

Description

Utilities for setting trellis themes.

Usage

```
col.brewer(n = 7,
           name = "Set1",
           favorite = 2,
           alldots = TRUE,
           filldots = TRUE,
           alpha = 1,
           background = "transparent")
col.grayscale(n=6,
             alldots = TRUE,
             filldots = TRUE,
             light = "gray60",
             dark = "gray20",
             alpha = 1,
             background = "transparent")
set.alpha(color, alpha = 0.5)
```

Arguments

<code>n</code>	number of colors in RColorBrewer palette
<code>name</code>	name of RColorBrewer palette
<code>favorite</code>	index of favorite color in the palette – used in situations where only one color is required.
<code>alldots</code>	if true, all plot characters are set to be dots
<code>filldots</code>	if true, all the dots are filled in, else open
<code>color</code>	a color or vector of colors
<code>light</code>	a light color for use in gray scale palette
<code>dark</code>	a dark color for use in gray scale palette
<code>alpha</code>	alpha level to use for colors – only works in certain graphics devices
<code>background</code>	background color – may want to set this to white for drivers that don't seem to handle "transparent"

Value

`col.brewer` returns a list suitable for use with `trellis.par.set` and is similar to `col.whitebg` but chooses colors using [RColorBrewer](#) packages. `col.grayscale` does the same but uses a grayscale palette. `set.alpha` is a utility that takes a color adds an alpha level.

Author(s)

Randall Pruim

See Also

[Lattice](#), [RColorBrewer](#), [trellis.par.set](#), [col.whitebg](#),

Examples

```
trellis.par.set(theme=col.brewer(8,"Paired"))
show.settings()
set.alpha("red",0.5)
set.alpha(rainbow(6),0.5)
```

<code>equal.width</code>	<i>function to form equal width shingle</i>
--------------------------	---

Description

This function is analagous to [equal.count](#) but attempts to form a shingle in which each level covers an equal numerical range rather than containing an equal number of data points.

Usage

```
equal.width(x, number = 6, overlap = 0.5)
```

Arguments

<code>x</code>	numeric variable or R object.
<code>number</code>	desired number of levels
<code>overlap</code>	number between 0 and 1 indicating how much overlap there should be between adjacent levels

Details

`equal.width` converts `x` to a shingle. Essentially a wrapper around `link{shingle}`.

Author(s)

Randall Pruim

See Also

[equal.width](#), [shingle](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##-- or do help(data=index) for the standard data sets.
x <- rnorm(100)
s <- equal.width(x,number=4,overlap=0.1)
summary(s)
```

`panel.leveldiag` *Replacement panel function for levelplot*

Description

Uses rectangles rotated 45 degrees to display the levelplot. This is useful for displaying the "diagonal" of a standard levelplot.

Usage

```
panel.leveldiag(x, y, z,
  subscripts,
  at = pretty(z),
  scaling=c(.5,1),
  shrink,
  labels = NULL,
  contour = FALSE,
  label.style = c("mixed", "flat", "align"),
  region = TRUE,
  col = add.line$col,
  lty = add.line$lty,
  lwd = add.line$lwd,
  cex = add.text$cex,
  font = add.text$font,
  fontfamily = add.text$fontfamily,
  fontface = add.text$fontface,
  col.text = add.text$col,
  ...,
  col.regions = regions$col,
  alpha.regions = regions$alpha)
```

Arguments

<code>x, y, z</code>	variables defining the plot. It is assumed that $x = (a+b)*scaling[1]$ and $y = (b-a)*scaling[2]$. The resulting plot is a 45-degree rotation of <code>levelplot{z~a*b}</code> .
<code>subscripts</code>	integer vector indicating what subset of <code>x</code> , <code>y</code> and <code>z</code> to draw
<code>at</code>	numeric vector specifying breakpoints for change in colors
<code>scaling</code>	numeric vector of length 2 specifying scaling factors mentioned above.

<code>shrink</code>	either a numeric vector of length 2 (meant to work as both x and y components), or a list with components x and y which are numeric vectors of length 2. This allows the rectangles to be scaled proportional to the z-value. The specification can be made separately for widths (x) and heights (y). The elements of the length 2 numeric vector gives the minimum and maximum proportion of shrinkage (corresponding to min and max of z).
<code>labels</code>	contour labels
<code>label.style</code>	controls choice of how label positions are determined.
<code>contour</code>	logical, specifying whether contour lines should be drawn. Currently this is ignored, as are all other parameters related to drawing contour lines.
<code>region</code>	logical, specifying whether inter-contour regions should be filled with the appropriate color
<code>col, lty, lwd</code>	graphical parameters for contour lines
<code>cex, col.text, font, fontfamily, fontface</code>	graphical parameters for contour labels
<code>...</code>	extra parameters
<code>col.regions</code>	a vector of colors used if <code>region=TRUE</code> . Usually a subset of the colors are used, the exact number being one more than the length of <code>at</code> . These are chosen to be roughly equally spaced along <code>col.regions</code> . In the unusual case when <code>col.regions</code> is not longer than <code>at</code> , it is repeated to be as long as necessary.
<code>alpha.regions</code>	numeric scalar controlling transparency of facets

Details

`contour=TRUE` has not yet been implemented. Otherwise, this is a drop-in replacement for `panel.levelplot`.

See Also

[panel.levelplot](#), [levelplot](#)

Examples

```
# here is an example with an equally spaced grid:
#
x=rep(1:10,each=10)
y=rep(1:10,times=10)
z=abs(x-y)
position = (x+y)/2
sum = (x+y)
distance = (y-x)
difference = (y-x)
df = as.data.frame(cbind(x,y,z,position,distance,sum,difference))
levelplot(z~x*y,df)
levelplot(z~position*distance,df,
          panel=panel.leveldiag,
          scales=list(y=list(draw=FALSE)),
```

```

        ylab="",
        ylim=c(0,4),
    )
levelplot(z~position*distance,df,
        panel=panel.leveldiag,
        ylim=c(0,4),
    )

# different scaling to work with sum instead of mean of x values
#
levelplot(z~sum*difference,df,
        panel=panel.leveldiag,
        scaling=c(1,1),
        ylim=c(0,5),
    )

# Can also be used with unequally spaced grid:
#
x=rep(c(sort(runif(10)),1),each=11)
y=rep(c(sort(runif(10)),1),times=11)
z=abs(x-y)
position = (x+y)/2
sum = (x+y)
distance = (y-x)
difference = (y-x)
df = as.data.frame(cbind(x,y,z,position,distance,sum,difference))
levelplot(z~x*y,df)
levelplot(z~position*distance,df,
        panel=panel.leveldiag,
        scales=list(y=list(draw=FALSE)),
        ylab="",
        ylim=c(0,.35),
    )

```

radplot

Radial Plot

Description

radplot is pretty much a drop-in replacement for xyplot. Radial plots use angles and radial distances rather than horizontal and vertical distances to represent the data values. Most of the arguments to xyplot have the analagous functionality in the context of radplot.

Usage

```

radplot(formula,
        data = parent.frame(),
        groups = NULL,
        xlab = "",
        ylab = "",
        ylim,

```

```
xlim,  
label.width = 1,  
...)
```

Arguments

<code>formula</code>	a formula describing the form of the plot to be generated. The syntax matches the syntax used for <code>xyplot</code> . In a formula of the form $y \sim x$, <code>y</code> is displayed as a radial distance from the center of a circle, and <code>x</code> is displayed as an angular direction from the center.
<code>data</code>	a data frame containing values for any variables in the formula, as well as groups and subset if applicable. By default the environment where the function was called from is used.
<code>groups</code>	a variable or expression to be evaluated in the data frame specified by <code>data</code> , expected to act as a grouping variable within each panel, typically used to distinguish different groups by varying graphical parameters like color and line type. Formally, if <code>groups</code> is specified, then <code>groups</code> along with subscripts is passed to the panel function, which is expected to handle these arguments. It is very common to use a key (legend) when a grouping variable is specified. See entries for arguments <code>key</code> and <code>auto.key</code> of <code>xyplot</code> and for the <code>simpleKey</code> function for how to draw a key.
<code>xlab</code>	character string or expression (or a “grob”) giving label for the x-axis. Defaults to the expression for <code>x</code> in formula. Can be specified as <code>NULL</code> to omit the label altogether. Finer control is possible, as described in the entry for <code>main</code> , with the additional feature that if the label component is omitted from the list, it is replaced by the default <code>xlab</code> . Currently this is ignored, but is here for compatibility with <code>xyplot</code> . Placing of <code>xlab</code> may be implemented in a future version.
<code>ylab</code>	See description of <code>xlab</code> .
<code>xlim</code>	Normally a numeric vector of length 2 giving minimum and maximum for the x-axis. <code>xlim</code> could also be a list, with as many components as the number of panels (recycled if necessary), (list format not yet implemented). with each component as described above.
<code>ylim</code>	See <code>xlim</code> .
<code>label.width</code>	a non-negative number indicating how much space to use for the y labels. The unit is the radius of the plotting circle. Defaults to 1 currently (until an algorithm for determining size of labels is implemented), corresponding to space for labels equal to space for plot.
<code>...</code>	additional arguments a la <code>xyplot</code> . Most additional arguments will either work as in <code>xyplot</code> or be ignored.

Note

This function is under development. The goal is to have a drop-in replacement for `xyplot`, but some features may need implementation, improvements, debugging, etc.

Author(s)

Randall Pruim

See Also

[Lattice trellis.par.set xyplot](#)

Examples

```
x <- 1:20
y <- runif(20,1,5)
z <- (y > 3)
df <- as.data.frame(cbind(x,y,z))
radplot(y~x|z,df)
```