Package 'mapplots'

August 25, 2023

Type Package

Title Data Visualisation on Maps

Version 1.5.2

Date 2023-08-21

Author Hans Gerritsen

Maintainer Hans Gerritsen <hans.gerritsen@marine.ie>

Depends R (>= 2.10.0)

Suggests shapefiles

Description Create simple maps; add sub-plots like pie plots to a map or any other plot; format, plot and export gridded data. The package was developed for displaying fisheries data but most functions can be used for more generic data visualisation.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-25 09:50:02 UTC

R topics documented:

`mapplots-package` *Data visualisation on maps*

Description

Create simple maps; add sub-plots like pie plots to a map or any other plot; format, plot and export gridded data. The package was developed for displaying fisheries data but most functions can be used for more generic data visualisation. For a complete list of functions with individual help pages, use `library(help="mapplots")`.

Details

The starting point is generally the function `basemap` which creates a blank map (although most mapplots functions could be applied to any plot, not just maps). Coastlines or other features can be added to the map with the function `draw.shape`. ICES rectangles can also be displayed on the map and axes by `draw.rect`. The main purpose of this package is to visualise data on maps. For univariate data, the main functions are `draw.bubble` (bubble plots) and `draw.grid` (heat maps). Multivariate data can be displayed with `draw.barplot2D` ('square pie plots'), `draw.pie` (pie plots) and `draw.xy` (xy or barplots). Some of these have a specific function for displaying a legend: `legend.bubble`, `legend.grid` and `legend.pie`. The following functions can help to get data in the right format for plotting: `make.grid` and `make.multigrid` (to create grd objects for `draw.grid`) and `make.xyz` (to create xyz objects for `draw.barplot2D` and `draw.pie`). Finally, there is a functions to export grd objects as csv or shapefiles: `write.grid`. The remaining functions are called by the main funcions listed above and were never intended to be used directly. However, they are documented and can be called directly.

Author(s)

Hans Gerritsen

Maintainer: <hans.gerritsen@marine.ie>

add.pie *Add pie plot to existing plot*

Description

This function is used by draw.pie to add a pie plot at a specific location to an existing plot. It can be used directly but in general it is advisable to use draw.pie instead.

Usage

```
add.pie(z, x = 0, y = 0, labels = names(z), radius = 1, edges = 200, clockwise =
  TRUE, init.angle = 90, density = NULL, angle = 45, col = NULL, border = NULL,
  lty =  NULL, label.dist = 1.1, ...)
```

Arguments

z	a vector of non-negative numerical quantities. The values in z are displayed as the areas of pie slices.
x, y	the location of the centre of the pie on the x and y-scale of the existing plot.
labels	one or more expressions or character strings giving names for the slices. Other objects are coerced by as.graphicsAnnot. For empty or NA (after coercion to character) labels, no label nor pointing line is drawn.
radius	the radius of the pie in units of the y-scale
edges	the circular outline of the pie is approximated by a polygon with this many edges.
clockwise	logical indicating if slices are drawn clockwise or counter clockwise (i.e., mathematically positive direction), the former is default.
init.angle	number specifying the starting angle (in degrees) for the slices. see pie for details.
density	the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.
angle	the slope of shading lines, given as an angle in degrees (counter-clockwise).
col	a vector of colors to be used in filling or shading the slices. If missing a set of 6 pastel colours is used, unless density is specified when par("fg") is used.
border	(possibly vector) argument passed to polygon which draws each slice.
lty	(possibly vector) argument passed to polygon which draws each slice.
label.dist	distance that the label is placed away from the pie (relative to the radius)
...	graphical parameters can be given as arguments to pie. They will affect the main title and labels only.

Details

Because this function is intended to add pie plots to a map, the radius is scaled to units on the y-scale. This is more convenient than using the x-scale as 1 degree latitude is exactly 60 nautical miles. (The conversion from degrees to distance on the x-scale is less straightforward as it depends on the latitude).

Note

The function obtaines the aspect ratio of the current plot from `get.asp` in order to draw circular pies. If the plot window is re-sized and `asp` is not defined in `par` then the pies will end up being oval.

Author(s)

Adapted from the function `pie` by Hans Gerritsen

See Also

`draw.pie`

Examples

```
plot(NA,NA, xlim=c(-1,1), ylim=c(-1,1) )
add.pie(z=rpois(6,10), x=-0.5, y=0.5, radius=0.5)
add.pie(z=rpois(4,10), x=0.5, y=-0.5, radius=0.3)
```

---

barplot2D *Draw 2-dimensional barplots*

Description

This function is used by `draw.barplot2D` to add a 2-dimensional barplot at a specific location to an existing plot. It can be used directly but in general it is advisable to use `draw.barplot2D` instead.

Usage

```
barplot2D(z, x = 0, y = 0, width = 1, height = 1, colour, add = TRUE, col.frame = NULL,
lwd.frame = 1, threshold = 1.1, ...)
```

Arguments

`z`	vector, containing positive numbers (NAs are discarded), to be used as the area of the rectangles
`colour`	a vector (same length as `z`) of colors to be used in filling the rectangles
`x, y`	the location of the centre of the pie on the x and y-scale of the existing plot.
`width, height`	the width and height of the 2D barplot (user coordinate units).

add	logical, should the 2D barplot be added to an existing plot> Defaults to TRUE
col.frame	the colour of the frame of the 2D barplot
lwd.frame	the line width of the frame of the 2D barplot
threshold	the maximum acceptable aspect ratio of the rectangles
...	arguments to be passed to polygon

Details

The algorithm that determines the location of each rectangle within the 2D-barplot is as follows: 1) Start with a rectangle representing the highest value of z. 2) Try to put the first rectangle on the left. 3) If it too elongated, try to put two rectangles, on top of each other, on the left. 4) When you have placed those rectangles, proceed with the remaining rectangles.

More precisely, we choose the number of rectables to stack so as to minimize the following penalty: penalty for the first rectangle in the stack + penalty for the last where the penalty of a rectangle is: `ratio - 1.1`. where `ratio` is the ratio of the longer side by the smaller.

Author(s)

Adapted by Hans Gerritsen

References

This function was adapted from code pulbished on http://zoonek2.free.fr/UNIX/48_R/03.html (accessed 1 Jun 2012).

See Also

draw.barplot2D

Examples

```
plot(NA,NA, xlim=c(-1,1), ylim=c(-1,1) )
barplot2D(z=rpois(6,10), x=-0.5, y=0.5, width=0.75, height=0.75, colour=rainbow(6))
barplot2D(z=rpois(4,10), x=0.5, y=-0.5, width=0.5, height=0.5, colour=rainbow(4))
```

---

basemap *Draw a (blank) map*

Description

A blank map is created that has approximatedly the correct aspect ratio for its latitude.

Usage

```
basemap(xlim, ylim, xlab = "Longitude", ylab = "Latitude", bg = "lightblue", ...)
```

Arguments

xlim	the x limits (x1, x2) of the plot.
ylim	the y limits of the plot.
xlab	a label for the x axis, defaults to "Longitude".
ylab	a label for the y axis, defaults to "Latitude".
bg	background colour for the map, defaults to "lightblue".
...	other arguments to be passed to plot.default.

Details

The aspect ratio of the map is based on the approximation that the earth is a perfect sphere of 21600 nautical miles in circumference. The straight-line distance between two medidians that lie 1 degree apart is then 60 nautical miles * cos(latitude). The aspect ratio is therefore set at the inverse of the cosine of the latitude at the middle of the y-scale. If the plot window is re-sized the aspect ratio will remain correct but the background colour will not fill the full plot area.

Author(s)

Hans Gerritsen

Examples

```
data(landings)
data(coast)
xlim <- c(-11,-5.5)
ylim <- c(51.5,55.5)
basemap(xlim, ylim)
draw.shape(coast, col="cornsilk")
```

---

breaks.grid *Define breakpoints for colour scales*

Description

This function can be used to define breakpoints for use with the functions draw.grid and legend.grid

Usage

```
breaks.grid(grd, quantile = 0.975, ncol = 12, zero = TRUE)
```

Arguments

grd	an array produced by make.grid or a list produced by make.multigrid or a vector of positive values.
quantile	the maximum value of the breaks will be determined by the quantile given here. This can be used to deal with outlying values in grd. If quantile = 1 then the maximum value of the breaks will be the same as the maximum value in grd.
ncol	number of colours to be used, always one more than the number of breakpoints. Defaults to 12.
zero	logical, should zero be included as a separate category? Defaults to TRUE.

Value

a vector of breakpoints to be used by draw.grid and legend.grid

Author(s)

Hans Gerritsen

See Also

draw.grid, legend.grid

Examples

```
breaks.grid(100,ncol=6)
breaks.grid(100,ncol=5,zero=FALSE)

# create breaks on the log scale
exp(breaks.grid(log(10000),ncol=4,zero=FALSE))
```

---

coast *Shapefile of the Irish and UK coastlines*

Description

Shapefile list object (see: shapefiles) of the Irish and UK coastline.

Usage

```
data(coast)
```

Format

shapefile list object

Source

GSHHS A Global Self-consistent, Hierarchical, High-resolution Shoreline Database.

References

https://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html

`draw.barplot2D` *Draw 2-dimensional barplots in an existing plot*

Description

2-Dimensional barplots are essentially rectangular pieplots. These plots can be used to display proportional data in certain locations on a map.

Usage

```
draw.barplot2D(x, y, z, width, height, scale = F, col = NULL, col.frame = "black",
lwd.frame = 1, silent = TRUE, ...)
```

Arguments

`x, y`	vector with x and y-locations of the centre of the 2D barplots
`z`	array where the rows correspond to `x` and `y` and the columns correspond to categories to be plotted. The function `make.xyz` can be useful to create `z`.
`width`	the width of the (largest) 2D barplot (user coordinate units). This can be a single value or a vector with the same length as `x`.
`height`	the width of the (largest) 2D barplot see `width`
`scale`	logical, should the surface area of each 2d-barplot automatically be scaled to the sum of its z-values? Only works if `width` and `height` are vectors with a length of 1.
`col`	a vector (same length as the number of columns in `z`) of colors to be used in filling the rectangles
`col.frame`	the colour of the frame of the 2D barplot
`lwd.frame`	the line width of the frame of the 2D barplot
`silent`	logical, should a progress message be displayed in the console? Defaults to `FALSE`.
`...`	arguments to be passed to `barplot2D`, particlarly `border`, which determines the colour of the borders of the rectangles (as opposed to the frame around each 2D barplot which is controlled by `col.frame`. Other relevant arguments include `density` and `lwd`.

Details

The algorithm that determines the location of each rectangle within the 2D-barplot is as follows: 1) Start with a rectangle representing the highest value of z. 2) Try to put the first rectangle on the left. 3) If it too elongated, try to put two rectangles, on top of each other, on the left. 4) When you have placed those rectangles, proceed with the remaining rectangles.

More precisely, we choose the number of rectables to stack so as to minimize the following penalty: penalty for the first rectangle in the stack + penalty for the last where the penalty of a rectangle is: `ratio - 1.1`. where `ratio` is the ratio of the longer side by the smaller.

Author(s)

Adapted by Hans Gerritsen

References

This function was adapted from http://zoonek2.free.fr/UNIX/48_R/03.html

Examples

```
data(landings)
data(coast)
xlim <- c(-15,0)
ylim <- c(50,56)
xyz <- make.xyz(landings$Lon,landings$Lat,landings$LiveWeight,landings$Species)
col <- rainbow(5)
basemap(xlim, ylim, main = "Species composition of gadoid landings")
draw.shape(coast, col="cornsilk")
draw.barplot2D(xyz$x, xyz$y, xyz$z, width = 0.8, height = 0.4, col=col)
legend("topright", legend=colnames(xyz$z), fill=col, bg="lightblue", inset=0.02)

basemap(xlim, ylim, main = "Species composition of gadoid landings")
draw.shape(coast, col="cornsilk")
draw.barplot2D(xyz$x, xyz$y, xyz$z, width = 1, height = 0.5, scale=TRUE, col=col)
legend("topright", legend=colnames(xyz$z), fill=col, bg="lightblue", inset=0.02)
```

---

draw.bubble *Draw bubble plots in an existing plot*

Description

Bubble plots are plots of circles whose surfaces areas are proportional to values in z.

Usage

```
draw.bubble(x, y, z, maxradius = 1, ...)
```

Arguments

x,y	vector with x and y-locations of the centre of the bubbles.
z	vector with positive values to correspond with the surface area of the bubbles.
maxradius	the radius of the largest bubble in units of the y-scale.
...	other arguments to be passed to points

Author(s)

Hans Gerritsen

Examples

```
data(landings)
data(coast)
xlim <- c(-12,-5)
ylim <- c(50,56)
agg <- aggregate(list(z=landings$LiveWeight),list(x=landings$Lon,y=landings$Lat),sum)
basemap(xlim, ylim, main = "Gadoid landings")
draw.shape(coast, col="cornsilk")
draw.bubble(agg$x, agg$y, agg$z, maxradius=0.5, pch=21, bg="#00FF0050")
legend.z <- round(max(agg$z)/1000,0)
legend.bubble("topright", z=legend.z, maxradius=0.5, inset=0.02, bg="lightblue", txt.cex=0.8,
 pch=21, pt.bg="#00FF0050")
```

---

draw.grid *Display a grd object as a heatmap*

Description

Displays a grid of colored or gray-scale rectangles with colors corresponding to the values in z. This can be used to display three-dimensional or spatial data as images.

Usage

```
draw.grid(grd, breaks = NULL, col = NULL)
```

Arguments

grd	a 2-dimensional array with data to be plotted. The row and column names should should correspond to the x and y locations of the gridlines at which the values are displayed. see make.grid.
breaks	a vector of breakpoints for the colours, must give one more breakpoint than colour.
col	a vector with colours. defaults to 12 colors ranging from white (lowest) through yellow and orange to red (highest).

Note

The resulting maps are often referred to as heat maps although this is not strictly correct as red would imply the lowest and white the highest value (an object glowing white is hotter than an object glowing red). However using the scale in this way is counter-intuitive.

Author(s)

Hans Gerritsen

See Also

image

Examples

```
data(coast)
data(landings)
byx = 1
byy = 0.5
xlim <- c(-15.5,0)
ylim <- c(50.25,56)
grd <- make.grid(landings$Lon,landings$Lat,landings$LiveWeight, byx, byy, xlim, ylim)
breaks <- breaks.grid(grd,zero=FALSE)
basemap(xlim, ylim, main = "Gadoid landings")
draw.grid(grd,breaks)
draw.shape(coast, col="darkgreen")
legend.grid("topright", breaks=breaks/1000, type=2, inset=0.02, title="tonnes")
```

---

draw.pie *Draw pie plots in an existing plot*

Description

Draw pie plots in an existing plot

Usage

```
draw.pie(x, y, z, radius, scale = T, labels = NA, silent = TRUE, ...)
```

Arguments

x, y	vector with x and y-locations of the centre of the pies
z	array where the rows correspond to x and y and the columns correspond to categories to be plotted. The function make.xyz can be useful to create z.
radius	the radius of the (largest) pie (y-scale units). This can be a single value or a vector with the same length as x.
scale	logical, should the surface area of each pie plot automatically be scaled to the sum of its z-values? Only works if radius is a vector with a length of 1.

labels	labels for each slice. Defaults to NA, labels are probably best placed in a legend by legend.pie.
silent	logical, should a progress message be displayed in the console? Defaults to FALSE.
...	other arguments to be passed to add.pie

Author(s)

Hans Gerritsen

See Also

add.pie

Examples

```
data(landings)
data(coast)
xlim <- c(-12,-5)
ylim <- c(50,56)
xyz <- make.xyz(landings$Lon,landings$Lat,landings$LiveWeight,landings$Species)
col <- rainbow(5)
basemap(xlim, ylim, main = "Species composition of gadoid landings")
draw.shape(coast, col="cornsilk")
draw.pie(xyz$x, xyz$y, xyz$z, radius = 0.3, col=col)
legend.pie(-13.25,54.8,labels=c("cod","had","hke","pok","whg"), radius=0.3, bty="n", col=col,
 cex=0.8, label.dist=1.3)
legend.z <- round(max(rowSums(xyz$z,na.rm=TRUE))/10^6,0)
legend.bubble(-13.25,55.5,z=legend.z,round=1,maxradius=0.3,bty="n",txt.cex=0.6)
text(-13.25,56,"landings (kt)",cex=0.8)
```

---

draw.rect *Draw ICES rectangles in an existing plot*

Description

Draw ICES rectangles in an existing map and add axis labels to the top and righ axes. ICES statistical rectangles (https://www.ices.dk) are rectangles of 1 degree longitude x 0.5 degrees latitude. They are used for reporting of fishinge effort and landings.

Usage

```
draw.rect(col = "grey", lty = 2, ...)
```

Arguments

col	colour of the lines, defaults to "grey".
lty	line type, defaults to 2.
...	any other arguments to be passed to abline.

Author(s)

Hans Gerritsen

References

https://www.ices.dk

See Also

basemap

Examples

```
xlim <- c(-15,0)
ylim <- c(50,56)
basemap(xlim, ylim)
draw.rect()
```

---

draw.shape *Draw shapefiles in an existing plot*

Description

Draw shapefiles in an existing plot

Usage

```
draw.shape(shape, type = "poly", col = 1, ...)
```

Arguments

shape	a shape list object created by (shapefiles
type	type of plot desired. The following values are possible: "p" for points, "l" or "lines" for lines and "poly" (default) for polygons.
col	the colour of the points, lines or polygons
...	other arguments to be passed to points, lines or polygon

Note

The shapefile needs to have the WGS 84 Geographic Coordinate System in order to display properly on a map of longitude and latitude.

Author(s)

Hans Gerritsen

See Also

read.shapefile

Examples

```
library(shapefiles)
shp.file <- file.path(system.file(package = "mapplots", "extdata"), "Ireland")
irl <- read.shapefile(shp.file)
xlim <- c(-11,-5.5)
ylim <- c(51.5,55.5)
basemap(xlim, ylim)
draw.shape(irl, col="cornsilk")
```

---

draw.xy *Draw xy sub-plots in an existing plot*

Description

This function is intended to draw xy-plots or barplots in existing maps or other plots.

Usage

```
draw.xy(x, y, xx, yy, xlim = NULL, ylim = NULL, width = 1, height = 0.5, bg = NULL,
border = 1, type = "p", col = 1, silent = TRUE, ...)
```

Arguments

x, y	vectors with x and y-locations of the centre of the xy-plots.
xx, yy	vectors (of the same length as x) with the x and y-values of the xy-plots
xlim, ylim	the x and y-limits of the xy-plots. the same limits will apply to all xyplots. Defaults to range(xx) and range(yy).
width, height	the width and height of the xy-plot area (coordinate units of the main map or plot).
bg	background colour of the xy-plots
border	the colour of the border of the xy-plots. NA will result in no border
type	the type of plot. See plot.default for possible values. Also see Details section below
col	a single colour or a vector of colours (same length as x) for the plotting symbols.
silent	logical, should a progress message be displayed in the console? Defaults to FALSE.
...	other arguments to be passed to points

Details

type = "h" will result in plots that resemble barplots. These are not true barplots as the x-axis is not categorical. However if suitable xx data are provided and lwd is tweaked correctly for the current graphical device it will result in adequate plots.

Author(s)

Hans Gerritsen

Examples

```
data(effort)
data(coast)
xlim <- c(-12,-5)
ylim <- c(51,54)
col <- terrain.colors(12)
effort$col <- col[match(effort$Month,1:12)]
basemap(xlim, ylim, main = "Monthly trends in haddock landings and fishing effort")
draw.rect(lty=1, col=1)
draw.shape(coast, col="cornsilk")
draw.xy(effort$Lon, effort$Lat, effort$Month, effort$LiveWeight, width=1, height=0.5,
 col=effort$col, type="h",lwd=3, border=NA)
draw.xy(effort$Lon, effort$Lat, effort$Month, effort$Effort, width=1, height=0.5, col="red",
 type="l", border=NA)
draw.xy(effort$Lon, effort$Lat, effort$Month, effort$Effort, width=1, height=0.5, col="red",
 type="p",cex=0.4,pch=16, border=NA)
legend("topleft", c(month.abb,"Effort"), pch=c(rep(22,12),16), pt.bg=c(col,NA),
 pt.cex=c(rep(2,12),0.8),col=c(rep(1,12),2), lty=c(rep(NA,12),1), bg="lightblue",
 inset=0.02, title="Landings", cex=0.8)
```

---

effort *Spatially disaggregated fishing effort and landings data*

Description

Monthly fishing effort and haddock landings landings at a spatial resolution of 1 degree longitude and 0.5 degrees latitude. The data are from Irish otter trawlers in 2009.

Usage

```
data(effort)
```

Format

data frame

Source

EU Logbooks database

get.asp *Get the current aspect ratio of a plot*

Description

This function is used by draw.pie to draw circles (rather than ellipses) in existing plots.

Usage

```
get.asp()
```

Value

Returns the aspect ratio of the current plot.

Note

If the plot is re-sized get.asp is **not** automatically updated

Author(s)

Hans Gerritsen

See Also

draw.pie

Examples

```
plot(1:10,(1:10)/10)
get.asp()
```

ices.rect *Convert ICES rectangles from or to geographical coordinates*

Description

ices.rect converts the names of ICES statistical rectangles into geographical coordinates (mid-points). ices.rect2 converts geographical coordinates into ICES statistical rectangles.

Usage

```
ices.rect(rectangle)
ices.rect2(lon, lat)
```

Arguments

rectangle	a characer vector with the names of ices rectangles. Note that the code can cope with rectangle names that have been converted into numbers by helpful microsoft office software, e.g. "36E2" tends to be converted to 3600.
lon, lat	a vector with longitude and latitude (not neccesarily the mid-points of rectangles).

Value

`ices.rect` will return a data frame with the midpoints of the ICES rectangles. `ices.rect2` will return a vector with the names of the ICES rectangles.

Author(s)

Hans Gerritsen

References

ICES statistical rectangles https://www.ices.dk are rectangles of 1 degree longitude x 0.5 degrees latitude. They are used for reporting of fishinge effort and landings.

Examples

```
# rectangle names to coordinates:
ices.rect(c("36E2","3600","40D8"))

# coordinates to rectangle names:
lon <- rnorm(10,-10,2)
lat <- rnorm(10,53,1)
rect <- ices.rect2(lon,lat)
basemap(xlim=range(lon), ylim=range(lat) )
draw.rect()
points(lon,lat)
text(lon,lat,rect,cex=0.7,pos=3)
```

---

landings *Spatially disaggregated landings data*

Description

Landings data of some gadoid species (cod, haddock, hake, saithe and whiting) at a spatial resolution of 1 degree longitude and 0.5 degrees latitude. The data are from Irish otter trawlers in 2009.

Usage

```
data(landings)
```

Format

data frame

Source

EU Logbooks database

`legend.box` *Define location of a legend box.*

Description

Define a location of a legend box. Used by `legend.bubble` and `legend.pie`. This is not intended as a user function.

Usage

```
legend.box(x, y = NULL, maxradius, mab = 1.2, inset = 0, double = F)
```

Arguments

`x, y`	the x and y co-ordinates to be used to position the legend, see `legend`.
`maxradius`	the (maximum) radius of the pie or bubble to be represented in the legend.
`mab`	the margin around the bubble or pie.
`inset`	inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.
`double`	logical, should the box be double the 'normal' size to allow for pies and bubbles to be displayed in a single legend box. generally not very pretty.

Value

Returns the corner points of the legend box in user coordinates

Author(s)

Hans Gerritsen

See Also

`legend.bubble`, `legend.pie`

Examples

```
plot(1)
box <- legend.box("topleft", maxradius=0.2, inset=0.02)
rect(box[1],box[2],box[3],box[4], border="red", lwd=3, lty=2)
legend.bubble("topleft", z=10, maxradius=0.2, inset=0.02)
```

`legend.bubble` *Legend for bubble plot*

Description

Draw a legend for a bubble plot (`draw.bubble`)

Usage

```
legend.bubble(x, y = NULL, z, maxradius = 1, n = 3, round = 0, bty = "o", mab = 1.2,
bg = NULL, inset = 0, pch = 21, pt.bg = NULL, txt.cex = 1,
txt.col = NULL, font = NULL, ...)
```

Arguments

`x, y`	the x and y co-ordinates to be used to position the legend. `x` can be a keyword (e.g. `"topleft"`). See `legend`.
`z`	either a single value representing the largest bubble, in which case `n` determines how many bubbles are displayed; or a vector of z values which will be to be shown as bubbles in the legend.
`maxradius`	the radius of the largest bubble. this should match `maxradius` in `draw.bubble`.
`n`	integer giving the number of bubbles that should be shown in the legend. Defaults to 3. Only relevant when z is a single value.
`round`	integer indicating the number of decimal places to be used in the legend.
`bty`	single character indicating the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n" (no box).
`mab`	the margin around the bubble. i.e. how much space should there be between the largest bubble and the legend box. relative to the size of the bubble.
`bg`	background colour of the legend box.
`inset`	inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.
`pch`	plotting character, i.e., symbol to use. Defaults to 21.
`pt.bg`	background (fill) color for the open plot symbols given by pch=21:25.
`txt.cex`	character expansion for the legend text.
`txt.col`	colour of the legend text.
`font`	An integer which specifies which font to use for text. See `graphical parameters`
`...`	any other arguments to be passed to `points`.

Author(s)

Hans Gerritsen

Examples

```
data(landings)
data(coast)
xlim <- c(-12,-5)
ylim <- c(50,56)
agg <- aggregate(list(z=landings$LiveWeight),list(x=landings$Lon,y=landings$Lat),sum)
basemap(xlim, ylim, main = "Gadoid landings")
draw.shape(coast, col="cornsilk")
draw.bubble(agg$x, agg$y, agg$z, maxradius=0.5, pch=21, bg="#00FF0050")
legend.bubble("topright", z=round(max(agg$z)/1000,0), maxradius=0.5, inset=0.02, bg="lightblue",
 txt.cex=0.8, pch=21, pt.bg="#00FF0050")
```

`legend.grid` *Legend for*

Description

Draw a legend for a `draw.grid` plot

Usage

```
legend.grid(x, y = NULL, breaks, col ,digits = 2, suffix = "", type = 1, pch = 15,
pt.cex = 2.5, bg = "lightblue", ...)
```

Arguments

`x, y`	the x and y co-ordinates to be used to position the legend. `x` can be a keyword (e.g. `"topleft"`). See `legend`.
`breaks`	a vector of breakpoints for the colours, must give one more breakpoint than colour.
`col`	a vector with colours. defaults to colors ranging from white (lowest) through yellow and orange to red (highest).
`digits`	integer indicating the number of significant places to be used in the legend. Defaults to 2.
`suffix`	character string to be placed after the legend entries, e.g. `"kg"`.
`type`	integer specifying the legend type, `1` will result in a single value for each colour (the average value for each colour category). Type `2` will result in a range being displayed for each colour category. This is more accurate but can make the legend take up a lot of space.
`pch`	plotting character, i.e., symbol to use. This can either be a single character or an integer code for one of a set of graphics symbols. See `points`. Defaults to 15.
`pt.cex`	character expansion of the symbols. Defaults to 2.5
`bg`	background colour for the legend box
`...`	other arguments to be passed to `legend`.

Author(s)

Hans Gerritsen

See Also

draw.grid

Examples

```
data(coast)
data(landings)
byx = 1
byy = 0.5
xlim <- c(-15.5,0)
ylim <- c(50.25,56)
grd <- make.grid(landings$Lon,landings$Lat,landings$LiveWeight, byx, byy, xlim, ylim)
breaks <- breaks.grid(grd,zero=FALSE)
basemap(xlim, ylim, main = "Gadoid landings")
draw.grid(grd,breaks)
draw.shape(coast, col="darkgreen")
legend.grid("topright", breaks=breaks/1000, type=2, inset=0.02, title="tonnes")
```

---

legend.pie *Legend for pie plots*

Description

Draw a legend for a draw.pie plot.

Usage

```
legend.pie(x, y = NULL, z, labels, radius = 1, bty = "o", mab = 1.2, bg = NULL,
inset = 0, ...)
```

Arguments

x, y	the x and y co-ordinates to be used to position the legend. x can be a keyword (e.g. "topleft". See legend.
z	optional vector of non-negative numerical quantities. The values in z are displayed as the areas of pie slices in the legend.
labels	character vector (same length as z) with labels for the pies.
radius	the radius of the pie to be displayed.
bty	single character indicating the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n" (no box).
mab	the margin around the pie. i.e. how much space should there be between the pie and the legend box. relative to the size of the pie.

bg	background colour of the legend box.
inset	inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.
...	any other arguments to be passed on to add.pie.

Author(s)

Hans Gerritsen

Examples

```
data(landings)
data(coast)
xlim <- c(-12,-5)
ylim <- c(50,56)
xyz <- make.xyz(landings$Lon,landings$Lat,landings$LiveWeight,landings$Species)
col <- rainbow(5)
basemap(xlim, ylim, main = "Species composition of gadoid landings")
draw.shape(coast, col="cornsilk")
draw.pie(xyz$x, xyz$y, xyz$z, radius = 0.3, col=col)
legend.pie(-13.25,54.8,labels=c("cod","had","hke","pok","whg"), radius=0.3, bty="n", col=col,
 cex=0.8, label.dist=1.3)
legend.z <- round(max(rowSums(xyz$z,na.rm=TRUE))/10^6,0)
legend.bubble(-13.25,55.5,z=legend.z,round=1,maxradius=0.3,bty="n",txt.cex=0.6)
text(-13.25,56,"landings (kt)",cex=0.8)
```

---

make.grid *Create grd object*

Description

Create a grd object from x, y and z data. For use with draw.grid and write.grid.

Usage

```
make.grid(x, y, z, byx , byy , xlim, ylim, fun = function(x) sum(x, na.rm = T))
```

Arguments

x	a vector of x-coordinates (longitude)
y	a vector of y-coordinates (latitude; same length as x)
z	a vector of values; same length as x
byx, byy	the size of the grid cells on the x and y scale
xlim, ylim	the x and y limits of the grid. Note that the origin of the grid depends on xlim[1] and ylim[1], these values will be taken as the mid-point of the bottom-left grid-cell. See example below how to use this to create a grid that matches that of the ICES rectangles.
fun	a function to be applied to z. Defaults to sum.

Details

Due to the way fractions are stored in binary format, rounding errors can occur, e.g.:

```
as.character(seq(-5,0,by=0.8))
```

results in:

```
"-5" "-4.2" "-3.4" "-2.6" "-1.8" "-1" "-0.199999999999999"
```

this can affect the `make.grid` function although this is generally not a problem.

Value

a grd object, which is simply a 2-dimensional array with row and column names that correspond to the x and y positions of the grid.

Author(s)

Hans Gerritsen

See Also

`draw.grid`

Examples

```
data(coast)
data(landings)
byx = 1
byy = 0.5
xlim <- c(-15.5,0)
ylim <- c(50.25,56)
grd <- make.grid(landings$Lon,landings$Lat,landings$LiveWeight, byx, byy, xlim, ylim)
breaks <- breaks.grid(grd,zero=FALSE)
basemap(xlim, ylim, main = "Gadoid landings")
draw.grid(grd,breaks)
draw.shape(coast, col="darkgreen")
legend.grid("topright", breaks=breaks/1000, type=2, inset=0.02, title="tonnes")
```

---

make.multigrid *Create a grd object*

Description

Create list of grd objects from `x`, `y`, `z` and `group` data, where each level of `group` provides a separate grd object. For use with `draw.grid`.

Usage

```
make.multigrid(x, y, z, group, ...)
```

Arguments

x	a vector of x-coordinates (longitude)
y	a vector of y-coordinates (latitude; same length as x)
z	a vector of values; same length as x
group	a factor; same length as x. E.g. species, season etc.
...	other arguments to be passed to make.grid.

Value

a list of grd objects, see make.grid.

Author(s)

Hans Gerritsen

See Also

make.grid, draw.grid

Examples

```
data(coast)
data(landings)
byx = 1
byy = 0.5
xlim <- c(-12.5,-5)
ylim <- c(50.25,56)
grd <- make.multigrid(landings$Lon,landings$Lat,landings$LiveWeight, landings$Species,
 byx, byy, xlim, ylim)
breaks <- breaks.grid(grd,zero=FALSE)
par(mfrow=c(2,3),mar=c(1,1,2,1) )
for(s in names(grd) ) {
  basemap(xlim, ylim, main = s, axes=FALSE)
  draw.grid(grd[[s]],breaks)
  draw.shape(coast, col="darkgreen")
}
plot.new()
legend.grid("center",breaks=breaks/1000,type=2)
```

---

make.xyz *Create xyz object*

Description

Creata an xyz object for use with the functions draw.barplot2D and draw.pie

Usage

```
make.xyz(x, y, z, group, FUN = sum, ...)
```

Arguments

x, y	vector with x and y-locations of the 2D barplots or pies to be plotted.
z	vector (same length as x) with the values to be displayed as the areas of the pie slices or 2D barplot.
group	factor (same length as x) with groupings for the x, y, z data, e.g. year, species etc.
FUN	function to apply to the z(in case there are duplicate combinations of x, y and group).
...	optional arguments to FUN.

Author(s)

Hans Gerritsen

See Also

draw.barplot2D and draw.pie

Examples

```
data(landings)
data(coast)
xlim <- c(-12,-5)
ylim <- c(50,56)
xyz <- make.xyz(landings$Lon,landings$Lat,landings$LiveWeight,landings$Species)
col <- rainbow(5)
basemap(xlim, ylim, main = "Species composition of gadoid landings")
draw.shape(coast, col="cornsilk")
draw.pie(xyz$x, xyz$y, xyz$z, radius = 0.3, col=col)
legend.pie(-13.25,54.8,labels=c("cod","had","hke","pok","whg"), radius=0.3, bty="n",
 col=col, cex=0.8, label.dist=1.3)
legend.z <- round(max(rowSums(xyz$z,na.rm=TRUE))/10^6,0)
legend.bubble(-13.25,55.5,z=legend.z,round=1,maxradius=0.3,bty="n",txt.cex=0.6)
text(-13.25,56,"landings (kt)",cex=0.8)
```

---

progressMsg *Progress message*

Description

progress message in the R console.

Usage

```
setProgressMsg(min = 0, max = 1)
progressMsg(pm, value, round = 0)
```

Arguments

min, max	(finite) numeric values for the extremes of the progress message. Must have min < max
pm	list created by setProgressMsg
value	value for the progress message.
round	integer indicating the number of decimal places for the percentage completed. Defaults to 0.

Details

setProgressMsg sets up a list with variables used and updated by progressMsg

Value

a list

Author(s)

Hans Gerritsen

See Also

See also txtProgressBar

Examples

```
## Not run:
  pm <- setProgressMsg(0,500)
  for(i in 1:500) {
    pm<- progressMsg(pm,i)
    Sys.sleep(0.01)

  }
  rm(pm)

## End(Not run)
```

write.grid *Export a grd object as csv or shapefile*

Description

Export a grd object (created by make.grid) as csv or shapefile.

Usage

```
write.grid(grd, file, type="csv")
```

Arguments

grd	a grid object created by the function make.grid.
file	character string naming the output filename. For csv files do include the extension (e.g. "my_file.csv"); for shapefiles omit the extension (e.g. "my_file").
type	character string specifying the output file type. Must be one of "csv" (default) or "shape"

Value

A csv file with three columns corresponing to x, y and z (lon, lat, value), or a shapefile with a polygon for each cell in the grd object. The assumed projection is EPSG:4326

Author(s)

Hans Gerritsen

See Also

make.grid, write.shapefile .

Examples

```
library(shapefiles)
data(landings)
data(coast)
byx = 1
byy = 0.5
xlim <- c(-15.5,0)
ylim <- c(50.25,56)
grd <- make.grid(landings$Lon, landings$Lat, landings$LiveWeight, byx, byy, xlim, ylim)
breaks <- breaks.grid(grd,zero=FALSE)
basemap(xlim, ylim, main = "Gadoid landings")
draw.grid(grd,breaks)
draw.shape(coast, col="darkgreen")
legend.grid("topright", breaks=breaks/1000, type=2)
## Not run:
```

```
write.grid(grd,"c:/test1.csv")
write.grid(grd,"c:/test1","shape")

## End(Not run)
```

Index