

# Package ‘hplot’

June 28, 2022

**Type** Package

**Title** Contains utility functions for when using base R graphics

**Description** Contains utility functions and custom plot functions that I commonly use when Writing R code. see ?hplot for the list of functions and their individual help pages for details and examples of their use.

**Version** 0.0.1

**Date** 2021-12-23

**Author** <malcolm.haddon@gmail.com>

**Maintainer** <malcolm.haddon@gmail.com>

**Depends** R (>= 3.5.0)

**License** GPL-2

**Imports** knitr

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**NeedsCompilation** no

## R topics documented:

addlnorm . . . . .	2
addnorm . . . . .	2
categoryplot . . . . .	3
countgtzero . . . . .	4
expandmatrix . . . . .	5
getmax . . . . .	5
getmin . . . . .	6
histyear . . . . .	7
hplot . . . . .	8
inthist . . . . .	8
linept . . . . .	10
newplot . . . . .	10
parset . . . . .	11
parsyn . . . . .	12
pickbound . . . . .	12
plot1 . . . . .	13
plotnull . . . . .	14

plotprep . . . . .	14
plotxyy . . . . .	15
setplot . . . . .	17
uphist . . . . .	17
xyplotyear . . . . .	18
yearBubble . . . . .	19

<b>Index</b>	<b>21</b>
--------------	-----------

---

addlnorm	<i>addlnorm estimates a log-normal distribution from output of hist.</i>
----------	--

---

## Description

addlnorm estimates a log-normal distribution from output of a histogram of a data set.

## Usage

```
addlnorm(inhist, xdata, inc = 0.01)
```

## Arguments

inhist	is the output from a call to 'hist' (see examples)
xdata	is the data that is being plotted in the histogram.
inc	defaults to a value of 0.01; is the fine grain increment used to define the normal curve. The histogram will be coarse grained relative to this.

## Value

a 4 x N matrix of x and y values to be used to plot the fitted normal probability density function. Combined with estimates of mean(log(indata)) and log(sd(indata))

## Examples

```
egdata <- rlnorm(200,meanlog=0.075,sdlog=0.5)
outh <- hist(egdata,main="",col=2,breaks=seq(0,8,0.2))
ans <- addlnorm(outh,egdata)
lines(ans[, "x"],ans[, "y"],lwd=2,col=4)
```

---

addnorm	<i>addnorm adds a normal distribution to a histogram of a data set.</i>
---------	---

---

## Description

addnorm adds a normal distribution to a histogram of a data set. This is generally to be used to illustrate whether log-transformation normalizes a set of catch or cpue data.

## Usage

```
addnorm(inhist, xdata, inc = 0.01)
```

**Arguments**

inhist	is the output from a call to 'hist' (see examples)
xdata	is the data that is being plotted in the histogram.
inc	defaults to a value of 0.01; is the fine grain increment used to define the normal curve. The histogram will be coarse grained relative to this.

**Value**

a list with a vector of 'x' values and a vector of 'y' values (to be used to plot the fitted normal probability density function), and a vector used two called 'stats' containing the mean and sandard deviation of the input data

**Examples**

```
x <- rnorm(1000,mean=5,sd=1)
dev.new(height=6,width=4,noRStudioGD = TRUE)
par(mfrow= c(1,1),mai=c(0.5,0.5,0.3,0.05))
par(cex=0.85, mgp=c(1.5,0.35,0), font.axis=7)
outH <- hist(x,breaks=25,col=3,main="")
nline <- addnorm(outH,x)
lines(nline$x,nline$y,lwd=3,col=2)
print(nline$stats)
```

categoryplot

*categoryplot generates a bubble plot of the contents of a matrix***Description**

categoryplot generates a bubble plot of the contents of a matrix in an effort to visualize 2-D trends in the data. It must have the numeric year variable in the columns. So if using table or tapply to generate the matrix put year last in the list of variables.

**Usage**

```
categoryplot(
  x,
  xlab = "",
  ylab = "",
  mult = 0.1,
  gridx = FALSE,
  addtotal = FALSE,
  addlines = 3
)
```

**Arguments**

x	the matrix of values to be plotted
xlab	the label for the x-axis, default=""
ylab	the label for the y-axis, default=""

mult	the multiplier for the values. Should be selected so that the circles produce a visual representation of the variation in the data
gridx	should grey grid-lines be added for each year. default=FALSE
addtotal	should the sum of the year columns be printed at the top of the diagram. default=FALSE. If TRUE it prints the column totals sequentially addlines-2 lines and then addlines-1 lines above the circles
addlines	if addtotal is TRUE then a number of lines are added at the top of the plot to contain the column totals. This argument determines the number of extra lines. default=3, but if only a few then a smaller number would be more appropriate. If addtotal = FALSE, then addlines is ignored

### Value

nothing but it does generate a plot

### Examples

```
xmat <- matrix(rnorm(25,5,2),nrow=5,ncol=5,dimnames=list(1:5,1:5))
categoryplot(xmat,mult=0.03,ylab="Random Numbers",addtotal=TRUE,addline=2)
```

---

countgtzero

*countgtzero used in apply to count the number >0 in a vector*

---

### Description

countgtzero used in apply to count number >0 in a vector

### Usage

```
countgtzero(invect)
```

### Arguments

invect                      vector of values

### Value

A single value of number of values > 0

### Examples

```
## Not run:
set.seed(12346)
x <- trunc(runif(10)*10)
x
countgtzero(x) # should be 9

## End(Not run)
```

---

expandmatrix	<i>expandmatrix reshapes a matrix of values into a 3 column data.frame</i>
--------------	--

---

**Description**

expandmatrix takes an oblong matrix of values and expands it into a three column data.frame of row, column, value. This is then easier to plot as a scattergram or is used within categoryplot. It expects to have the year values in the columns = xvalues

**Usage**

```
expandmatrix(x)
```

**Arguments**

x	a matrix of values
---	--------------------

**Value**

a 3-column matrix of (rows x cols) rows from the input matrix

**Examples**

```
x <- matrix(rnorm(25,5,1),nrow=5,ncol=5,dimnames=list(1:5,1:5))
res <- expandmatrix(x)
res
```

---

getmax	<i>getmax generates the upper bound for a plot</i>
--------	--

---

**Description**

getmax generates upper bound for a plot where it is unknown whether the maximum is greater than zero or not. If > 0 then multiplying by the default mult of 1.05 works well but if the outcome is < 0 then the multiplier needs to be adjusted appropriately so the maximum is slightly higher than the maximum of the data

**Usage**

```
getmax(x, mult = 1.05)
```

**Arguments**

x	the vector of data to be tested for its maximum
mult	the multiplier for both ends, defaults to 1.05 (=0.95 if < 0)

**Value**

a suitable upper bound for a plot if required

## Examples

```
## Not run:
vect <- rnorm(10,mean=0,sd=2)
sort(vect,decreasing=TRUE)
getmax(vect,mult=1.0)
vect <- rnorm(10,mean = -5,sd = 1.5)
sort(vect,decreasing=TRUE)
getmax(vect,mult=1.0)

## End(Not run)
```

---

getmin

*getmin generates the lower bound for a plot*


---

## Description

getmin generates lower bound for a plot where it is unknown whether the minimum is less than zero or not. If less than 0 then multiplying by the default mult of 1.05 works well but if the outcome is  $> 0$  then the multiplier needs to be adjusted appropriately so the minimum is slightly lower than the minimum of the data

## Usage

```
getmin(x, mult = 1.05)
```

## Arguments

x	the vector of data to be tested for its minimum
mult	the multiplier for both ends, defaults to 1.05 (=0.95 if $>0$ )

## Value

a suitable lower bound for a plot if required

## Examples

```
## Not run:
vect <- rnorm(10,mean=0,sd=2)
sort(vect)
getmin(vect,mult=1.0)

## End(Not run)
```

---

histyear	<i>histyear plots a histogram of a given variable for each year available</i>
----------	---

---

## Description

histyear plots a histogram of a given variable for each year available

## Usage

```
histyear(
  x,
  xlimit = c(NA, NA, NA),
  pickvar = "cpue",
  years = "year",
  varlabel = "CPUE",
  vline = NA,
  plots = c(5, 5),
  normadd = TRUE,
  left = TRUE
)
```

## Arguments

x	the data.frame of data with at least a 'Year' and pickvar present
xlim	the xaxis bounds for all histograms, defaults to c(NA,NA,NA), the values would be as used in seq(xlim[1],xlim[2],xlim[3]). If the default is used then the 0 and 0.98 quantiles of the variable are used as the bounds with 25 bins
pickvar	which variable to plot each year default = 'cpue'
years	which variable name identifies the year column, default='year'
varlabel	what label to use on x-axis, default = 'CPUE'
vline	an optional vertical line to aid interpretation. If it is numeric it will be added to each plot
plots	how many plots to generate, default = c(5,5)
normadd	should a normal distribution be added to each plot. default=TRUE
left	on which side of each plot should the year and number of records be placed left=TRUE is the default. left=FALSE will place text on right

## Value

invisibly, a matrix of the year, mean value, stdev, and N number of observations. It also plots a histogram for each year and fits a normal distribution to each one.

## Examples

```
## Not run:
print("still to be developed")
# pickvar="x100nethr";years="year";varlabel="log(CPUE)";vline=NA;plots=plotnum;
# normadd=TRUE;left=FALSE;xlim=c(0,250,10)

## End(Not run)
```

---

hplot	<i>hplot: a set of functions providing aid for case graphics plus custom plots</i>
-------	--

---

### Description

The hplot package provides an array of utility functions and custom graphics functions.

### plotutils

**newplot** bare-bones plotprep, opens a new device + default par  
**parsyn** prints par command syntax to the console to be copied  
**plotprep** sets up a plotting device external to Rstudio  
**parset** simplifies the definition of main par parameters  
**setplot** writes a base graphics template to the console

---

inthist	<i>inthist a replacement for the hist and boxplot functions</i>
---------	---

---

### Description

inthist it is common to want to generate a list of counts as integers from an input vector and then plot then as columns of those counts. Alternatively, it is common to have a two-column matrix of values and counts or totals where one wants to plot columns of those counts or totals against those values. inthist allows one to enter either a vector of integers to be counted and plotted OR a matrix of values in column 1 and counts or totals in column 2. The option of rounding non-integers is available.

### Usage

```
inthist(
  x,
  col = 1,
  border = NULL,
  width = 0.9,
  xlabel = "",
  ylabel = "",
  main = "",
  lwd = 1,
  xmin = NA,
  xmax = NA,
  ymax = NA,
  plotout = TRUE,
  prop = FALSE,
  inc = 1,
  xaxis = TRUE,
  roundoff = TRUE,
  ...
)
```



## Arguments

x	a vector of integers to be counted and plotted OR a matrix of values in column 1 and counts or totals in column 2
col	the colour of the fill; defaults to black = 1, set this to 0 for an empty bar, but then give a value for border
border	the colour of the outline of each bar defaults to col
width	denotes the width of each bar; defaults to 0.9, should be >0 and <= 1
xlabel	the label for the x axis; defaults to ""
ylabel	the label for the y axis; defaults to ""
main	the title for the individual plot; defaults to ""
lwd	the line width of the border; defaults to 1
xmin	sets the lower bound for x-axis; used to match plots, defaults to NA whereupon the minimum of values is used
xmax	sets the upper bound for x axis; used with multiple plots, defaults to NA whereupon the maximum of values is used
ymax	enables external control of the maximum y value; mainly of use when plotting multiple plots together.
plotout	plot the histogram or not? Defaults to TRUE
prop	plot the proportions rather than the counts, default=FALSE
inc	sets the xaxis increment; used to customize the axis; defaults to 1.
xaxis	set to FALSE to define the xaxis outside of inthis; defaults to TRUE
roundoff	if values are not integers should they be rounded off to become integers? default=TRUE. Obviously only useful when inputting a matrix.
...	available to pass extra plot arguments, such as panel.first=grid(), or whatever to the internal plot call

## Value

a matrix of values and counts with the proportions of counts and values is returned invisibly

## Examples

```
x <- trunc(runif(1000)*10) + 1
inthis(x,col="grey",border=3,width=0.75,xlabel="Random Uniform",
       ylabel="Frequency")
x <- as.matrix(cbind(c(1,2,3,4,5,6,7,8),trunc(runif(8,1,20))))
inthis(x,col="grey",border=3,width=0.75,xlabel="integers",
       ylabel="Frequency")
```

---

linept	<i>linept adds a line and a series of points to a plot</i>
--------	--

---

### Description

linept adds both a line and a series of points to a plot but without the gaps introduced in the line when using type='b' within the base R lines function. This is simply a format issue as I do not like those gaps

### Usage

```
linept(x, y, lwd = 1, pch = 16, ...)
```

### Arguments

x	the x series of points
y	the corresponding y series of points
lwd	the line width, default=1
pch	the character used, default = 16 (a large dot)
...	and other graphics arguments typically used with either lines or points

### Value

nothing but it does add a pointed line to a plot

### Examples

```
print("wait on example data")
```

---

newplot	<i>newplot simple floating window setup a plot</i>
---------	--

---

### Description

newplot is a bare-bones setup routine to generate a plot in RStudio using a floating window. If you want to alter the default par settings then you can use either setplot() to get suitable syntax or, more simply, use parsyn() which only gives a template for the par syntax

### Usage

```
newplot(width = 6, height = 3.6, newdev = TRUE)
```

### Arguments

width	defaults to 6 inches = 15.24cm - width of plot
height	defaults to 3.6 inches = 9.144cm - height of plot
newdev	reuse a previously defined graphics device or make a new one, defaults to TRUE

**Value**

Checks for and sets up a graphics device and sets the default plotting par values. This changes the current plotting options!

**Examples**

```
## Not run:
x <- rnorm(1000,mean=0,sd=1.0)
plotprep()
hist(x,breaks=30,main="",col=2)

## End(Not run)
```

---

parset

*parset alters the current base graphics par settings*


---

**Description**

parset alters the current base graphics par settings to suit a single standard plot. It is merely here to simplify and speed the coding for exploratory base graphics. The font and its size default to 0.85 and font 7 (Times bold). The default values can be seen by typing parset with no brackets in the console. If a different set of par values are needed then the function parsyn() can be used to act as a prompt for the correct syntax. The output to the console can be copied to your script and modified to suit.

**Usage**

```
parset(
  plots = c(1, 1),
  cex = 0.75,
  font = 7,
  outmargin = c(0, 0, 0, 0),
  margin = c(0.45, 0.45, 0.05, 0.05),
  byrow = TRUE,
  ...
)
```

**Arguments**

plots	vector of number of rows and columns, defaults to c(1,1)
cex	the size of the font used, defaults to 0.85
font	the font used, defaults to 7 which is Times Bold, 6 is Times, 1 is Sans and 2 is Sans Bold.
outmargin	default=c(0,0,0,0) and defines the outer margin used by mtext
margin	default=c(0.45,0.45,0.05,0.05), which avoids whitespace but leaves plenty of room for titles
byrow	should plots be made by row (mfrow; byrow=TRUE, the default), of by column (mfcol; byrow=FALSE)
...	the generic ellipsis allowing for the inclusion of other graphics arguments such as xaxs="n", etc.

**Value**

nothing but it changes the base graphics par settings

**Examples**

```
## Not run:
parset()
parsyn()

## End(Not run)
```

---

`parsyn`

*parsyn types standard syntax for the par command to the console*

---

**Description**

`parsyn` types the standard syntax for the `par` command to the console so it can be copied and pasted into your own code.

**Usage**

```
parsyn()
```

**Value**

it writes two lines of R code to the console

**Examples**

```
## Not run:
parsyn()

## End(Not run)
```

---

`pickbound`

*pickbound selects an optimum number of rows and cols for a plot*

---

**Description**

`pickbound` enables the automatic selection of a pre-determined optimum combination of plot rows and columns to suit a number of plots up to 30. So, given a number of plots from 1 to 30 this returns a numeric dimer containing the number of rows and columns needed for `par` statement

**Usage**

```
pickbound(n)
```

**Arguments**

`n` the number of plots to be included in a combined plot

**Value**

a vector of two with the number of rows and columns for a plot

**Examples**

```
pickbound(5)
pickbound(8)
```

---

plot1

---

*plot1 a simple way to plot an xy line plot*


---

**Description**

plot1 provides a quick way to plot out a single xy line plot. It can be used with plotprep to generate a plot outside of Rstudio or by itself to generate one within Rstudio. It uses a standard par setup and permits custom labels, font, and font size (cex). It checks the spread of y and if a ymax is not given in the parameters finds the ymax and checks to see if y goes negative in which case it uses getmin, so the y-axis is set to 0 - ymax or ymin - ymax

**Usage**

```
plot1(
  x,
  y,
  xlab = "",
  ylab = "",
  type = "l",
  usefont = 7,
  cex = 0.75,
  maxy = 0,
  defpar = TRUE,
  ...
)
```

**Arguments**

x	The single vector of x data
y	the single vector of y data. If more are required they can be added separately after calling plot1.
xlab	the label for the x-axis, defaults to empty
ylab	the label for the y-axis, defaults to empty
type	the type of plot "l" is for line, the default, "p" is points. If you want both plot a line and add points afterwards.
usefont	which font to use, defaults to 7 which is Times bold
cex	the size of the fonts used. defaults to 0.85
maxy	defaults to 0, if a value is given then that value is used rather than estimating from the input y using getmax
defpar	if TRUE then plot1 will declare a par statement. If false it will expect one outside the function. In this way plot1 can be used when plotting multiple graphs, perhaps as mfrow=c(2,2)
...	room for other graphics commands like col, pch, and lwd

**Value**

nothing but it does plot a graph and changes the par setting

**Examples**

```
## Not run:
x <- rnorm(20,mean=5,sd=1)
plot1(x,x,xlabel="x-values",ylabel="yvalues")

## End(Not run)
```

---

plotnull	<i>plotnull generates an empty plot when one is needed</i>
----------	--

---

**Description**

plotnull, there are often circumstances, for example, when plotting up results from each year and each SAU, where there will be combinations of year and SAU that have no data, but to avoid a problem with the plotting it is necessary to generate an empty plot.

**Usage**

```
plotnull(msg = "")
```

**Arguments**

msg                      a message to be printed in the middle of the empty plot.

**Value**

nothing but it does generate a plot

**Examples**

```
plotnull("An empty plot")
```

---

plotprep	<i>plotprep sets up a window and the par values for a single plot</i>
----------	---

---

**Description**

plotprep sets up a window and the par values for a single plot. It checks to see if a graphics device is open and opens a new one if not. This is simply a utility function to save typing the standard syntax. Some of the defaults can be changed. Typing the name without () will provide a template for modification. If 'windows' is called repeatedly this will generate a new active graphics device each time leaving the older ones inactive but present. For quick exploratory plots this behaviour is not wanted, hence the check if an active device exists already or not.

**Usage**

```
plotprep(
  width = 6,
  height = 3.6,
  usefont = 7,
  cex = 0.85,
  newdev = TRUE,
  filename = "",
  resol = 300,
  verbose = TRUE
)
```

**Arguments**

width	defaults to 6 inches = 15.24cm - width of plot
height	defaults to 3 inches = 7.62cm - height of plot
usefont	default is 7 (bold Times) 1 sans serif, 2 sans serif bold
cex	default is 0.85, the font size font used for text in the plots
newdev	reuse a previously defined graphics device or make new one; defaults to TRUE
filename	defaults to "" = do not save to a filename. If a filename is input the last three characters will be checked and if they are not png then .png will be added.
resol	resolution of the png file, if defined, default=300
verbose	set this to FALSE to turn off the reminder to include a graphics.off() command after the plot. Default=TRUE

**Value**

Checks for and sets up a graphics device and sets the default plotting par values. This changes the current plotting options!

**Examples**

```
## Not run:
x <- rnorm(1000, mean=0, sd=1.0)
plotprep()
hist(x, breaks=30, main="", col=2)

## End(Not run)
```

---

plotxyy

---

*plotxyy plots two vectors of numbers against single x-axis*


---

**Description**

plotxyy plots two plots on the single graph so that they share the x-axis. The first series is plotted on the left vertical axis and the second on the right-hand axis.

**Usage**

```
plotxyy(
  x,
  y1,
  y2,
  xlab = "",
  ylab1 = "",
  ylab2 = "",
  cex = 0.85,
  fnt = 7,
  colour = c(1, 2),
  defpar = FALSE
)
```

**Arguments**

x	the x values
y1	the left-hand axis values
y2	the right-hand axis values
xlab	the x label, default=""
ylab1	the left-hand y label, default=""
ylab2	the right-hand y label, default=""
cex	the size of font on the axes, default=0.85
fnt	the font used on axes, default=7 (bold times)
colour	a vector of two values for the colour of each line, default=c(1,2) black and red
defpar	should the internal 'par' statement be used = defpar=TRUE, or the default=FALSE, which means the plot 'par' will be defined outside the plot.

**Value**

nothing but it plots a graph

**Examples**

```
## Not run:
x <- 1:20
yval1 <- rnorm(20,mean=5,sd=1)
yval2 <- rnorm(20,mean=10,sd=1)
plotxyy(x,yval1,yval2)

## End(Not run)
```



---

setplot

*setplot provides an example plot with defaults for a standard plot*


---

### Description

Provides an example plot with defaults for a standard plot includes details of how to generate tiff, pdf, and png versions, mtext and legends. Currently no parameters, but the function is open to development for customization of the example plot.

### Usage

```
setplot()
```

### Value

prints lines of R that will define a standard plot and can be copied into an R script.

### Examples

```
## Not run:
setplot()

## End(Not run)
```

---

uphist

*uphist a histogram with an upper limit on the x-axis*


---

### Description

uphist is merely a wrapper around the base hist function, which adds the ability to limit the upper value on the x-axis. With fisheries data it is surprisingly common to have data that has a very few extreme values that can obscure a standard plot of the data. The data are only truncated within the uphist function so any other analyses will be on all available data. If a maximum value is selected which accidentally eliminates all available data the script stops with an appropriate warning. If a value is selected which fails to eliminate any data then all data are used.

### Usage

```
uphist(x, maxval = NA, ...)
```

### Arguments

x	the vector of values to be plotted as a histogram
maxval	the maximum value to be retained in the plotted data
...	all the other arguments used by the base hist function

### Value

nothing, but it does plot a histogram

## Examples

```
## Not run:
x <- rlnorm(5000, meanlog=2, sdlog=1)
hist(x,breaks=30,main="",xlab="log-normal values")
uphist(x,breaks=30,main="",xlab="log-normal values",maxval=100)
uphist(x,breaks=30,main="",xlab="log-normal values",maxval=1000)

## End(Not run)
```

---

xyplotyear

*xyplotyear generates n year xy plots from a data.frame*


---

## Description

xyplotyear meets a common need that occurs when we have xy data from multiple years and want to plot them we can use xyplotyear. The y-label for each plot is the year of data. The numeric label at the top of each plot includes the number of observations, the gradient of the regression, if included, and the sum of the yvar for each year. The same y-axis scale is used for each plot.

## Usage

```
xyplotyear(
  x,
  yvar = "",
  xvar = "",
  year = "year",
  plotnum = c(1, 1),
  xlim = c(0, 12),
  addline = TRUE,
  origin = FALSE,
  xlab = "",
  ylab = "",
  maxy = NA
)
```

## Arguments

x	the data.frame containing the data
yvar	the character name of y-axis column in the data.frame x
xvar	the character name of x-axis column in the data.frame x
year	the name of the year variable, default="year"
plotnum	a vector of rows and cols for the plots, default=c(1,1). This assumes that the columns are filled first using mfc col
xlim	the range of the xvar to be plotted, default=c(0,12); If c(NA,NA) the xlim is set to the range of the input xvar
addline	should a linear regression be fitted and added to each plot default=TRUE
origin	should the regression pass through the origin, default=FALSE
xlab	the generic label for the x-axis, default="", if left empty the xvar name will be used

ylab	the generic label for the y-axis, default="", if left empty the yvar name will be used
maxy	is available if you wish to vary the maximum y-axis value. The default=NA, which means it will use getmax x 1.15 to find a maximum

### Value

currently nothing but it does plot a graph

### Examples

```
print("wait on internal data")
```

---

yearBubble	<i>yearBubble Generates a bubbleplot of x against Year.</i>
------------	---

---

### Description

yearBubble Generates a bubbleplot of x against Year.

### Usage

```
yearBubble(
  x,
  xlabel = "",
  ylabel = "",
  diam = 0.1,
  vline = NA,
  txt = c(4, 6, 9, 11),
  Fyear = FALSE,
  xaxis = TRUE,
  yaxis = TRUE,
  hline = FALSE,
  nozero = FALSE
)
```

### Arguments

x	a matrix of variable * Year; although it needn't be year
xlabel	defaults to nothing but allows a custom x-axis label
ylabel	defaults to nothing but allows a custom y-axis label
diam	defaults to 0.1, is a scaling factor to adjust bubble size
vline	defaults to NA but allows vertical ablines to highlight regions
txt	defaults are lines to vessel numbers, catches, catches, maximumY
Fyear	defaults to FALSE, if TRUE generates a fishing year x-axis
xaxis	defaults to TRUE, allows for a custom x-axis if desired by using something like axis(1,at=years,labels=years).
yaxis	defaults to TRUE, allows for a custom y-axis if desired by using something like axis(side=2,at=years,labels=years).

hline	defaults to FALSE
nozero	defaults to FALSE, if TRUE replaces all zeros with NA so they do not appear in the plot

**Value**

invisible, vectors of catch and vessels by year, and radii matrix

**Examples**

```
## Not run:
data(sps)
cbv <- tapply(sps$catch_kg, list(sps$Vessel, sps$Year), sum, na.rm=TRUE)/1000
dim(cbv)
early <- rowSums(cbv[,1:6], na.rm=TRUE)
late <- rowSums(cbv[,7:14], na.rm=TRUE)
cbv1 <- cbv[order(late, -early),]
plotprep(width=7, height=6)
yearBubble(cbv1, ylabel="Catch by Trawl", vline=2006.5, diam=0.2)

## End(Not run)
```

# Index

`addlnorm`, [2](#)  
`addnorm`, [2](#)  
  
`categoryplot`, [3](#)  
`countgtzero`, [4](#)  
  
`expandmatrix`, [5](#)  
  
`getmax`, [5](#)  
`getmin`, [6](#)  
  
`histyear`, [7](#)  
`hplot`, [8](#)  
  
`inthist`, [8](#)  
  
`linept`, [10](#)  
  
`newplot`, [10](#)  
  
`parset`, [11](#)  
`parsyn`, [12](#)  
`pickbound`, [12](#)  
`plot1`, [13](#)  
`plotnull`, [14](#)  
`plotprep`, [14](#)  
`plotxyy`, [15](#)  
  
`setplot`, [17](#)  
  
`uphist`, [17](#)  
  
`xyplotyear`, [18](#)  
  
`yearBubble`, [19](#)