Package 'aqfig'

July 2, 2024

Type Package

Version 0.9

Date 2024-06-27

Title Display Air Quality Model Output and Monitoring Data

Imports graphics, grDevices, stats, geoR

Suggests maps

Description
Display air quality model output and monitoring data using scatterplots, grids, and legends.

License Unlimited

LazyData true

NeedsCompilation no

Author Jenise Swall [aut, cre],
Kristen Foley [ctb]

Maintainer Jenise Swall <jswall@vcu.edu>

Repository CRAN

Date/Publication 2024-07-02 06:20:10 UTC

Contents

`aqfig-package` *Functions to help draw figures displaying air quality data and model output*

Description

This package contains several functions to help users draw figures to display air pollution measurements and air quality model output. These include functions to place a color legend to the right of a plot, to draw a scatterplot with the points' colors or sizes reflecting the value of a response variable, etc.

`kristen.colors` *Sets up a color palette which is particularly appropriate for use with the scatterplot.density function*

Description

This function sets up a color palette with `n` colors, with gradually changing hues of grey, purple, blue, green, yellow, orange, red, and brown (in this order). This palette is a particularly good choice for `scatterplot.density` function, for which it is the default choice.

Usage

```
kristen.colors(n=64)
```

Arguments

`n`	Number of colors to include in palette.

Details

This function calls `colorRampPalette` and returns `n` colors from this palette.

Value

Returns `n` colors (in hexadecimal).

Note

This function is called by default by the function `scatterplot.density`, which is included in this package.

Author(s)

Original code was authored by Kristen Foley and included in the precursor to `scatterplot.density`. Packaged into a new function by Jenise Swall.

See Also

colorRampPalette

---

ozone1 *Daily maximum ozone concentrations*

Description

This data set gives the daily maximum ozone concentrations (in ppb) observed at a subset of 23 Clean Air Status and Trends Network (CASTNET) monitoring sites on August 9, 2004.

Usage

```
ozone1
```

Format

A file in CSV format containing 23 records.

Source

Hourly ozone data at CASTNET sites can be obtained from EPA's Clean Air Status and Trends Network program at http://www.epa.gov/castnet. Thanks to Steven Howard for his help collating and summarizing these hourly data.

---

ozone2 *Daily maximum ozone concentrations*

Description

This data set gives the daily maximum ozone concentrations (in ppb) observed at a subset of 23 Clean Air Status and Trends Network (CASTNET) monitoring sites on August 10, 2004.

Usage

```
ozone2
```

Format

A file in CSV format containing 23 records.

Source

Hourly ozone data at CASTNET sites can be obtained from EPA's Clean Air Status and Trends Network program at http://www.epa.gov/castnet. Thanks to Steven Howard for his help collating and summarizing these hourly data.

plot3d.points *Make 3-D scatterplot (using colored or differently-sized points)*

Description

Given a list of points' coordinates and the values observed at those points, return a scatterplot with points located as specified by the coordinates and coded by color and/or size to represnt the observed value at the location. This code is basically a wrapper for a call to the function points.geodata in the geoR package.

Usage

```
plot3d.points(x, y, z, zlim = range(z, na.rm = TRUE),
   add = FALSE, col = heat.colors(12), xlab, ylab,
   pch = 21, cex.min = 1, cex.max = 1,
   symbol.border.col = "black",
   plt.beyond.zlim = FALSE, ...)
```

Arguments

x	x-coordinates of locations at which response values z are recorded.
y	y-coordinates of locations at which response values z are recorded.
z	Response values z observed at locations whose coordinates are given by (x, y).
zlim	Vector of minimum and maximum values of z to which to assign the two most extreme colors in the col argument (col[1] and col[length(col)]). Default is to use the range of z. This is very much like the zlim argument to the image function.
add	If FALSE (default), the function will begin a new plot. If TRUE, adds scatterplot to a pre-existing plot.
col	Color range to use for the scatterplot, with the first color assigned to zlim[1] and last color assigned to zlim[2]. Default is "heat.colors(12)", as it is for image.
xlab	The label for the x-axis. If not specified by the user, defaults to the expression the user named as parameter x. This behavior is similar to that for plot.
ylab	The label for the y-axis. If not specified by the user, defaults to the expression the user named as parameter y. This behavior is similar to that for plot.
pch	The point symbol to use. Possible values are 21, 22, 23, 24, and 25. This is because points.geodata requires these points, which have outlines around them. Default is a circle ('pch=21').
cex.min	Minimum amount to shrink/expand the point symbols.
cex.max	Maximum amount to shrink/expand the point symbols.

Parameters cex.min and cex.max control the minimum and maximum amounts to shrink/expland the points, based on the value of z. By default, these are both set to one, which makes all the points the same size. For more information, see the help page for points.geodata.

`symbol.border.col`
: This controls the color of the border around the plotting symbol. By default, it is black. If a border is not desired, use 'symbol.border.col="transparent"'.

`plt.beyond.zlim`
: IF TRUE, and if `zlim` is specified by the user, z values beyond the limits given in `zlim` are plotted. Values less than zlim[1] are plotted in the same color as zlim[1]; values greater than zlim[2] are plotted in the same color as zlim[2]. If TRUE, and `zlim` is not specified by the user, zlim[1] and zlim[2] will be assigned the minimum and maximum values of `z`. In this case, user is warned and plt.beyond.zlim is set to FALSE. Default is plt.beyond.zlim==FALSE.

`...`
: Any other parameters the user adds will be passed to the `plot` function if 'add=FALSE', and may include options for axis labels, titles, etc.

Details

This function is a wrapper to the `points.geodata` function in the geoR package.

Value

A scatterplot with points at (x,y). These points are colored according to the correspoinding value of `z` and the colors specified in `col`. They are sized according to the corresponding value of `z` and the minimum and maximum sizes specified by `cex.min` and `cex.max`.

Author(s)

Jenise Swall

See Also

`points.geodata`, `points`

Examples

```
# Plot ozone at each location using colors from rainbow.colors
# and differently-sized points.  Add a legend using function
# vertical.image.legend (included in this package).
data(ozone1)
col.rng <- rev(rainbow(n=10, start=0, end=4/6))
z.rng <- c(40, 90)
plot3d.points(x=ozone1$longitude, y=ozone1$latitude, z=ozone1$daily.max,
              xlab="longitude", ylab="latitude", col=col.rng,
              zlim=z.rng, cex.min=0.5, cex.max=1.5)
# To verify, label the points with their concentrations.
text(ozone1$longitude, ozone1$latitude+0.15, ozone1$daily.max, cex=0.7)
# If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
# Put on legend.
vertical.image.legend(col=col.rng, zlim=z.rng)


# Plot second day of ozone data.  Note that day 2 experienced a smaller
```

```
# range of concentrations, so we plot day 2 on same scale as day 1.
data(ozone1)
data(ozone2)
z.rng <- c(40, 90)
col.rng <- rev(rainbow(n=10, start=0, end=4/6))
plot3d.points(x=ozone2$longitude, y=ozone2$latitude, z=ozone2$daily.max,
              xlab="longitude", ylab="latitude", col=col.rng,
              zlim=z.rng, cex.min=0.5, cex.max=1.5)
# To verify, label the points with their concentrations.
text(ozone2$longitude, ozone2$latitude+0.15, ozone2$daily.max, cex=0.7)
# If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
vertical.image.legend(col=col.rng, zlim=z.rng)



# When some z value(s) is/are much lower/higher than the others,
# the outlying value(s) may appear in color at the extent
# of the range, with the remainder of the data clustered in one (or
# just a few) color bin(s).
x <- 1:9
y <- 1:9
z <- c(0, 47:53, 100)
col.rng <- rev(rainbow(n=7, start=0, end=4/6))
plot3d.points(x, y, z, col=col.rng)
text(x, y+0.2, z, cex=0.8)

# In vain, you might try to "fix" this by setting zlim so that the
# color range reflects the main portion of the z values.  You may
# assume that the outlying value(s) will show up in the extreme edges
# of the color range, but what will actually happen is that the
# outlying values won't be plotted.
plot3d.points(x, y, z, col=col.rng, zlim=c(47, 53))
text(x, y+0.2, z, cex=0.8)

# Instead, specify zlim to reflect the main porition of the z values,
# and set plt.beyond.zlim=TRUE.  Now, z values below zlim[1] will be
# plotted in the same color as zlim[1]; those above zlim[2] will be
# plotted like z values of zlim[2].  But, remember, now there are
# outlying values whose maginitudes cannot be easily ascertained!
plot3d.points(x, y, z, zlim=c(47, 53), col=col.rng, plt.beyond.zlim=TRUE)
text(x, y+0.2, z, cex=0.8)
```

`ragged.image` *Produces a "ragged" image plot*

Description

This code produces an image plot in the case in which there is not a known response value `z` for every possible combination of `x` and `y`. This ragged image plot is a variant of an image plot which is not complete across the entire rectangle of the gridded area.

Usage

```
ragged.image(x, y, z, zlim = range(z, na.rm = TRUE), add = FALSE,
   col = heat.colors(12), xlab, ylab, plt.beyond.zlim = FALSE, ...)
```

Arguments

x	x-coordinates of grid cell centers at which response values z are available.
y	y-coordinates of grid cell centers at which response values z are available.
z	Response values recorded at the grid cell centers whose coordinates are given by (x, y).
zlim	Vector of minimum and maximum values of z to which to assign the two most extreme colors in the 'col' argument (col[1] and col[length(col)]). Default is to use the range of z. This is very much like the 'zlim' argument to the image function.
add	If FALSE (default), the ragged image will begin a new plot. If TRUE, adds ragged image to a pre-existing plot.
col	Color range to use for the ragged image, with the first color assigned to zlim[1] and last color assigned to zlim[2]. Default is "heat.colors(12)", as it is for image.
xlab	The label for the x-axis. If not specified by the user, defaults to the expression the user named as parameter x. This behavior is similar to that for image.
ylab	The label for the y-axis. If not specified by the user, defaults to the expression the user named as parameter y. This behavior is similar to that for image.
plt.beyond.zlim	IF TRUE, and if zlim is specified by the user, z values beyond the limits given in zlim are plotted. Values less than zlim[1] are plotted in the same color as zlim[1]; values greater than zlim[2] are plotted in the same color as zlim[2]. If TRUE, and zlim is not specified by the user, zlim[1] and zlim[2] will be assigned the minimum and maximum values of z. In this case, user is warned and plt.beyond.zlim is set to FALSE. Default is plt.beyond.zlim==FALSE.
...	Any additional parameters to be passed to the image function, if add=FALSE.

Details

This code produces a ragged image plot. This is in contrast to the standard image function, which assumes that there is a known response value z for every combination of the elements of x and y, i.e. that there is a complete rectangular grid, or image. A ragged image plot is a variant of the regular image plot which is not complete across the entire rectangle. The user specifies vectors x, y, and z, such that x and y identify a portion of the grid. This function maps the vector z onto a matrix of the type needed for the image function, but has NA entries for combinations of x and y that are not listed. The NA values are not plotted by image, so a ragged image will appear.

Value

A ragged image, i.e. a portion of an image for which we have specified grid cell centers x and y.

Note

This function is slow if x, y, and z are long vectors.

Author(s)

Jenise Swall

See Also

image, heat.colors

Examples

```
# Build x, y, and z.
x <- c(1, 2, 3, 1, 2, 3)
y <- c(1, 1, 1, 2, 2, 2)
z <- 1:6
z.mat <- matrix(c(1:6), ncol=2)
col.rng <- terrain.colors(6)
# Show complete matrix.
image(x=unique(x), y=unique(y), z.mat, zlim=range(z), col=col.rng,
      xlab="x", ylab="y")
# Plot only part of this as a ragged image.  Set z range so that this
# image will use colors consistent with the previous one.
ragged.image(x=x[1:4], y=y[1:4], z=z[1:4], zlim=range(z), col=col.rng,
             xlab="x", ylab="y")



# When some z value(s) is/are much lower/higher than the others,
# the outlying value(s) may appear in color at the extent
# of the range, with the remainder of the data clustered in one (or
# just a few) color bin(s).
x <- c(1, 2, 3, 1, 3, 2, 3, 1, 3)
y <- c(4, 4, 4, 3, 3, 2, 2, 1, 1)
z <- c(0, 47:53, 100)
col.rng <- rev(rainbow(n=7, start=0, end=4/6))
ragged.image(x, y, z, col=col.rng)
text(x, y, z, cex=0.8)

# In vain, you might try to "fix" this by setting zlim so that the
# color range reflects the main portion of the z values.  You may
# assume that the outlying value(s) will show up in the extreme edges
# of the color range, but what will actually happen is that the
# outlying values won't be plotted.
ragged.image(x, y, z, col=col.rng, zlim=c(47, 53))
text(x, y, z, cex=0.8)

# Instead, specify zlim to reflect the main porition of the z values,
# and set plt.beyond.zlim=TRUE.  Now, z values below zlim[1] will be
# plotted in the same color as zlim[1]; those above zlim[2] will be
# plotted like z values of zlim[2].  But, remember, now there are
```

```
# outlying values whose maginitudes cannot be easily ascertained!
ragged.image(x, y, z, zlim=c(47, 53), col=col.rng, plt.beyond.zlim=TRUE)
text(x, y, z, cex=0.8)
```

`scatterplot.density` *Use color to show the density of points in a scatterplot*

Description

The plotting region of the scatterplot is divided into bins. The number of data points falling within each bin is summed and then plotted using the image function. This is particularly useful when there are so many points that each point cannot be distinctly identified.

Usage

```
scatterplot.density(x, y, zlim, xylim, num.bins=64,
   col=kristen.colors(32), xlab, ylab, main, density.in.percent=TRUE,
   col.regression.line=1, col.one.to.one.line=grey(0.4),
   col.bar.legend=TRUE, plt.beyond.zlim=FALSE, ...)
```

Arguments

`x`	Vector or matrix of x-coordinates of points to be plotted. Missing values are not permitted.
`y`	Vector or matrix of y-coordinates of points to be plotted. Missing values are not permitted.
`zlim`	Vector defining the minimum and maximum of the data density values, to which to assign the two most extreme colors in the `col` argument. If not specified, the range of the calculated density values to be plotted is used.
`xylim`	Specification of extreme values that the first and last bins are expected to contain in the x- and y-directions. May be a single vector of the limits for the x and y axes; e.g., using '`xylim=c(0,120)`' specifies that, in both the x- and y-directions, the first bin should contain 0 and the last contain 120. May also be a list in the form: '`xylim=list(xlim=c(x1 ,x2), ylim=c(y1, y2))`', allowing for the different ranges on the axes. If not specified, xlim is the range of `x` and ylim is the range of `y`. Note that `xylim` and `num.bins` together determine how the bins are defined. For more information, see "Details" below.
`num.bins`	Number of bins to be used when calculating the data density in both the x- and y-directions. May be a single number, e.g. '`num.bins=50`', which produces 50 bins in each direction. May also be a list in the form '`num.bins=list(num.bins.x=n1, num.bins.y=n2)`' to specify differing numbering of bins for the x- and y-directions. The default is to use 64 bins for both axes ('`num.bins=64`'). Note that `xylim` and `num.bins` together determine how the bins are defined. For more information, see "Details" below.

`col` Color range to use when drawing bins, with the first color assigned to '`zlim[1]`' and last color assigned to '`zlim[2]`'. Default is '`kristen.colors(32)`'.

`xlab` The label for the x-axis. If not specified by the user, defaults to the expression the user named as parameter `x`. This behavior is similar to that for `image`.

`ylab` The label for the y-axis. If not specified by the user, defaults to the expression the user named as parameter `y`. This behavior is similar to that for `image`.

`main` The main title for the density scatterplot. If not specified, the default is "Data Density Plot (%)" when '`density.in.percent=TRUE`', and "Data Frequency Plot (counts)" otherwise.

`density.in.percent`
A logical indicating whether the density values should represent a percentage of the total number of data points, rather than a count value. Default is '`density.in.percent=TRUE`'.

`col.regression.line`
A color number or color name for the regression line and estimated regression equation (`y` as a linear function of `x`) to be overlaid on density scatterplot. If NULL, the regression line and equation are not displayed. Defaults to a black line and equation text.

`col.one.to.one.line`
A color number or color name for the regression one-to-one line to be overlaid on density scatterplot. If NULL, the one-to-one line is not displayed. Defaults to a dark grey line. If the one-to-one line is displayed, it will be as a dashed line ('`lty=3`').

`col.bar.legend` A logical indicating whether a "color legend" of the form given by `vertical.image.legend` should be displayed. The default is '`col.bar.legend=TRUE`'.

`plt.beyond.zlim`
IF TRUE, and if `zlim` is specified by the user, density values beyond the limits given in `zlim` are plotted. Values less than '`zlim[1]`' are plotted in the same color as '`zlim[1]`'; values greater than '`zlim[2]`' are plotted in the same color as '`zlim[2]`'. If TRUE, and `zlim` is not specified by the user, '`zlim[1]`' and '`zlim[2]`' will be assigned the minimum and maximum values of `z`. In this case, user is warned and `plt.beyond.zlim` is set to FALSE. Default is '`plt.beyond.zlim=FALSE`'.

`...` Any additional parameters to be passed to the `image` function.

Details

The plotting region of the scatterplot is divided into bins. The number of data points falling within each bin is summed and then plotted using the `image` function. The default is to plot the percent of the data falling within each bin, rather than a raw count value. The arguments xylim and num.bins can include different settings for the x- and y-axis. This makes it easier to plot different variables on each axis, e.g. temperature vs. ozone. Note that `xylim` and `num.bins` together determine how the bins are defined.

Note that `xylim` and `num.bins` together determine how the bins are defined. This is done using the `cut` function. Assigning values to bins is more complicated than might be expected. For example, values that fall at cutoff points between bins are difficult to deal with. This function accepts the default setting for `cut`, which assigns values which fall on a cutoff point to the bin on the left; that

is, the intervals are open on the left and closed on the right. This means that a point with x-value equal to 'xlim[1]' and/or y-value equal to 'ylim[1]' would not be assigned to any interval, which is probably not what the user intends in this circumstance. Therefore, this code determines the number of bins in the x-direction so that 'xlim[1]' and 'xlim[2]' are at the center of the first and last bin in the x-direction (and similarly for the y-direction). This means that the first and last bins actually extend a bit past the limits specified. For most applicatons, which use large numbers of data points and bins, this shouldn't be noticeable, but it may be in smalled examples like the first one given below.

Value

A density scatterplot; that is, a pattern of shaded squares representing the counts/percentages of the points falling in each square.

Author(s)

Original version (plot.density.scatter.plot) by Kristen Foley, adapted for **aqfig** by Jenise Swall

See Also

vertical.image.legend, kristen.colors, image, cut

Examples

```
# As a simple test case, build x and y vectors consisting only of the
# integers 1-3.
x <- c( rep(1, 7), rep(2, 12), rep(3, 6) )
y <- c( rep(1, 5), rep(2, 2), rep(1, 2), rep(2, 8), rep(3, 2),
        rep(2, 2), rep(3, 4) )

# For this test case, I've totaled the counts below.
count.df <- data.frame(x=rep(1:3, each=3), y=rep(1:3, times=3), ct=c(5,
2, 0, 2, 8, 2, 0, 2, 4) )

# Make a density scatterplot with counts.
scatterplot.density(x, y, num.bins=3, col=heat.colors(7),
                    density.in.percent=FALSE,
                    col.one.to.one.line="green")
text(count.df$x, count.df$y, count.df$ct, col="purple")

# Make a density scatterplot with percentages.
scatterplot.density(x, y, num.bins=3, col=heat.colors(7), col.one.to.one.line=1)
text(count.df$x, count.df$y, count.df$ct/sum(count.df$ct))


# An example closer to actual usage.
x <- rnorm(100000,50,5)
y <- 3 + (.89*x) + rnorm(100000,0,5)
scatterplot.density(x, y)
```

`vertical.image.legend` *Put color bar legend in the right plot margin.*

Description

Put color bar legend in the right-hand side margin of an existing plot.

Usage

```
vertical.image.legend(zlim, col)
```

Arguments

`zlim`	Gives the range of z values to which the colors specified in `col` are assigned.
`col`	Gives the range of colors to use. To keep multiple plots consistent in terms of the colors assigned to various values, keep `zlim` and `col` the same for each of the plots and the legend.

Details

This function works best when there is only one plot on the device, in which case the margin space is straightforward (no confusion between `oma`/`omi` and `mar`/`mai`, etc. The user should finish making the main portion of the plot before adding the legend; i.e., the user should **add the legend last**. This function alters the `par` settings to draw the legend. Upon exit, it resets them back to what they were before the function call.

Value

Puts vertical color bar legend to the right of the plot.

Note

Putting a legend on a plot is harder than it might seem. The user may have to experiment with this function a bit to get it to work well for a specific application. The user may also want to try the `imagePlot` function in the fields package.

Author(s)

Jenise Swall

See Also

`image.plot`

Examples

```
# Plot ozone at each location using colors from rainbow.colors
# and differently-sized points.  Add a legend using function
# vertical.image.legend (included in this package).
data(ozone1)
col.rng <- rev(rainbow(n=10, start=0, end=4/6))
z.rng <- c(40, 90)
plot3d.points(x=ozone1$longitude, y=ozone1$latitude, z=ozone1$daily.max,
              xlab="longitude", ylab="latitude", zlim=z.rng, col=col.rng,
              cex.min=0.5, cex.max=1.5)
# To verify, label the points with their concentrations.
text(ozone1$longitude, ozone1$latitude+0.15, ozone1$daily.max, cex=0.7)
# If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
# Put on legend.
vertical.image.legend(col=col.rng, zlim=z.rng)
```

Index