Area)+1
)

edgeWts <- scagEdgeWeights(data = ng.olive,
scags = c("Clumpy", "Skinny"))
G1 <- scagGraph(edgeWts, topFrac = 0.2)

edgeWts <- scagEdgeWeights(data = ng.olive,
scags = c("Clumpy", "Skinny"),
combineFn = max)
G2 <- scagGraph(edgeWts, topFrac = 0.1)

## Now you can start a navGraph session
nav <- navGraph(ng.olive, G1)

nav <- navGraph(ng.olive, G2)
```

---

scagNav *Start a navGraph session and filter the navigation graph's node according to some scagnostic measures*

Description

Scanotstics assigns each possible scatterplot combination a weight for the attribute "Outlying", "Skewed", "Clumpy", "Sparse", "Striated", "Convex", "Skinny", "Stringy", "Monotonic".

With scagNav one can start a navGraph session that constructs a navigation graph that shows certain properties most.

See the vignette for more examples.

Usage

```
scagNav(data,
  scags = c("Clumpy", "NotClumpy", "Monotonic", "NotMonotonic",
"Convex", "NotConvex", "Stringy", "NotStringy",
"Skinny", "NotSkinny", "Outlying","NotOutlying",
"Sparse", "NotSparse", "Striated", "NotStriated",
"Skewed", "NotSkewed"),
        topFrac = 0.2, combineFn = NULL,
        settings = NULL, glyphs = NULL,
```

```
          images = NULL, sep = ":",
          layout = "circle")
```

Arguments

data	a single- or a list of objects generated by the ng_graph function. I.e. objects from the NG_data class.
scags	Single element or a subset of (with possible a "Not" preceding): "Outlying", "Skewed", "Clumpy", "Sparse", "Striated", "Convex", "Skinny", "Stringy", "Monotonic"
topFrac	Keep the nodes with the topFrac fraction of the scagnostic weights.
combineFn	Must be a function that takes in a vector of length scags and returns a single value. This return value comprises the new weights of the nodes get selected from.
settings	a list of pailists. See the navGraph documentation.
glyphs	Vector of character strings matching either the names or shortnames of the NG_data object.
images	NG_image object. Order of the images must match the order of the data in the NG_data object.
sep	Node names represent a set of variables whose name are separated by the character string sep (containing no spaces).
layout	One of the following strings: "circle", "kamadaKawaiSpring", "fruchterman-Reingold" or "random".

Value

navGraph handler.

Author(s)

Adrian Waddell and R. Wayne Oldford

See Also

navGraph, scagEdgeWeights, scagGraph, ng_data

Examples

```
## Define a NG_data object
data(olive)
ng.olive <- ng_data(name = "Olive",
data = olive[,-c(1,2)],
shortnames = c("p1","p2","s","ol","l1","l2","a","e"),
group = as.numeric(olive$Area)+1
)

nav <- scagNav(data = ng.olive,
scags = c("Skinny", "Sparse", "NotConvex"),
```

```
topFrac = 0.2,
combineFn = max,
glyphs = shortnames(ng.olive)[1:8],
sep = ':')

nav <- scagNav(data = ng.olive,
scags = c("Skinny", "Sparse", "NotConvex"),
topFrac = 0.2,
glyphs = shortnames(ng.olive)[1:8],
sep = ':')
```

---

shortnames *Returns the shortnames of a NG_data object.*

Description

Shortnames are useful to reduce the node names of navigation graphs. The node names of navigation graphs are either linked with the names or the shortnames of the NG_data object.

Usage

```
shortnames(x)
```

Arguments

x	NG_data object.

Value

Vector of strings.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

navGraph, ng_data

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "iris", data = iris[,1:4])

## Display shortnames
shortnames(ng.iris) ## no shortnames defined yet

## Modify the shortnames
shortnames(ng.iris) <- c("sL","sW","pL","pW")
```

```
## Display updated shortnames
shortnames(ng.iris) ## no shortnames defined yet
```

`shortnames-methods` *Show the shortnames of a NG_data object*

Description

The variable names coded in the node names of a graph in a `navGraph` session will be matched either with the names or shortnames of a NG_data object.

Methods

```
signature(x = "NG_data")
```

`shortnames<-` *Modify the shortnames of a NG_data object.*

Description

Shortnames are useful to reduce the node names of navigation graphs. The node names of navigation graphs are either linked with the names or the shortnames of the NG_data object.

Usage

```
shortnames(x) <- value
```

Arguments

`x`	NG_data object.
`value`	Vector of stings.

Author(s)

Adrian Waddell and Wayne Oldford

See Also

`navGraph`, `ng_data`

Examples

```
## Define a NG_data object
ng.iris <- ng_data(name = "iris", data = iris[,1:4])

## Display shortnames
shortnames(ng.iris) ## no shortnames defined yet

## Modify the shortnames
shortnames(ng.iris) <- c("sL","sW","pL","pW")

## Display updated shortnames
shortnames(ng.iris) ## no shortnames defined yet
```

`shortnames<--methods` *Change the shortnames of a NG_data object*

Description

Change the shortnames of a NG_data object.

Note that the shortnames of a NG_data object can not contain any spaces.

Methods

```
signature(x = "NG_data")
```

`updateViz` *Method that gets called when bullet state is changed on a graph that is connected with the particular visualization instructions*

Description

See the vignette for more details.

Usage

```
updateViz(viz,ngEnv)
```

Arguments

`viz`	Object of class NG_Visualization
`ngEnv`	Environment of navGraph session.

Value

Object of NG_Visualization (viz from argument)

Author(s)

Adrian Waddell and Wayne Oldford

See Also

initializeViz, closeViz

Examples

```
## See vignette
```

---

updateViz-methods *Initialization of a new Display*

Description

If custom visualization class (NG_Visualization) has been defined. This is the method that gets called if the bullet changes its state.

See package vignette for more detail.

Methods

```
signature(obj = "NG_Visualization")
```

Index