hexbin

October 25, 2011

`NHANES` *NHANES Data : National Health and Nutrition Examination Survey*

Description

This is a somewhat large interesting dataset, a data frame of 15 variables (columns) on 9575 persons (rows).

Usage

```
data(NHANES)
```

Format

This data frame contains the following columns:

Cancer.Incidence binary factor with levels `No` and `Yes`.

Cancer.Death binary factor with levels `No` and `Yes`.

Age numeric vector giving age of the person in years.

Smoke a factor with levels `Current`, `Past`, `Nonsmoker`, and `Unknown`.

Ed numeric vector of $\{0, 1\}$ codes giving the education level.

Race numeric vector of $\{0, 1\}$ codes giving the person's race.

Weight numeric vector giving the weight in kilograms

BMI numeric vector giving Body Mass Index, i.e., `Weight/Height^2` where Height is in meters, and missings (61% !) are coded as `0` originally.

Diet.Iron numeric giving Dietary iron.

Albumin numeric giving albumin level in g/l.

Serum.Iron numeric giving Serum iron in μg/l.

TIBC numeric giving Total Iron Binding Capacity in μg/l.

Transferin numeric giving Transferin Saturation which is just `100*serum.iron/TIBC`.

Hemoglobin numeric giving Hemoglobin level.

Sex a factor with levels `F` (female) and `M` (male).

Source

unknown

Examples

```
data(NHANES)
summary(NHANES)
## Missing Data overview :
nNA <- sapply(NHANES, function(x)sum(is.na(x)))
cbind(nNA[nNA > 0])
# Which are just these 6 :
## Not run:
Diet.Iron         141
Albumin           252
Serum.Iron       1008
TIBC              853
Transferin       1019
Hemoglobin        759

## End(Not run)
```

ColorRamps *Color Ramps on Perceptually Linear Scales*

Description

Functions for returning colors on perceptually linear scales, where steps correspond to 'just detectable differences'.

Usage

```
LinGray (n, beg=1, end=92)
BTC     (n, beg=1, end=256)
LinOCS  (n, beg=1, end=256)
heat.ob (n, beg=1, end=256)
magent  (n, beg=1, end=256)
plinrain(n, beg=1, end=256)
```

Arguments

n	number of colors to return from the ramp
beg	begining of ramp, integer from 1-255
end	end of ramp, integer from 1-255

Details

Several precalulated color ramps, that are on a perceptually linear color scale. A perceptually linear color scale is a scale where each jump corresponds to a "just detectable difference" in color and the scale is percieved as linear by the human eye (emprically determined).

When using the ramps, if beg is less than end the ramp will be reversed.

Value

returns an array of colors

Author(s)

Nicholas Lewin-Koh

References

Haim Levkowitz (1997) *Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications*. Kluwer Academic Publishers, Boston/London/Dordrecht. http://www.cs.uml.edu/~haim/ColorCenter/

See Also

rainbow, terrain.colors, rgb, hsv

Examples

```
h <- hexbin(rnorm(10000),rnorm(10000))
plot(h, colramp= BTY)
## looks better if you shave the tails:
plot(h, colramp= function(n){LinOCS(n,beg=15,end=225)})
```

---

erode.hexbin *Erosion of a Hexagon Count Image*

Description

This erosion algorithm removes counts from hexagon cells at a rate proportional to the cells' exposed surface area. When a cell becomes empty, algorithm removes the emptied cell and notes the removal order. Cell removal increases the exposure of any neighboring cells. The last cell removed is a type of bivariate median.

Usage

```
erode(hbin, cdfcut = 0.5)
erode.hexbin(hbin, cdfcut = 0.5)
```

Arguments

hbin	an object of class hexbin.
cdfcut	number in (0,1) indicating the confidence level for the limits.

Details

The algorithm extracts high count cells with containing a given fraction (cdfcut) of the total counts. The algorithm extracts all cells if cdfcut=0. The algorithm performs gray-level erosion on the extracted cells. Each erosion cycle removes counts from cells. The counts removed for each cell are a multiple of the cell's exposed-face count. The algorithm choses the multiple so at least one cell will be empty or have a count deficit on each erosion cycle. The erode vector contain an erosion number for each cell. The value of erode is

6*erosion_cycle_ at_ cell_ removal - cell_deficit_at_removal

Cells with low values are eroded first. The cell with the highest erosion number is a candidate bivariate median. A few ties in erode are common.

Value

An `"erodebin"` object (with all the slots from `hbin`) and additionally with high count cells and a component `erode` that gives the erosion order.

See Also

`hexbin`, `smooth.hexbin`, `hcell2xy`,

`gplot.hexbin`, `grid.hexagons`, `grid.hexlegend`

Examples

```
set.seed(153)
x <- rnorm(10000)
y <- rnorm(10000)
bin <- hexbin(x,y)

smbin  <- smooth.hexbin(bin)
erodebin <- erode.hexbin(smbin, cdfcut=.5)
plot(erodebin)

## bivariate boxplot
hboxplot(erodebin, main = "hboxplot(erodebin)")



# show erosion order
plot(bin,style= "lat", minarea=1, maxarea=1,
     legend=FALSE, border=gray(.7))


grid.hexagons(erodebin,style= "lat", minarea=1, maxarea=1,pen="green")
xy <- hcell2xy(erodebin)
grid.text(lab = as.character(erodebin@erode), xy$x, xy$y,
          gp = gpar(col="white", cex=0.65))
```

getHMedian *Get coordiantes of the median cell after the erode operation*

Description

A method for a eroded hexbin object to extract the coordinates of the median cell. The median is simply the cell with the highest erosion number or the last cell to be eroded.

Usage

```
getHMedian(ebin)
```

Arguments

ebin	result of erode.hexbin().

Methods

ebin = "erodebin" ...

See Also

erode.hexbin

Examples

```
set.seed(153)
x <- rnorm(10000)
y <- rnorm(10000)
bin <- hexbin(x,y)

smbin  <- smooth.hexbin(bin)
erodebin <- erode.hexbin(smbin, cdfcut=.5)
getHMedian(erodebin)
```

gplot.hexbin *Plotting Hexagon Cells with a Legend*

Description

Plots Hexagons visualizing the counts in an hexbin object. Different styles are availables. Provides a legend indicating the count representations.

Usage

```
gplot.hexbin(x, style = "colorscale", legend = 1.2, lcex = 1,
      minarea = 0.04, maxarea = 0.8, mincnt = 1, maxcnt = max(x@count),
      trans = NULL, inv = NULL, colorcut = seq(0, 1, length = min(17, maxcnt)),
      border = NULL, density = NULL, pen = NULL,
      colramp = function(n) LinGray(n,beg = 90,end = 15),
      xlab = "", ylab = "", main = "", newpage = TRUE,
      type = c("p", "l", "n"), xaxt = c("s", "n"), yaxt = c("s", "n"),
      clip = "on", verbose = getOption("verbose"))



## S4 method for signature 'hexbin,missing'
plot(x, style = "colorscale", legend = 1.2, lcex = 1,
      minarea = 0.04, maxarea = 0.8, mincnt = 1, maxcnt = max(x@count),
      trans = NULL, inv = NULL, colorcut = seq(0, 1, length = min(17, maxcnt)),
      border = NULL, density = NULL, pen = NULL,
      colramp = function(n) LinGray(n,beg = 90,end = 15),
      xlab = "", ylab = "", main = "", newpage = TRUE,
      type = c("p", "l", "n"), xaxt = c("s", "n"), yaxt = c("s", "n"),
      clip = "on", verbose = getOption("verbose"))
```

Arguments

x	an object of class hexbin.
style	string specifying the style of hexagon plot, see grid.hexagons for the possibilities.
legend	numeric width of the legend in inches of FALSE. In the latter case, or when 0, no legend is not produced.
lcex	characters expansion size for the text in the legend
minarea	fraction of cell area for the lowest count
maxarea	fraction of the cell area for the largest count
mincnt	cells with fewer counts are ignored.
maxcnt	cells with more counts are ignored.
trans	function specifying a transformation for the counts such as sqrt.
inv	the inverse transformation of trans.
colorcut	vector of values covering [0, 1] that determine hexagon color class boundaries and hexagon legend size boundaries. Alternatively, an integer (<= maxcnt) specifying the *number* of equispaced colorcut values in [0,1].
border, density, pen	color for polygon borders and filling of each hexagon drawn, passed to grid.hexagons.
colramp	function accepting an integer n as an argument and returning n colors.
xlab, ylab	x- and y-axis label.
main	main title.
newpage	should a new page start?.

`type, xaxt, yaxt`
: strings to be used (when set to `"n"`) for suppressing the plotting of hexagon symbols, or the x- or y-axis, respectively.

`clip`
: either 'on' or 'off' are the allowed arguments, when on everything is clipped to the plotting region.

`verbose`
: logical indicating if some diagnostic output should happen.

`...`
: all arguments of `gplot.hexbin` can also be used for the S4 `plot` method.

Details

This is the (S4) `plot` method for `hexbin` (and `erodebin`) objects (erodebin-class).

To use the standalone function `gplot.hexbin()` is ***deprecated***. For `style`, `minarea` etc, see the **Details** section of `grid.hexagons`'s help page.

The legend functionality is somewhat preliminary. Later versions may include refinements and handle extreme cases (small and large) for cell size and counts.

Value

invisibly, a list with components

`plot.vp`
: the `hexViewport` constructed and used.

`legend.vp`
: if a legend has been produced, its `viewport`.

Author(s)

Dan Carr <dcarr@voxel.galaxy.gmu.edu>, ported by Nicholas Lewin-Koh <kohnicho@comp.nus.edu.s and Martin Maechler.

References

see in `grid.hexagons`.

See Also

`hexbin`, `hexViewport`, `smooth.hexbin`, `erode.hexbin`, `hcell2xy`, `hboxplot`, `hdiffplot`.

Examples

```
## 1) simple binning of spherical normal:
x <- rnorm(10000)
y <- rnorm(10000)
bin <- hexbin(x,y)

## Plot method for hexbin !
## ---- ------     --------
plot(bin)
# nested lattice
plot(bin, style= "nested.lattice")

# controlling the colorscheme
plot(bin, colramp=BTY, colorcut=c(0,.1,.2,.3,.4,.6,1))

## 2) A mixture distribution
x <- c(rnorm(5000),rnorm(5000,4,1.5))
```

```
y <- c(rnorm(5000),rnorm(5000,2,3))
bin <- hexbin(x,y)

pens <- cbind(c("#ECE2F0","#A6BDDB","#1C9099"),
              c("#FFF7BC","#FEC44F","#D95F0E"))
plot(bin, style = "nested.lattice", pen=pens)
# now really crazy
plot(bin, style = "nested.lattice", pen=pens,border=2,density=35)

# lower resolution binning and overplotting with counts
bin <- hexbin(x,y,xbins=25)
P <- plot(bin, style="lattice", legend=FALSE,
          minarea=1, maxarea=1, border="white")
##

pushHexport(P$plot.vp)
xy <- hcell2xy(bin)
  # to show points rather than counts :
grid.points(x,y,pch=18,gp=gpar(cex=.3,col="green"))
grid.text(as.character(bin@count), xy$x,xy$y,
          gp=gpar(cex=0.3, col="red"),default.units="native")
popViewport()

# Be creative, have fun!
```

`grid.hexagons` *Add Hexagon Cells to Plot*

Description

Plots cells in an hexbin object. The function distinquishes among counts using 5 different styles. This function is the hexagon plotting engine from the `plot` method for `hexbin` objects.

Usage

```
grid.hexagons(dat, style = c("colorscale", "centroids", "lattice",
      "nested.lattice", "nested.centroids", "constant.col"),
         use.count=TRUE, cell.at=NULL,
 minarea = 0.05, maxarea = 0.8, check.erosion = TRUE,
 mincnt = 1, maxcnt = max(dat@count), trans = NULL,
 colorcut = seq(0, 1, length = 17),
 density = NULL, border = NULL, pen = NULL,
 colramp = function(n){ LinGray(n,beg = 90, end = 15) },
 def.unit=  "native",
 verbose = getOption("verbose"))
```

Arguments

`dat`	an object of class `hexbin`, see `hexbin`.
`style`	character string specifying the type of plotting; must be (a unique abbrevation) of the values given in 'Usage' above.
`use.count`	logical specifying if counts should be used.

cell.at	numeric vector to be plotted instead of counts, must besame length as the number of cells.
minarea	numeric, the fraction of cell area for the lowest count.
maxarea	the fraction of the cell area for the largest count.
check.erosion	logical indicating only eroded points should be used for "erodebin" objects; simply passed to hcell2xy, see its documentation.
mincnt	numeric; cells with counts smaller than mincnt are not shown.
maxcnt	cells with counts larger than this are not shown.
trans	a transformation function (or NULL) for the counts, e.g., sqrt.
colorcut	a vector of values covering [0, 1] which determine hexagon color class boundaries or hexagon size boundaries – for style = "colorscale" only.
density	grid.polygon argument for shading. 0 causes the polygon not to be filled. *This is not implemented (for grid.polygon) yet.*
border	grid.polygon() argument. Draw the border for each hexagon.
pen	colors for grid.polygon(). Determines the color with which the polygon will be filled.
colramp	function of an integer argument n returning n colors. n is determined
def.unit	default unit to be used.
verbose	logical indicating if some diagnostic output should happen.

Details

The six plotting styles have the following effect:

style="lattice" or "centroids": Plots the hexagons in different sizes based on counts. The "lattice" version centers the hexagons at the cell centers whereas "centroids" moves the hexagon centers close to the center of mass for the cells. In all cases the hexagons will not plot outside the cell unless maxarea > 1. Counts are rescaled into the interval [0,1] and colorcuts determine the class boundaries for sizes and counts. The pen argument for this style should be a single color or a vector of colors of length(bin@count).

style="colorscale": Counts are rescaled into the interval [0,1] and colorcuts determines the class boundaries for the color classes. For this style, the function passed as colramp is used to define the n colors for the n+1 color cuts. The pen argument is ignored.

See LinGray for the default colramp and alternative "color ramp" functions.

style="constant.col": This is an even simpler alternative to "colorscale", using constant colors (determined pen optionally).

style="nested.lattice" and "nested.centroids": Counts are partitioned into classes by power of 10. The encoding nests hexagon size within powers of 10 color contours.

If the pen argument is used it should be a matrix of colors with 2 columns and either ceiling(log10(max(bin
or length(bin@count) rows. The default uses the R color palatte so that pens numbers 2-11 determine colors for completely filled cell Pen 2 is the color for 1's, Pen 3 is the color for 10's, etc. Pens numbers 12-21 determine the color of the foreground hexagons. The hexagon size shows the relative count for the power of 10. Different color schemes give different effects including 3-D illusions

Hexagon size encoding `minarea` *and* `maxarea` determine the area of the smallest and largest hexagons plotted. Both are expressed fractions of the bin cell size. Typical values might be .04 and 1. When both values are 1, all plotted hexagons are bin cell size, if `maxarea` is greater than 1 than hexagons will overlap. This is sometimes interesting with the lattice and centroid styles.

Count scaling

```
relcnt <- (trans(cnt)-trans(mincnt)) / (trans(maxcnt)-trans(mincnt))
area <- minarea + relcnt*maxarea
```

By default the transformation `trans()` is the identity function. The legend routine requires the transformation inverse for some options.

Count windowing `mincnt` *and* `maxcnt` Only routine only plots cells with cnts in [mincnts, maxcnts]

SIDE EFFECTS

Adds hexagons to the plot.

Author(s)

Dan Carr <dcarr@voxel.galaxy.gmu.edu>; ported and extended by Nicholas Lewin-Koh `<nikko@hailmail.net>`.

References

Carr, D. B. (1991) Looking at Large Data Sets Using Binned Data Plots, pp. 7–39 in *Computing and Graphics in Statistics*; Eds. A. Buja and P. Tukey, Springer-Verlag, New York.

See Also

`hexbin`, `smooth.hexbin`, `erode.hexbin`, `hcell2xy`, `gplot.hexbin`, `hboxplot`, `hdiffplot`, `grid.hexlegend`

Examples

```
set.seed(506)
x <- rnorm(10000)
y <- rnorm(10000)

# bin the points
bin <- hexbin(x,y)

# Typical approach uses plot( <hexbin> ) which controls the plot shape :
plot(bin, main = "Bivariate rnorm(10000)")

## but we can have more manual control:

# A mixture distribution
x <- c(rnorm(5000),rnorm(5000,4,1.5))
y <- c(rnorm(5000),rnorm(5000,2,3))
hb2 <- hexbin(x,y)

# Show color control and overplotting of hexagons
## 1) setup coordinate system:
P <- plot(hb2, type="n", main = "Bivariate mixture (10000)")# asp=1

## 2) add hexagons (in the proper viewport):
```

```
pushHexport(P$plot.vp)
grid.hexagons(hb2, style= "lattice", border = gray(.1), pen = gray(.6),
              minarea = .1, maxarea = 1.5)
popViewport()

## How to treat 'singletons' specially:
P <- plot(hb2, type="n", main = "Bivariate mixture (10000)")# asp=1
pushHexport(P$plot.vp)
grid.hexagons(hb2, style= "nested.centroids", mincnt = 2)# not the single ones
grid.hexagons(hb2, style= "centroids", maxcnt = 1, maxarea=0.04)# single points
popViewport()
```

`grid.hexlegend` *Add a Legend to a Hexbin Plot*

Description

Plots the legend for the `plot` method of `hexbin`. Provides a legend indicating the count representations.

Usage

```
grid.hexlegend(legend, ysize, lcex, inner, style = ,
      minarea = 0.05, maxarea = 0.8, mincnt = 1, maxcnt, trans = NULL,
   inv = NULL, colorcut, density = NULL, border = NULL, pen = NULL,
   colramp = function(n) { LinGray(n,beg = 90,end = 15) },
           leg.unit = "native")
```

Arguments

`legend`	positive number giving width of the legend in inches.
`ysize`	height of legend in inches
`lcex`	the characters expansion size for the text in the legend, see `par(cex=)`.

inner	the inner diameter of a hexagon in inches.
style	the hexagon style; see grid.hexagons.
minarea, maxarea	fraction of the cell area for the lowest and largest count, respectively.
mincnt, maxcnt	minimum and maximum count accepted in plot.
trans	a transformation function for the counts such as sqrt.
inv	the inverse transformation function.
colorcut	numeric vector of values covering [0, 1] the determine hexagon color classes boundaries and hexagon legend size boundaries.
border	argument for polygon(). Draw the border for each hexagon.
density	argument for polygon() filling. A 0 causes the polygon not to be filled.
pen	color argument used for polygon(col = .). Determines the color with which the polygon will be filled.
colramp	function accepting an integer n as an argument and returning n colors.
leg.unit	unit to use

Details

The plot method for hexbin objects calls this function to produce a legend by setting the graphics parameters, so hex.legend itself is not a standalone function.

The legend function is **preliminary**. Later version will include refinements and handle extreme cases (small and large) for cell size and counts.

See the **Details** section of grid.hexagons's help page.

Value

This function does not return any value.

Author(s)

Dan Carr <dcarr@voxel.galaxy.gmu.edu>

ported by Nicholas Lewin-Koh <kohnicho@comp.nus.edu.sg>

References

see in grid.hexagons.

See Also

hexbin, grid.hexagons,

smooth.hexbin, erode.hexbin,

hcell2xy, gplot.hexbin,

Examples

```
## Not a stand alone function; typically only called from plot.hexbin()


## Not run:
  grid.hexlegend(legend = 2, ysize = 1,lcex=8,inner=0.2,
                 maxcnt = 100, colorcut = c(0.5,0.5))

## End(Not run)
```

---

hboxplot *2-D Generalization of Boxplot*

Description

If `bin` is an *eroded* `hexbin` object, i.e., an `erodebin` object, `hboxplot()` plots the high counts cells selected by `erode()`. By default, the high counts cells contain 50 percent of the counts so analagous to the interquartile "range". The function distinguishes the last cells eroded using color. These cells correspond to one definition of the bivariate median.

Usage

```
hboxplot(bin, xbnds = NULL, ybnds = NULL,
         density, border = c(0, grey(0.7)), pen = c(2, 3),
         unzoom = 1.1, clip ="off", reshape = FALSE,
         xlab = NULL, ylab = NULL, main = "")
```

Arguments

`bin`	an object of class `hexbin`.
`xbnds,ybnds`	global x- and y-axis plotting limits for multiple plots.
`density, border`	arguments for `polygon()` each of length two, the first for the median, the second for the other cells.
`pen`	colors ("pen numbers") for `polygon()`.
`unzoom`	plot limit expansion factor when `xbnds` is missing.
`clip`	either 'on' or 'off' are the allowed arguments, when on everything is clipped to the plotting region.
`reshape`	logical value to reshape the plot although `xbnds` and `ybnds` are present.
`xlab, ylab, main`	x- and y- axis labels and main title

Details

The `density`, `border`, and `pen` arguments correspond to the `polygon` function calls for plotting two types of cells. The cell types, pen numbers and suggested colors are

TYPE	PEN	COLOR
cells of bin	2	light gray
last eroded cells of bin (median cells)	1	black

The erode components of the hexbin objects must be present for the medians cells to plot.

When `xbnds` is missing or `reshape` is true, the plot changes graphics parameters and resets them. When `xbnds` is missing the function also zooms in based on the available data to provide increased resolution.

The zoom used the hexagon cell centers. The unzoom argument backs off a bit so the whole hexagon will fit in the plot.

`Hboxplot()` is used as a stand alone function, for producing separate legends .....

Value

invisibly, the `hexViewport()` used internally. Used to add to the plot afterwards.

References

see in `grid.hexagons`.

See Also

`hexbin`, `erode`,

`hcell2xy`, `gplot.hexbin`,

`grid.hexagons`, `grid.hexlegend`

Examples

```
##  boxplot of smoothed counts
x <- rnorm(10000)
y <- rnorm(10000)

bin <- hexbin(x,y)
erodebin <- erode(smooth.hexbin(bin))

hboxplot(erodebin)
hboxplot(erodebin, density = c(32,7), border = c(2,4))
hp <- hboxplot(erodebin, density = c(-1,17),
               main = "hboxplot(erode*(smooth*(.)))")
pushHexport(hp)
grid.points(x[1:10], y[1:10])# just non-sense to show the principle
popViewport()
```

`hcell2xy` *Compute X and Y Coordinates for Hexagon Cells*

Description

Computes x and y coordinates from hexagon cell id's.

Usage

```
hcell2xy(hbin, check.erosion = TRUE)
```

Arguments

hbin	a object of class "hexbin", typically produced by hexbin(*).
check.erosion	logical indicating if only the eroded points should be returned in the case where hbin inherits from "erodebin" (see erodebin-class); is TRUE by default.

Details

The hexbin object hbin contains all the needed information. The purpose of this function is to reduce storage. The cost is additional calculation.

Value

A list with two components of the same length as bin$cell,

x

y

See Also

hexbin.

Examples

```
x <- rnorm(10000)
y <- rnorm(10000)
plot(x,y, pch=".")
hbin <- hexbin(x,y)
str(xys <- hcell2xy(hbin))
points(xys, cex=1.5, col=2) ; title("hcell2xy( hexbin(..) )", col.main=2)
```

---

hcell2xyInt *Change cell ids to 2d integer coordinate system*

Description

Transforms the cell representation of a a lattice into a 2d integer coordinate system.

Usage

```
hcell2xyInt(hbin, xbins=NULL, xbnds=NULL, ybnds=NULL, shape=NULL)
```

Arguments

hbin	a object of class "hexbin", typically produced by hexbin(*).
xbins	the number of bins partitioning the range of xbnds.
xbnds, ybnds	horizontal and vertical limits of the binning region in x or y units respectively; must be numeric vector of length 2.
shape	the *shape* = yheight/xwidth of the plotting regions.

Details

Takes a grid defined by either the hexbin parameters or dimen in a hexbin object and translates the cell ids for the grid into 2d integer coordinates.

Value

An integer matrix with two columns, i and j representing the integer xy coordinates of the hexagon grid.

`i`	Integer coordiante of the rows, increases from bottom to top
`j`	Integer coordiante of the columns, increases from left to right

Author(s)

Nicholas Lewin-Koh

See Also

`hcell2xy`

Examples

```
x<-rnorm(10000)
y<-rnorm(10000)
hbin<-hexbin(x,y)
ijInt<-hcell2xyInt(hbin)
```

---

hdiffplot *Plot of Domain and Median Differences of Two "hexbin" Objects*

Description

Let `bin1` and `bin2` represent two `hexbin` objects with scaling, plot shapes, and bin sizes. This plot distinguishes cells unique to `bin1`, cells in common, and cells unique to `bin2` using color. When the erode components are present, color also distinguishes the two erosion medians. An arrow shows the vector from the median of `bin1` to the median of `bin2`.

Usage

```
hdiffplot(bin1, bin2 = NULL, xbnds, ybnds,
          focus = NULL,
          col.control = list(medhex = "white", med.bord = "black",
          focus = NULL, focus.border = NULL, back.col = "grey"),
          arrows = TRUE, size = unit(0.1, "inches"), lwd = 2,
          eps = 1e-6, unzoom = 1.08, clip="off", xlab = "", ylab = "",
          main = deparse(mycall), ...)
```

Arguments

bin1, bin2	two objects of class hexbin.
xbnds,ybnds	global x- and y-axis plotting limits. Used primarily for multiple comparison plots.
focus	a vector of integers specifying which hexbin objects should be treated as focal. Excluded hexbins are treated as background.
col.control	a list for detailed color control.
arrows	a logical indicating wheter or not to draw arrows between the focal hexbin objects median cells.
border	border arguments to polygon
size	arrow type size in inches.
eps	distance criteria for distinct medians
unzoom	plot limit expansion factor when xbnds is missing
clip	either 'on' or 'off' are the allowed arguments, when on everything is clipped to the plotting region.
lwd	Line width for arrows, ignored when arrows=FALSE or when bins have no erosion component
xlab	label for x-axis
ylab	label for y-axis
main	main title for the plot; automatically constructed by default.
...	.....

Details

The hexbin objects for comparison, bin1 and bin2, must have the same plotting limits and cell size. The plot produces a comparison overlay of the cells in the two objects. If external global scaling is not supplied, the algorithm determines plotting limits to increase resolution. For example, the objects may be the result of the erode.hexbin() and include only high count cells containing 50 of the counts. The density, border, and pen arguments correspond to the polygon function calls for plotting six types of cells. The cell types are respectively:

unique cells of bin1,
joint cells,
unique cells of bin2,
median cell of bin1,
median cell of bin2,
median cell if identical.

The erode components of the hexbin objects must be present for the medians to plot. The algorithm select a single cell for the median if there are algorithmic ties.

The pen numbers for types of cells start at Pen 2. Pen 1 is presumed black. The suggested six additional colors are light blue, light gray, light red, blue, red, and black. Carr (1991) shows an example for black and white printing. That plot changes the six colors to light gray, dark gray, white, black, black, and black. It changes the 4th, 5th, and 6th argument of border to TRUE. It also changes 4th, 5th and 6th argument of density to 0. In other words cells in common do not show and medians cells appear as outlines.

When xbnds is missing, the plot changes graphics parameters and resets them. The function also zooms in based on the available data to provide increased resolution.

References

see in grid.hexagons.

See Also

hexbin, smooth.hexbin, erode.hexbin,

hcell2xy, gplot.hexbin, hboxplot, grid.hexagons, grid.hexlegend.

Examples

```
## Comparison of two bivariate boxplots
x1 <- rnorm(10000)
y1 <- rnorm(10000)
x2 <- rnorm(10000,mean=.5)
y2 <- rnorm(10000,mean=.5)
xbnds <- range(x1,x2)
ybnds <- range(y1,y2)

bin1 <- hexbin(x1,y1,xbnds=xbnds,ybnds=ybnds)
bin2 <- hexbin(x2,y2,xbnds=xbnds,ybnds=ybnds)
erodebin1 <- erode.hexbin(smooth.hexbin(bin1))
erodebin2 <- erode.hexbin(smooth.hexbin(bin2))

hdiffplot(erodebin1,erodebin2)

## Compare  *three* of them: --------------------

x3 <- rnorm(10000,mean=-1)
y3 <- rnorm(10000,mean=-.5)
xbnds <- range(x1,x2,x3)
ybnds <- range(y1,y2,y3)

bin1 <- hexbin(x1,y1,xbnds=xbnds,ybnds=ybnds)
bin2 <- hexbin(x2,y2,xbnds=xbnds,ybnds=ybnds)
bin3 <- hexbin(x3,y3,xbnds=xbnds,ybnds=ybnds)
erodebin1 <- erode.hexbin(smooth.hexbin(bin1))
erodebin2 <- erode.hexbin(smooth.hexbin(bin2))
erodebin3 <- erode.hexbin(smooth.hexbin(bin3))

bnlst <- list(b1=erodebin1, b2=erodebin2, b3=erodebin3)
hdiffplot(bnlst)
```

---

hexGraphPaper *Create a Hexgon Grid*

Description

Creates a hexagon grid that can be added to a plot created with grid graphics.

Usage

```
hexGraphPaper(hb, xbnds = NULL, ybnds = NULL, xbins = 30, shape = 1,
              add = TRUE, fill.edges = 1, fill = 0, border = 1)

hgridcent(xbins, xbnds, ybnds, shape, edge.add = 0)
```

Arguments

hb	a object of class "hexbin", typically produced by hexbin(*).
xbnds, ybnds	horizontal and vertical limits of the binning region in x or y units respectively; must be numeric vector of length 2.
xbins	the number of bins partitioning the range of xbnds.
shape	the *shape* = yheight/xwidth of the plotting regions.
add	a logical value indicating whether or not to add the grid to the current plot.
fill.edges	integer number of hexagons to add around the border
fill	the fill color for the hexagons
border	the color of the border of the hexagons
edge.add	offset (typically fill.edges above) used in hgridcent.

Details

If a hexbin object is given then the parameters xbins and shape are ignored. Different bounds can still be specified. The fill.edges parameter should be an integer. fill.edges takes the current grid and adds a layer of hexagons around the grid for each level of fill. So for example if fill.edges= 2 than the dimensions of the grid would be (i,j)+4.

hgridcent() is the utility function computing the resulting list (see section "Value").

WARNING! If using a hexVP be sure to set clip to "on", otherwise the hexagon grid will bleed over the plot edges.

Value

Invisibly returns a list with th following components

x	The x coordinates of the grid
y	the y coordinates of the grid
dimen	a vector of length 2 gining the rows and columns of the grid
dx	the horizontal diameter of the hexagons
dy	the vertical diameter of the hexagons

Author(s)

Nicholas Lewin-Koh

See Also

hcell2xy, hexpolygon, grid.hexagons

Examples

```
x <- rnorm(10000)
y <- rnorm(10000,x,x)
hbin <- hexbin(x,y)
hvp <- plot(hbin,type="n")
pushHexport(hvp$plot,clip="on")
hexGraphPaper(hbin,border=grey(.8))
grid.hexagons(hbin)
```

---

hexList *Conditional Bivariate Binning into Hexagon Cells*

Description

Creates a list of hexbin objects. Basic components are a cell id and a count of points falling in each occupied cell. Basic methods are show(), plot() and summary(), but also erode.

Usage

```
hexList(x, y = NULL, given = NULL, xbins = 30, shape = 1,
        xbnds = NULL, ybnds = NULL, xlab = NULL, ylab = NULL)
```

Arguments

x	x coordinate to be binned
y	y coordinate to be binned
given	..
xbins	number of bins partitioning the range of xbnds
shape	the *shape* = yheight/xwidth of the plotting regions
xbnds	horizontal limits of binning
ybnds	vertical limits of binning
xlab	character strings used as labels for x
ylab	character strings used as labels for y

Details

There is also a coerce method to produce hexbinList objects from lists.

Value

If it is a LIST, use

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

Author(s)

Nicholas Lewin-Koh

See Also

hexbin, hdiffplot

`hexMA.loess` *Add Loess Fit to Hexplot*

Description

Fit a loess line using the hexagon centers of mass as the x and y coordinates and the cell counts as weights.

Usage

```
hexMA.loess(pMA, span = 0.4, col = "red", n = 200)
hexVP.loess(hbin, hvp = NULL, span = 0.4, col = "red", n = 200)
```

Arguments

`hbin`	an object of class `hexbin`, see `hexbin`.
`hvp`	A `hexViewport` object.
`pMA`	the list returned by `plotMAhex`.
`span`	the parameter alpha which controls the degree of smoothing.
`col`	line color for the loess fit.
`n`	number of points at which the fit should be evaluated.

Value

Returns invisibly the object associated with the loess fit.

Author(s)

Nicholas Lewin-Koh

See Also

`hexVP.abline`, `plotMAhex`, `gplot.hexbin`, `hexViewport`; `loess`

Examples

```
if(require(marray)){
  data(swirl)

  hb <- plotMAhex(swirl[,1], main = "M vs A plot with hexagons", legend=0)
  hexVP.abline(hb$plot, h=0, col= gray(.6))
  hexMA.loess(hb)
}
```

hexTapply *Apply function to data from each hexagon bin.*

Description

A wrapper for tapply except that it operates with each hexagon bin being the category. The function operates on the data associated on the points from each bin.

Usage

```
hexTapply(hbin, dat, FUN = sum, ..., simplify=TRUE)
```

Arguments

hbin	a object of class "hexbin", typically produced by hexbin(*).
dat	A vector of data the same length as hbin@cID
FUN	the function to be applied. In the case of functions like +, %*%, etc., the function name must be quoted. If FUN is NULL, tapply returns a vector which can be used to subscript the multi-way array tapply normally produces.
...	optional arguments to FUN.
simplify	If FALSE, tapply always returns an array of mode "list". If TRUE (the default), then if FUN always returns a scalar, tapply returns an array with the mode of the scalar.

Details

This function is a wrapper for tapply, except that the cell id is always the categorical variable. This function is specifically good for adding variables to the cAtt slot of a hexbin object or for plotting a third variable in a hexagon plot. See below for examples.

Value

Returns a vector of the result of 'FUN' as in tapply. See tapply for detailed description of output.

Author(s)

Nicholas Lewin-Koh

See Also

tapply,hexbin

Examples

```
data(NHANES)
hbin<-hexbin(log(NHANES$Diet.Iron+1),log(NHANES$BMI),xbins=25,IDs=TRUE)
hvp<-plot(hbin)
mtrans<-hexTapply(hbin,NHANES$Transferin,median,na.rm=TRUE)
pushHexport(hvp$plot.vp)
grid.hexagons(hbin,style='lattice',pen=0,border='red',use.count=FALSE,
```

```
cell.at=mtrans)
```

`hexVP-class` *Formal class "hexVP" of a Hexagon Viewport*

Description

Hexagon Viewports are "value-added" grid viewports (see `viewport`) where the extra slots contain scaling and "embedding" information. A hexViewport is created my taking the available area in the cuurent viewport on the graphics device and maximizing the amount of area with a fied aspect ratio. The default when the shape parameter is 1, is a 1:1 aspect ratio in terms of the size of the viewport, not the scale of the x and y axis. The plotting area is centered within the existing margins and the maximum size determined. Extra area is then allocated to the margins. This viewport is replicated twice, once with clipping set to "on" and once with clipping "off". This feature can be used for toggling clipping on and off while editing the plot.

Objects from the Class

Objects are typically created by calls to `hexViewport()` or by low level calls of the form `new("hexVP", ...)`.

Slots

`hexVp.off`**:** Object of class `"viewport"` with clipping set to off, see `viewport`.

`hexVp.on`**:** Object of class `"viewport"`, with the same dimensions and parameters as hexVp.off, but with clipping set to on, see `viewport`.

`hp.name`**:** The name of the viewport for searching a vptree.

`mar`**:** `unit` vector of four margins (typically in `"lines"`).

`fig`**:** `unit` vector of two figure sizes (typically in `"npc"`).

`plt`**:** `unit` vector of two figure sizes (typically in `"npc"`).

`shape`**:** The shape parameter from the plotted `hexbin` object.

`xscale`**:** numeric of length two specifying x-range.

`yscale`**:** numeric of length two specifying y-range.

Methods

These are methods accessing the slots of corresponding name.

getFig `signature(hvp = "hexVP"):` ...

getMargins `signature(hvp = "hexVP"):` ...

getPlt `signature(hvp = "hexVP"):` ...

getXscale `signature(hvp = "hexVP"):` ...

getYscale `signature(hvp = "hexVP"):` ...

Author(s)

Nicholas Lewin-Koh <kohnicho@comp.nus.edu.sg>.

See Also

The constructor function `hexViewport`. `hexbin`, and its S4 plotting method, `gplot.hexbin`.

Examples

```
example(hexViewport, echo=FALSE)
## continued:
str(P$plot.vp)
```

---

hexVP.abline *Add a Straight Line to a HexPlot*

Description

This function adds one or more straight lines through the current plot; it is the hexbin version of `abline()`.

Usage

```
hexVP.abline(hvp, a = NULL, b = NULL, h = numeric(0), v = numeric(0),
             col = "black", lty = 1, lwd = 2, ...)
```

Arguments

`hvp`	A hexViewport object that is currently on the active device
`a,b`	the intercept and slope or if `b` is `NULL`, an `lm` object or a vector of length 2 with `c(intercept,slope)`
`h`	the y-value for a horizontal line.
`v`	the x-value for a vertical line.
`col, lty, lwd`	line color, type and width.
`...`	further graphical parameters.

Details

The first form specifies the line in intercept/slope form (alternatively `a` can be specified on its own and is taken to contain the slope and intercept in vector form).

The `h=` and `v=` forms draw horizontal and vertical lines at the specified coordinates.

The `coef` form specifies the line by a vector containing the slope and intercept.

`lm` is a regression object which contains `reg$coef`. If it is of length 1 then the value is taken to be the slope of a line through the origin, otherwise, the first 2 values are taken to be the intercept and slope.

Author(s)

Nicholas Lewin-Koh

See Also

gplot.hexbin, hexViewport, hexMA.loess

---

hexViewport *Compute a Grid Viewport for Hexagon / Hexbin Graphics*

Description

Builds a grid viewport for hexagon or hexbin graphics. This builds on the concepts of the **grid** package, see viewport.

Usage

```
hexViewport(x, offset = unit(0,"inches"), mar = NULL,
    xbnds = NULL, ybnds = NULL, newpage = FALSE,
            clip = "off", vp.name = NULL)
```

Arguments

x	a hexbin object.
offset	a unit object.
mar	margins as units, of length 4 or 1.
xbnds, ybnds	bounds for x- and y- plotting range; these default to the corresponding slots of x.
newpage	logical indicating if a new graphics page should be openend, i.e., grid.newpage().
clip	simply passed to viewport().
vp.name	name of viewport; defaults to random name.

Value

an S4 object of class "hexVP", see hexVP-class for more, with its main slot hexVp a viewport for grid graphics.

See Also

viewport and the main *"handlers"* pushHexport and popViewport; further gplot.hexbin and hboxplot which build on hexViewport.

Examples

```
set.seed(131)
x <- rnorm(7777)
y <- rt   (7777, df=3)

## lower resolution binning and overplotting with counts
bin <- hexbin(x,y,xbins=25)
P <- plot(bin)
xy <- hcell2xy(bin)
pushHexport(P$plot.vp)
```

```
i <- bin@count <= 3
grid.text(as.character(bin@count[i]), xy$x[i], xy$y[i],
          default.units = "native")
grid.points(x[1:20],y[1:20]) # to show some points rather than counts
popViewport()
```

`hexbin` *Bivariate Binning into Hexagon Cells*

Description

Creates a `"hexbin"` object. Basic components are a cell id and a count of points falling in each occupied cell.

Basic methods are `show()`, `plot()` and `summary()`, but also `erode`.

Usage

```
hexbin(x, y, xbins = 30, shape = 1,
       xbnds = range(x), ybnds = range(y),
       xlab = NULL, ylab = NULL, IDs = FALSE)
```

Arguments

`x, y`	vectors giving the coordinates of the bivariate data points to be binned. Alternatively a single plotting structure can be specified: see `xy.coords`. `NA`'s are allowed and silently omitted.
`xbins`	the number of bins partitioning the range of xbnds.
`shape`	the *shape* = yheight/xwidth of the plotting regions.
`xbnds, ybnds`	horizontal and vertical limits of the binning region in x or y units respectively; must be numeric vector of length 2.
`xlab, ylab`	optional character strings used as labels for `x` and `y`. If `NULL`, sensible defaults are used.
`IDs`	logical indicating if the individual cell "IDs" should be returned, see also below.

Details

Returns counts for non-empty cells only. The plot shape must be maintained for hexagons to appear with equal sides. Some calculations are in single precision.

Note that when plotting a `hexbin` object, the **grid** package is used. You must use its graphics (or those from package **lattice** if you know how) to add to such plots.

Value

an S4 object of class `"hexbin"`. It has the following slots:

`cell`	vector of cell ids that can be mapped into the (x,y) bin centers in data units.
`count`	vector of counts in the cells.
`xcm`	The x center of mass (average of x values) for the cell.
`ycm`	The y center of mass (average of y values) for the cell.

xbins	number of hexagons across the x axis. hexagon inner diameter =diff(xbnds)/xbins in x units
shape	plot shape which is yheight(inches) / xwidth(inches)
xbnds	x coordinate bounds for binning and plotting
ybnds	y coordinate bounds for binning and plotting
dimen	The i and j limits of cnt treated as a matrix cnt[i,j]
n	number of (non NA) (x,y) points, i.e., `sum(* @count)`.
ncells	number of cells, i.e., `length(* @count)`, etc
call	the function call.
xlab, ylab	character strings to be used as axis labels.
cID	of class, `"integer or NULL"`, only if `IDs` was true, an integer vector of length `n` where `cID[i]` is the cell number of the i-th original point `(x[i], y[i])`. Consequently, the `cell` and `count` slots are the same as the `names` and entries of `table(cID)`, see the example.

References

Carr, D. B. et al. (1987) Scatterplot Matrix Techniques for Large N. *JASA* **83**, 398, 424–436.

See Also

`hcell2xy`

`gplot.hexbin`,

`grid.hexagons`, `grid.hexlegend`.

Examples

```
set.seed(101)
x <- rnorm(10000)
y <- rnorm(10000)
(bin <- hexbin(x, y))
## or
plot(hexbin(x, y + x*(x+1)/4),
     main = "(X, X(X+1)/4 + Y)  where X,Y ~ rnorm(10000)")

## Using plot method for hexbin objects:
plot(bin, style = "nested.lattice")

hbi <- hexbin(y ~ x, xbins = 80, IDs= TRUE)
str(hbi)
tI <- table(hbi@cID)
stopifnot(names(tI) == hbi@cell,
                tI  == hbi@count)

## NA's now work too:
x[runif(6, 0, length(x))] <- NA
y[runif(7, 0, length(y))] <- NA
hbN <- hexbin(x,y)
summary(hbN)
```

`hexbinplot` *Trellis Hexbin Displays*

Description

Display of hexagonally binned data, as implemented in the `hexbin` packge, under the Trellis framework, with associated utilities. `hexbinplot` is the high level generic function, with the `"formula"` method doing the actual work. `prepanel.hexbinplot` and `panel.hexbinplot` are associated prepanel and panel functions. `hexlegendGrob` produces a suitable legend.

Usage

```
hexbinplot(x, data, ...)

## S3 method for class 'formula'
hexbinplot(x, data = NULL,
           prepanel = prepanel.hexbinplot,
           panel = panel.hexbinplot,
   groups = NULL,
           aspect = "xy",
           trans = NULL,
           inv = NULL,
           colorkey = TRUE,
           ...,
           maxcnt,
           legend = NULL,
           legend.width = TRUE,
           subset)

prepanel.hexbinplot(x, y, type = character(0), ...)

panel.hexbinplot(x, y, ..., groups = NULL)

hexlegendGrob(legend = 1.2,
              inner = legend / 5,
              cex.labels = 1,
              cex.title = 1.2,
              style = "colorscale",
              minarea = 0.05, maxarea = 0.8,
              mincnt = 1, maxcnt,
              trans = NULL, inv = NULL,
              colorcut = seq(0, 1, length = 17),
              density = NULL, border = NULL, pen = NULL,
              colramp = function(n) { LinGray(n,beg = 90,end = 15) },
              ...,
              vp = NULL,
              draw = FALSE)
```

Arguments

x	For hexbinplot, the object on which method dispatch is carried out. For the "formula" methods, a formula describing the form of conditioning plot. Formulas that are valid for xyplot are acceptable. In panel.hexbinplot, the x variable.
y	In panel.hexbinplot, the y variable.
data	For the formula method, a data frame containing values for any variables in the formula, as well as groups and subset if applicable (using groups currently causes an error with the default panel function). By default, the environment where the function was called from is used.
minarea, maxarea, mincnt, maxcnt, trans, inv, colorcut, density, border, pen, co	see gplot.hexbin
prepanel, panel, aspect	See xyplot. aspect="fill" is not allowed. The current default of "xy" may not always be the best choice, often aspect=1 will be more reasonable.
colorkey	logical, whether a legend should be drawn. Currently a legend can be drawn only on the right.
legend.width, legend	width of the legend in inches when style is "nested.lattice" or "nested.centroids". The name legend.width is used to avoid conflict with the standard trellis argument legend. It is possible to specify additional legends using the legend or key arguments as long as they do not conflict with the hexbin legend (i.e., are not on the right).
inner	Inner radius in inches of hexagons in the legend when style is "nested.lattice" or "nested.centroids".
cex.labels, cex.title	in the legend, multiplier for numeric labels and text annotation respectively
type	character vector controlling additional augmentation of the display. A "g" in type adds a reference grid, "r" adds a regression line (y on x), "smooth" adds a loess smooth
draw	logical, whether to draw the legend grob. Useful when hexlegendGrob is used separately
vp	grid viewport to draw the legend in
...	extra arguments, passed on as appropriate. Arguments to gplot.hexbin, xyplot, panel.hexbinplot and hexlegendGrob can be supplied to the high level hexbinplot call. panel.hexbinplot calls one of two (unexported) low-level functions depending on whether groups is supplied (although specifying groups currently leads to an error). Arguments of the appropriate function can be supplied; some important ones are xbins: number of hexagons covering x values. The number of y-bins depends on this, the aspect ratio, and xbnds and ybnds xbnds, ybnds: Numeric vector specifying range of values that should be covered by the binning. In a multi-panel display, it is not necessarily a good idea to use the same bounds (which along with xbins and the aspect ratio determine the size of the hexagons) for all panels. For example, when data is concentrated in small subregions of different panels, more detail will be shown by using smaller hexagons covering those regions. To

control this, xbnds and ybnds can also be character strings "panel" or "data" (which are not very good names and may be changed in future). In the first case, the bounds are taken to be the limits of the panel, in the second case, the limits of the data (packet) in that panel. Note that all panels will have the same limits (enough to cover all the data) by default if relation="free" in the standard trellis argument scales, but not otherwise.

groups — in hexbinplot, a grouping variable that is evaluated in data, and passed on to the panel function.

subset — an expression that is evaluated in evaluated in data to produce a logical vector that is used to subset the data before being used in the plot.

Details

The panel function panel.hexbinplot creates a hexbin object from data supplied to it and plots it using grid.hexagons. To make panels comparable, all panels have the same maxcnt value, by default the maximum count over all panels. This default value can be calculated only if the aspect ratio is known, and so aspect="fill" is not allowed. The default choice of aspect ratio is different from the choice in hexbin (namely, 1), which may sometimes give better results for multi-panel displays. xbnds and ybnds can be numeric range vectors as in hexbin, but they can also be character strings specifying whether all panels should have the same bins. If they are not, then bins in different panels could be of different sizes, in which case style="lattice" and style="centroids" should be interpreted carefully.

The dimensions of the legend and the size of the hexagons therein are given in absolute units (inches) by legend.width and inner only when style is "nested.lattice" or "nested.centroids". For other styles, the dimensions of the legend are determined relative to the plot. Specifically, the height of the legend is the same as the height of the plot (the panel and strip regions combined), and the width is the minimum required to fit the legend in the display. This is different in some ways from the hexbin implementation. In particular, the size of the hexagons in the legend are completely unrelated to the sizes in the panels, which is pretty much unavoidable because the sizes need not be the same across panels if xbnds or ybnds is "data". The size of the hexagons encode information when style is "lattice" or "centroids", consequently a warning is issued when a legend is drawn with wither of these styles.

Value

hexbinplot produces an object of class "trellis". The update method can be used to update components of the object and the print method (usually called by default) will plot it on an appropriate plotting device. hexlegendGrob produces a "grob" (grid object).

Author(s)

Deepayan Sarkar <deepayan@stat.wisc.edu>

See Also

hexbin, xyplot

Examples

```
mixdata <-
    data.frame(x = c(rnorm(5000),rnorm(5000,4,1.5)),
               y = c(rnorm(5000),rnorm(5000,2,3)),
```

```
                a = gl(2, 5000))
hexbinplot(y ~ x, mixdata, aspect = 1,
           trans = sqrt, inv = function(x) x^2)
hexbinplot(y ~ x | a, mixdata)
hexbinplot(y ~ x | a, mixdata, style = "lattice",
           xbnds = "data", ybnds = "data")
hexbinplot(y ~ x | a, mixdata, style = "nested.centroids")
hexbinplot(y ~ x | a, mixdata, style = "nested.centroids",
           border = FALSE, type = c("g", "smooth"))
```

`hexplom` *Hexbin Plot Matrices*

Description

`hexplom` draws Conditional Hexbin Plot Matrices. It is similar to `splom`, expect that the default display is different. Specifically, the default display is created using `panel.hexplom`, which is an alias for `panel.hexbinplot`.

Usage

```
hexplom(x, data, ...)

## S3 method for class 'formula'
hexplom(x, data = NULL, ...)

## S3 method for class 'data.frame'
hexplom(x, data = NULL, ..., groups = NULL,
   subset = TRUE)

## S3 method for class 'matrix'
hexplom(x, data = NULL, ..., groups = NULL, subset = TRUE)

panel.hexplom(...)
```

Arguments

`x`	The object on which method dispatch is carried out. For the `"formula"` method, a formula describing the structure of the plot, which should be of the form `~ x | g1 * g2 * ...`, where `x` is a data frame or matrix. Each of `g1, g2, ...` must be either factors or shingles. The conditioning variables `g1, g2, ...` may be omitted. For the `data.frame` and `matrix` methods, a data frame or matrix as appropriate.
`data`	For the `formula` method, an optional data frame in which variables in the formula (as well as `groups` and `subset`, if any) are to be evaluated. By default, the environment where the function was called from is used.
`groups, subset, ...`	see `splom`. The non-standard evaluation of `groups` and `subset` only applies in the `formula` method. Apart from arguments that apply to `splom` (many of which are only documented in `xyplot`), additional arguments meant for

panel.hexplom (which is an alias for panel.hexbinplot) may also be supplied. Such arguments may include ones that control details of the hexbin calculations, documented in gplot.hexbin

Value

An object of class "trellis". The update method can be used to update components of the object and the print method (usually called by default) will plot it on an appropriate plotting device.

Author(s)

Deepayan Sarkar <Deepayan.Sarkar@R-project.org>, Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

splom, xyplot, hexbinplot, Lattice, panel.pairs

Examples

```
## Simple hexplom
data(NHANES)
hexplom(~NHANES[,7:14], xbins=15)

## With colors and conditioning
hexplom(~NHANES[,9:13] | Sex, data = NHANES,
        xbins = 15, colramp = magent)

## With custom panel function
hexplom(NHANES[,9:13], xbins = 20,colramp = BTY,
        upper.panel = panel.hexboxplot)
```

---

hexpolygon *Hexagon Coordinates and Polygon Drawing*

Description

Simple 'low-level' function for computing and drawing hexagons. Can be used for 'grid' (package **grid**) or 'traditional' (package **graphics**) graphics.

Usage

```
hexcoords(dx, dy = NULL, n = 1, sep = NULL)

hexpolygon(x, y, hexC = hexcoords(dx, dy, n = 1), dx, dy = NULL,
           fill = 1, border = 0, hUnit = "native", ...)
```

Arguments

`dx,dy`	horizontal and vertical width of the hexagon(s).
`n`	number of hexagon "repeats".
`sep`	separator value to be put between coordinates of different hexagons. The default, `NULL` doesn't use a separator.
`x,y`	numeric vectors of the same length specifying the hexagon *centers* around which to draw.
`hexC`	a list as returned from `hexcoords()`. Its component `no.sep` determines if grid or traditional graphics are used. The default (via default of `hexcoords`) is now to use grid graphics.
`fill,border`	passed to `grid.polygon` (for **grid**).
`hUnit`	string or `NULL` determining in which units (x,y) values are.
`...`	further arguments passed to `polygon` (for **graphics**).

Value

`hexcoords()` returns a list with components

`x,y`	numeric vectors of length $n \times 6$ (or $n \times 7$ if `sep` is not NULL) specifying the hexagon polygon coordinates (with `sep` appended to each 6-tuple).
`no.sep`	a logical indicating if `sep` was `NULL`.

`hexpolygon` returns what its last `grid.polygon(.)` or `polygon(.)` call returns.

Author(s)

Martin Maechler, originally.

See Also

`grid.hexagons` which builds on these.

Examples

```
str(hexcoords(1, sep = NA))  # multiple of (6 + 1)
str(hexcoords(1, sep = NULL))# no separator -> multiple of 6


## hexpolygon()s:
x <- runif(20, -2, 2)
y <- x + rnorm(20)

## 1) traditional 'graphics'
plot(x,y, asp = 1, "plot() + hexpolygon()")
hexpolygon(x,y, dx = 0.1, density = 25, col = 2, lwd = 1.5)

## 2) "grid" :

addBit <- function(bnds, f = 0.05) bnds + c(-f, f) * diff(bnds)
sc <- addBit(rxy <- range(x,y))# same extents (cheating asp=1)
grid.newpage()
pushViewport(plotViewport(.1+c(4,4,2,1), xscale = sc, yscale = sc))
grid.rect()
```

```
grid.xaxis()
grid.yaxis()
grid.points(x,y)
hexpolygon(x,y, hexcoords(dx = 0.1, sep=NULL), border = "blue", fill=NA)
popViewport()
```

`hsmooth-methods` *Hexagon Bin Smoothing: Generic hsmooth() and Methods*

Description

Methods for the generic function `hsmooth` in package **hexbin**: There is currently only the one for `hexbin` objects.

Usage

```
## S4 method for signature 'hexbin'
hsmooth(bin, wts)
```

Arguments

`bin`	a `hexbin` object, or an extension such as `erodebin-class`.
`wts`	weights vector, see `smooth.hexbin`

Methods

bin = "hexbin" is just the `smooth.hexbin` function (for back compatibility); see its documentation, also for examples.

`inout.hex` *Check points for inclusion*

Description

Check which points are in hexagons with `count` <= mincnt.

Usage

```
inout.hex(hbin, mincnt)
```

Arguments

`hbin`	an object of class `hexbin`.
`mincnt`	Cutoff, id's for counts less than mincnt are returned

Details

Check which points are in hexagons with `count` <= mincnt and returns the row ids for those points. One can use the ids to plot low ount hexagons as points instead.

Value

A vector with the row ids of points which fall in hexagons with `count` less than or equal to mincnt

Author(s)

Nicholas Lewin-Koh

See Also

`plotMAhex`

`list2hexList` *Convert list to hexList*

Description

Converts a list of hexbin objects with same xbnds, ybnds, shape and xbins to a `hexList` object.

Usage

```
list2hexList(binlst)
```

Arguments

`binlst`	A list of hexbin objects

Value

a `hexList` object

Author(s)

Nicholas Lewin-Koh

See Also

`hexList`,`hdiffplot`

old-classes *Class "unit" and "viewport" as S4 classes*

Description

Package "hexbin" now uses S4 classes throughout and hence needs to `setOldClass` both `"unit"` and `"viewport"` (which are S3 classes from the **grid** package), in order to be able to use those in slots of its own classes.

Objects from the Class

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

Extends

Class `"oldClass"`, directly.

Methods

No methods defined with class "unit" in the signature.

optShape *Optimal Shape Parameter for Hexbin Viewport*

Description

Takes a viewport or a given height and width and returns the shape parameter that will fill the specified plotting region with the appropriately shaped hexagons. If margins are specified the margins are subtracted from height and width before the shape parameter is specified.

Usage

```
optShape(vp, height = NULL, width = NULL, mar = NULL)
```

Arguments

`vp`	a `viewport` object, optional see details
`height`	the height of the plotting region, can be numeric or units
`width`	The width of the plotting region, can be numeric or units
`mar`	A four element numeric or units vector describing the margins in the order `c(bottom, left, top, right)`

Value

a scalar numeric value specifiyng `shape`.

Warning

If a viewport is given as an argument it should already be pushed on the graphics device or it will have null units and a meaningless shape parameter will be returned.

Author(s)

Nicholas Lewin-Koh

See Also

hexViewport, hexVP-class, hexbin

Examples

```
x <- rgamma(10000,.9)
m <- as.logical(rbinom(10000,1,.17))
x[m] <- -x[m]
y <- rnorm(x,abs(x))
vp <- plotViewport(xscale= range(x)+c(-.5,.5),
                   yscale= range(y)+c(-.5,.5),
             default.units = "native")
grid.newpage()
pushViewport(vp)
grid.rect()
shape <- optShape(vp)
shape
hb <- hexbin(x,y,xbins=40,shape=shape)
grid.hexagons(hb,colramp=BTY)
```

---

panel.hexboxplot *Boxplot for hexbin lattice plot*

Description

A panel function to add a boxplot to a hexbin lattice plot.

Usage

```
panel.hexboxplot(x, y, xbins = 30,
                 xbnds = c("data", "panel"), ybnds = c("data", "panel"),
                 .prelim = FALSE, .cpl = current.panel.limits(),
                 .xlim = .cpl$xlim, .ylim = .cpl$ylim,
                 .aspect.ratio, type = character(0), cdfcut = 0.25,
                 shadow = 0.05, ..., check.erosion = TRUE)
```

Arguments

x, y	numeric vector or factor.
xbins	the number of bins partitioning the range of xbnds.
xbnds, ybnds	horizontal and vertical limits of the binning region in x or y units respectively; must be numeric vector of length 2.
.prelim, .cpl, .xlim, .ylim, .aspect.ratio	for internal use.
type	character vector controlling additional augmentation of the display. A "g" in type adds a reference grid, an "hg" adds a hexagonal grid.

cdfcut	number in (0,1) indicating the confidence level for the erosion limits. See erode.hexbin for more information.
shadow	number in (0,1) indicating the confidence level for the erosion limits of a boxplot shadow. See erode.hexbin for more information.
...	potential further arguments passed on.
check.erosion	logical indicating only eroded points should be used for "erodebin" objects; simply passed to hcell2xy, see its documentation.

Value

There is no return value from this function. The results are plotted on the current active device.

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

hexbinplot, panel.hexgrid, panel.bwplot

Examples

```
mixdata <-
    data.frame(x = c(rnorm(5000),rnorm(5000,4,1.5)),
               y = rep(1:2, 5000))
hexbinplot(y ~ x, mixdata, panel = panel.hexboxplot)
```

---

panel.hexgrid *Hexagonal grid for a lattice plot*

Description

A panel function to add a hexagonal grid to a lattice plot.

Usage

```
panel.hexgrid(h, border = grey(0.85))
```

Arguments

h	an object of class hexbin.
border	a color for the hexagon border colors

Value

There is no return value from this function. The results are plotted on the current active device.

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

hexbinplot, hexGraphPaper

---

panel.hexloess *Loess line for hexbin lattice plot*

Description

A panel function to add a loess line to a hexbin lattice plot.

Usage

```
panel.hexloess(bin, w = NULL, span = 2/3, degree = 1,
               family = c("symmetric", "gaussian"), evaluation = 50,
               lwd = add.line$lwd, lty = add.line$lty,
               col, col.line = add.line$col, ...)
```

Arguments

bin	an object of class hexbin.
w	optional counts for object bin.
span	smoothness parameter for loess.
degree	degree of local polynomial used.
family	if "gaussian" fitting is by least-squares, and if "symmetric" a re-descending M-estimator is used.
evaluation	number of points at which to evaluate the smooth curve.
lwd	line weight graphical parameter.
lty	line type graphical parameter.
col	same as col.line.
col.line	line color graphical parameter.
...	optional arguments to loess.control.

Value

There is no return value from this function. The results are plotted on the current active device.

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

hexbinplot, panel.hexgrid, loess.smooth, loess.control, panel.loess

plotMAhex *MA-plot using hexagon bins*

Description

Creates an MA-plot using hexagons with color/glyph coding for control spots.

Usage

```
plotMAhex(MA, array = 1, xlab = "A", ylab = "M",
          main = colnames(MA)[array], xlim = NULL, ylim = NULL,
          status = NULL, values, pch, col, cex, nbin = 40,
          zero.weights = FALSE, style = "colorscale", legend = 1.2,
          lcex = 1, minarea = 0.04, maxarea = 0.8, mincnt = 2,
          maxcnt = NULL, trans = NULL, inv = NULL, colorcut = NULL,
          border = NULL, density = NULL, pen = NULL,
          colramp = function(n) { LinGray(n, beg = 90, end = 15) },
          newpage = TRUE, type = c("p", "l", "n"),
          xaxt = c("s", "n"), yaxt = c("s", "n"),
          verbose = getOption("verbose"))
```

Arguments

MA	an RGList, MAList or MArrayLM object, or any list with components M containing log-ratios and A containing average intensities. Alternatively a matrix, Affybatch or ExpressionSet object.
array	integer giving the array to be plotted. Corresponds to columns of M and A.
xlab, ylab, main	character strings giving label for x-axis, y-axis or main tile of the plot.
xlim, ylim	numeric vectors of length 2 giving limits for x-axis (or y-axis respectively), defaulting to min and max of the data.
status	character vector giving the control status of each spot on the array, of same length as the number of rows of MA$M. If omitted, all points are plotted in the default color, symbol and size.
values	character vector giving values of status to be highlighted on the plot. Defaults to unique values of status. Ignored if there is no status vector.
pch	vector or list of plotting characters. Default to integer code 16. Ignored is there is no status vector.
col	numeric or character vector of colors, of the same length as values. Defaults to 1:length(values). Ignored if there is no status vector.
cex	numeric vector of plot symbol expansions, of the the same length as values. Defaults to 0.2 for the most common status value and 1 for the others. Ignored if there is no status vector.
nbin	Number of bins
zero.weights	logical, should spots with zero or negative weights be plotted?
style	string specifying the style of hexagon plot, see grid.hexagons for the possibilities.

legend	numeric width of the legend in inches of FALSE. In the latter case, or when 0, no legend is not produced.
lcex	characters expansion size for the text in the legend.
minarea	fraction of cell area for the lowest count.
maxarea	fraction of the cell area for the largest count.
mincnt	cells with fewer counts are ignored.
maxcnt	cells with more counts are ignored.
trans	function specifying a transformation for the counts such as sqrt.
inv	the inverse transformation of trans.
colorcut	vector of values covering [0, 1] that determine hexagon color class boundaries and hexagon legend size boundaries. Alternatively, an integer (<= maxcnt) specifying the *number* of equispaced colorcut values in [0,1].
border, density, pen	color for polygon borders and filling of each hexagon drawn, passed to grid.hexagons.
colramp	function accepting an integer n as an argument and returning n colors.
newpage	should a new page start?
type, xaxt, yaxt	strings to be used (when set to "n") for suppressing the plotting of hexagon symbols, or the x- or y-axis, respectively.
verbose	logical indicating if some diagnostic output should happen.

Details

An MA-plot is a plot of log-intensity ratios (M-values) versus log-intensity averages (A-values). If MA is an RGList or MAList then this function produces an ordinary within-array MA-plot. If MA is an MArrayLM object, then the plot is an fitted model MA-plot in which the estimated coefficient is on the y-axis and the average A-value is on the x-axis.

If MA is a matrix or ExpressionSet object, then this function produces a between-array MA-plot. In this case the A-values in the plot are the average log-intensities across the arrays and the M-values are the deviations of the log-intensities for the specified array from the average. If there are more than five arrays, then the average is computed robustly using medians. With five or fewer arrays, it is computed by means.

The status vector is intended to specify the control status of each spot, for example "gene", "ratio control", "house keeping gene", "buffer" and so on. The vector is usually computed using the function controlStatus from package **limma** and a spot-types file. However the function may be used to highlight any subset of spots.

The arguments values, pch, col and cex can be included as attributes to status instead of being passed as arguments to plotMA.

See points for possible values for pch, col and cex.

Value

A plot is created on the current graphics device. and a list with the following items is returned invisibly:

plot.vp	the hexViewport constructed and used.
legend.vp	if a legend has been produced, its viewport.
hbin	a hexbin object built with A as the x coordinate and M as the y coordinate.

Author(s)

Nicholas Lewin-Koh, adapted from code by Gordon Smyth

References

See http://www.statsci.org/micrarra/refs/maplots.html

See Also

plotMA from package **limma**, and gplot.hexbin.

Examples

```
if(require(marray)){
  data(swirl)
  hb <- plotMAhex(swirl[,1],newpage=FALSE,
                  main = "M vs A plot with hexagons", legend=0)
  hexVP.abline(hb$plot.vp,h=0,col=gray(.6))
  hexMA.loess(hb)
}
```

---

pushHexport *Push a Hexagon Viewport ("hexVP")*

Description

Push a Hexagon Viewport ("hexVP", see hexVP-class) on to the tree of (grid) viewports, calling pushViewport.

Usage

```
pushHexport(hvp, clip = "off")
```

Arguments

hvp	a hexagon viewport, i.e., an object of class "hexVP", see hexVP-class, typically produced by hexViewport(..).
clip	which viewport to push, either 'on' or 'off' are the allowed arguments, see details.

Details

A hexagon viewport ("hexVP") object has slots for two replicate viewports one with clipping turned on and one with clipping off. This allows toggling the clipping option.

See Also

the underlying pushViewport from the **grid** package.

`smooth.hexbin` *Hexagon Bin Smoothing*

Description

Given a `"hexbin"` (hexagon bin) object, compute a discrete kernel smoother that covers seven cells, namely a center cell and its six neighbors. With two iterations the kernel effectively covers 1+6+12=19 cells.

Usage

```
smooth.hexbin(bin, wts=c(48,4,1))
```

Arguments

`bin`	object of class `"hexbin"`, typically resulting from `hexbin()` or `erode,hexbin-method`.
`wts`	numeric vector of length 3 for relative weights of the center, the six neighbor cells, and twelve second neighbors.

Details

This discrete kernel smoother uses the center cell, immediate neighbors and second neighbors to smooth the counts. The counts for each resulting cell is a linear combination of previous cell counts and weights. The weights are

1 center cell,	weight = wts[1]
6 immediate neighbors	weight = wts[2]
12 second neighbors	weight =wts[3]

If a cell, its immediate and second neighbors all have a value of `max(cnt)`, the new maximum count would be `max(cnt)*sum(wts)`. It is possible for the counts to overflow.

The domain for cells with positive counts increases. The hexbin slots `xbins`, `xbnds`, `ybnds`, and `dimen` all reflect this increase. Note that usually `dimen[2] = xbins+1`.

The intent was to provide a fast, iterated, immediate neighbor smoother. However, the current hexbin plotting routines only support shifting even numbered rows to the right. Future work can

(1) add a shift indicator to hexbin objects that indicates left or right shifting.
(2) generalize plot.hexbin() and hexagons()
(3) provide an iterated kernel.

With `wts[3]=0`, the smoother only uses the immediate neighbors. With a shift indicator the domain could increase by 2 rows (one bottom and on top) and 2 columns (one left and one right). However the current implementation increases the domain by 4 rows and 4 columns, thus reducing plotting resolution.

Value

an object of class `"smoothbin"`, extending class `"hexbin"`, see `hexbin`. The object includes the additional slot `wts`.

References

see grid.hexagons and hexbin.

See Also

hexbin, erode.hexbin, hcell2xy, gplot.hexbin, hboxplot, grid.hexagons, grid.hexlegend.

Examples

```
x <- rnorm(10000)
y <- rnorm(10000)
bin <- hexbin(x,y)
# show the smooth counts in gray level
smbin  <- smooth.hexbin(bin)
plot(smbin, main = "smooth.hexbin(.)")

# Compare the smooth and the origin
smbin1 <- smbin
smbin1@count <- as.integer(ceiling(smbin@count/sum(smbin@wts)))
plot(smbin1)
smbin2 <- smooth.hexbin(bin,wts=c(1,0,0))  # expand the domain for comparability
plot(smbin2)
```

Index