

# Package ‘squash’

July 23, 2025

**Version** 1.0.9

**Date** 2020-02-19

**Title** Color-Based Plots for Multivariate Visualization

**Author** Aron C. Eklund

**Maintainer** Aron C. Eklund <aroneklund@gmail.com>

**Imports** graphics, grDevices, methods, stats

**Description**

Functions for color-based visualization of multivariate data, i.e. colorgrams or heatmaps. Lower-level functions map numeric values to colors, display a matrix as an array of colors, and draw color keys. Higher-level plotting functions generate a bivariate histogram, a dendrogram aligned with a color-coded matrix, a triangular distance matrix, and more.

**License** Artistic-2.0

**URL** <https://github.com/aroneklund/squash>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-02-20 07:00:05 UTC

## Contents

cimage . . . . .	2
cmap . . . . .	4
colorgram . . . . .	5
ColorPalettes . . . . .	7
corrogram . . . . .	9
dendromat . . . . .	10
diamond . . . . .	12
distogram . . . . .	12
hist2 . . . . .	13
hkey . . . . .	15
makecmap . . . . .	16
matapply . . . . .	17
prettyInt . . . . .	19

savemat . . . . .	20
squashgram . . . . .	21
trianglegram . . . . .	23
xyzmat.coords . . . . .	24
xyzmat2xyz . . . . .	25
<b>Index</b>	<b>26</b>

---

cimage	<i>Draw a matrix of colored rectangles</i>
--------	--

---

**Description**

Draw a matrix of colored rectangles, possibly of varying sizes.

**Usage**

```
cimage(x = NULL, y = NULL, zcol = NULL, zsize = 1,  
       xlab = NULL, ylab = NULL, xlabels = NULL, ylabels = NULL,  
       border = NA, add = FALSE, axes = TRUE, useRaster = FALSE, ...)
```

**Arguments**

x	Vector of rectangle midpoints or breakpoints along X-axis (corresponding to the columns of zcol).
y	Vector of rectangle midpoints or breakpoints along Y-axis (corresponding to the rows of zcol).
zcol	Matrix of colors for each rectangle, e.g. RGB values or integer indices.
zsize	Relative size for each rectangle, ranging from 0 to 1. Will be recycled if necessary.
xlab, ylab	Labels for the axes.
xlabels, ylabels	Categorical labels for rows/columns.
border	Color for rectangle borders.
add	Add to the current plot instead of creating a new one?
axes	Draw axes on the plot?
useRaster	TRUE = draw a true raster image (using <a href="#">rasterImage</a> ). FALSE = draw a series of individual rectangles.
...	Further arguments passed to <a href="#">plot</a> .

## Details

Data (x, y, and zcol) can be passed to this function in any format recognized by [xyzmat.coords](#).

This function is somewhat similar to the function [image](#), except that the colors are specified explicitly, and the size of each rectangle can be adjusted.

If xlabels is NULL (the default), standard numeric axes are drawn on the X-axis. If xlabels is TRUE, the rownames of zcol are placed below each column. Otherwise, xlabels is taken as a vector of labels to be placed below each column. Likewise for ylabels and the Y-axis.

Using useRaster=TRUE can reduce the file size for large matrices drawn to vector-based graphics output such as PDFs. However, the output may look strange with smaller matrices on graphics devices that do smoothing by default (such as PDF output viewed in Preview).

## Value

None.

## Note

Currently, this function will may not behave as expected if the x and/or y values are specified as midpoints and are not evenly spaced.

## See Also

[image](#) and [rasterImage](#) provide somewhat similar functionality.

This function is called by [colorgram](#), which accepts a numeric (rather than color) matrix as input.

The package **pixmap** may be more suitable for plotting images that are not data-driven (e.g. external files).

## Examples

```
## Visualize nearly all built-in R colors
color.mat <- matrix(colors()[1:625], nrow = 25)
cimage(zcol = color.mat)

## An example using "zsize"
x <- y <- 1:10
zcolor <- matrix( rainbow(100)[outer(x, y)], nrow = 10 )
zsize <- matrix( runif(100), nrow = 10 )
cimage(x, y, zcol = zcolor, zsize = zsize)

## Another simple example
red <- green <- 0:255
rg <- outer(red, green, rgb, blue = 1, maxColorValue = 255)
cimage(red, green, zcol = rg)

## The same, but using useRaster (resulting in faster image generation,
## and smaller file size if saved as a PDF)
cimage(red, green, zcol = rg, useRaster = TRUE)

## An example with categorical axes
```

```

colormixer <- function(x, y) {
  r <- (col2rgb(x) + col2rgb(y)) / 2
  rgb(as.data.frame(t(r)), maxColorValue = 255)
}
set.seed(123)
x <- sample(colors(), 15)
y <- sample(colors(), 10)
mix <- outer(x, y, colormixer)
op <- par(mar = c(8, 8, 2, 2), las = 2)
cimage(zcol = mix, xlabels = x, ylabels = y, xlab = NA, ylab = NA)
par(op)

## An example with non-uniform midpoints and breakpoints
rg2 <- rg[seq(1, 255, by = 62), seq(1, 255, by = 62)]
cimage(x = (1:5)^2, y = c(3, 5, 6, 9, 10, 11), zcol = rg2,
       zsize = matrix(runif(25, min = 0.5), nrow = 5))

```

---

cmap

*Apply a color map to numeric data*


---

## Description

Map numeric (scalars, vectors, matrices) into colors, (optionally) using a specified color map.

## Usage

```
cmap(x, map, outlier = NULL, ...)
```

## Arguments

x	Something numeric (vector, matrix).
map	The color map to use (as created by <a href="#">makecmap</a> ). If missing, a color map is created.
outlier	Color for values outside the map domain, or NULL to generate an error in case of such values (see Details).
...	Arguments passed to <a href="#">makecmap</a> , if map is undefined.

## Details

Values in x outside the domain of map cause either an error (if outlier=NULL) or a warning (otherwise).

## Value

Something of the same size as x. May be character (RGB) or integer (palettes) depending on the color map used. Dimensions and dimnames are preserved.

**See Also**

[makecmap](#). Also, [as.raster](#) and [level.colors](#) have similar functionality.

**Examples**

```
x <- y <- 1:50
mat1 <- outer(x, y)

## several ways of visualizing the matrix mat1:
plot(col(mat1), row(mat1), col = cmap(mat1), pch = 16)

cimage(x, y, zcol = cmap(mat1))

colorgram(x = x, y = y, z = mat1)

## treatment of out-of-domain values
map <- makecmap(0:100, colFn = greyscale)
x <- y <- -10:10
mat2 <- outer(x, y, "*")

## Not run:
## Values outside the domain of "map" generate an error...
plot(col(mat2), row(mat2), col = cmap(mat2, map), pch = 15, cex = 2)

## ... unless we specify "outlier", but this still generates a warning
plot(col(mat2), row(mat2), col = cmap(mat2, map, outlier = 'red'), pch = 15, cex = 2)

## End(Not run)
```

---

colorgram

---

*Draw a colorgram (heatmap) of a matrix*


---

**Description**

Plot a visual representation of a numeric matrix using colors to indicate values.

**Usage**

```
colorgram(x = NULL, y = NULL, z = NULL, zsize = 1,
          map, nz = 10, breaks = pretty, symm = FALSE, base = NA, colFn = jet,
          key = hkey, key.args = list(),
          xlab = NULL, ylab = NULL, zlab = NULL,
          outlier = NULL, ...)
```

## Arguments

<code>x, y</code>	Locations of grid lines at which the values in <code>z</code> are measured. These must be finite, non-missing and in (strictly) ascending order. (see Details below)
<code>z</code>	A numeric matrix containing the values to be visualized as colors (NAs are allowed). Note that <code>x</code> can be used instead of <code>z</code> for convenience.
<code>zsize</code>	A numeric matrix specifying the relative size of each rectangle.
<code>map</code>	A list, as generated by <a href="#">makecmap</a> . If missing, a color map is generated automatically.
<code>nz, breaks, symm, base, colFn</code>	Arguments passed to <a href="#">makecmap</a> , if <code>map</code> is missing.
<code>key</code>	A function to draw a color key, such as <a href="#">hkey</a> or <a href="#">vkey</a> .
<code>key.args</code>	Arguments passed to the function given by <code>key</code> .
<code>xlab, ylab</code>	Labels for axes.
<code>zlab</code>	Label (title) for the color key.
<code>outlier</code>	Color for values outside the map domain. If NULL, values falling outside the map domain will generate an error.
<code>...</code>	Further arguments passed to <a href="#">cimage</a> .

## Details

This function assigns colors to the elements of a matrix and plots it using [cimage](#).

Data can be passed to this function in any format recognized by [xyzmat.coords](#).

`colorgram` is somewhat similar to [image](#). However, `colorgram` adds the following functionality: 1. The value-to-color mapping can be specified (thus allowing unequal bin sizes). 2. A color key can be added, optionally. 3. A color can be specified for missing values. 4. The size of each grid rectangle can be adjusted to convey additional information.

Two color key functions are provided in the `squash` package: 1) [hkey](#) draws a horizontal key, in the lower-left corner by default. 2) [vkey](#) draws a vertical key, in the lower-right corner by default. The latter usually looks better if the right-hand margin is increased. These keys can be controlled somewhat using `key.args`. However, that `title` and `map` cannot be specified in `key.args`; use the `zlab` and `map` arguments instead.

## Value

Invisibly, `map`.

## See Also

If this is not quite what you are looking for, consider [image](#), [filled.contour](#), or [levelplot](#). Also `color2D.matplot` in the `plotrix` package.

## Examples

```
## median Petal.Length as function of Sepal.Length and Sepal.Width
pl <- matapply( iris[,1:3], FUN = median, nx = 20, ny = 15 )

## Draw a colorgram with the default horizontal color key
colorgram(pl, main = 'iris')

## ... or with the vertical color key
colorgram(pl, main = 'iris', key = vkey)

## ... add margin space to improve legibility
op <- par(mar = c(5,4,4,4)+0.1)
colorgram(pl, main = 'iris', key = vkey,
  key.args = list(skip = 2), zlab = 'Petal\nlength')
par(op)

## Here is the example from the base function "persp"
x <- seq(-10, 10, length= 30)
y <- x
f <- function(x,y) { r <- sqrt(x^2+y^2); 10 * sin(r)/(r) }
z <- outer(x, y, f)
colorgram(x, y, z)

## ... and with a slight fix to the key:
colorgram(x, y, z, key.args = list(wh = c(1, 4, 14)))

## We could also make more space for the key:
op <- par(mar = c(7,4,4,2)+0.1)
colorgram(x, y, z, key.args = list(stretch = 3))
par(op)

## Here are some alternatives to colorgram
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
image(x, y, z)
contour(x, y, z)

## Use 'xlabels' and 'ylabels' to create categorical axes
colorgram(t(mtcars[,c(2,8:11)]), colFn = heat,
  xlabels = TRUE, ylabels = TRUE,
  xlab = NA, ylab = NA, zlab = 'Value',
  main = 'Motor car specifications', las = 1)
```

## Description

Generate a vector of contiguous colors of a specified length.

## Usage

```
rainbow2(n)
jet(n)
heat(n)
coolheat(n)
blueorange(n)
bluered(n)
darkbluered(n)
greyscale(n, start = 0.9, end = 0)
grayscale(n, start = 0.9, end = 0)
```

## Arguments

n	Number of colors to return.
start, end	Levels of gray (1 = white, 0 = black).

## Details

rainbow2 is a variation of [rainbow](#), in which the colors do not cycle completely around. Thus, rainbow2 may be less ambiguous as a color scale.

jet is similar to the Matlab color scheme of the same name and is taken from an example in [colorRamp](#).

heat is similar to [heat.colors](#), but starts at black rather than red.

coolheat is the diverging version of heat, running from cyan to black to yellow.

blueorange and bluered range from blue to grey to orange (or red), and are intended to be used as diverging color scales.

darkbluered ranges from dark blue to grey to dark red, and is intended to be used as a diverging color scale that emphasizes the magnitude more than the sign.

greyscale or grayscale ranges from off-white to black.

## Value

A vector of RGB colors.

## See Also

Standard R palettes such as [rainbow](#).

Custom palettes can be generated with [colorRamp](#).

## Examples

```
## Present the squash palettes along with the built-in R palettes
squash.palettes <- c('rainbow2', 'jet', 'grayscale', 'heat',
                    'coolheat', 'blueorange', 'bluered', 'darkbluered')
R.palettes <- c('rainbow', 'heat.colors', 'terrain.colors', 'topo.colors', 'cm.colors')

plot(0:8, type = 'n', ann = FALSE, axes = FALSE)
```



```

for (i in 1:5) {
  p <- R.palettes[i]
  hkey(makecmap(c(0, 9), colFn = get(p)),
        title = p, x = 2, y = i - 1)
}
for (i in 1:8) {
  p <- squash.palettes[i]
  hkey(makecmap(c(0, 9), colFn = get(p)),
        title = p, x = 6, y = i - 1)
}
text(3, 8, 'R palettes', font = 2)
text(7, 8, 'squash palettes', font = 2)

```

---

corrogram

*Draw a color-coded triangular matrix of pairwise correlations*


---

## Description

This figure is a color-coded, rotated triangular matrix indicating the correlation between every pair of items.

## Usage

```
corrogram(...)
```

## Arguments

... Arguments passed to [distogram](#).

## Details

This is a simple wrapper around [distogram](#), with the color scale set by default to use [blueorange](#) with a range from -1 to +1.

## Value

A color map (as generated by [makecmap](#)), invisibly.

## See Also

[distogram](#)

## Examples

```
corrogram(cor(swiss), title = 'Pearson correlation')
```

dendromat

*Plot a dendrogram with a colorgram underneath*

## Description

Plot a dendrogram with a colorgram underneath. The colorgram typically indicates characteristics about each element in the dendrogram.

## Usage

```
dendromat(x, mat,
          labRow = rownames(mat), labCol = colnames(mat),
          height = NA, gap = 0, matlabside = 2, border = NA,
          cex.lab = par('cex.axis'), ...)
```

## Arguments

<code>x</code>	An object of type <a href="#">hclust</a> or <a href="#">dendrogram</a> .
<code>mat</code>	A matrix or data frame of colors, with each row corresponding to an item in the dendrogram.
<code>labRow</code>	Labels of items, to be placed underneath the matrix.
<code>labCol</code>	Labels for characteristics, to be placed next to the matrix.
<code>height</code>	Fraction of the plot area to reserve for the color matrix. If NA, the spacing is set automatically.
<code>gap</code>	Extra space (in lines) to add between the dendrogram and the matrix.
<code>matlabside</code>	Which side of the matrix to put <code>labCol</code> (2 or 4).
<code>border</code>	Border color for the color matrix.
<code>cex.lab</code>	Relative text size for the item labels.
<code>...</code>	Further arguments passed to <a href="#">plot.dendrogram</a> .

## Details

The order of `labRow` and the rows of `mat` should correspond to the input to [hclust](#) (or whatever function created `x`). This function reorders `mat` and `labRow` to match the dendrogram, using [order.dendrogram](#).

This function combines two plots using [layout](#); therefore it is incompatible with other multiple-plot schemes (e.g. `par(mfrow)`).

If `height == NA` (the default), the function tries to leave enough room for the item labels at the bottom, and enough room for the color matrix in the middle. The leftover plotting area on the top is used for the dendrogram. The lower margin setting (see [par](#)) is ignored.

If `labRow` is set to NULL, or is equal to NULL because `mat` lacks rownames, then the item labels are taken from `x` instead.

**Value**

none.

**Note**

Currently, horizontal dendrograms are not supported.

After dendromat is finished, the user coordinates are set to  $c(0, 1, 0, 1)$ .

**See Also**

[heatmap](#)

**Examples**

```
## Motor Trend car road test data
mt.dend <- hclust(dist(mtcars[,1:7]))
mt.mat <- mtcars[,8:11]

## A minimal dendromat
dendromat(mt.dend, mt.mat)

## The same plot, but with a few enhancements
names(mt.mat) <- c('Straight', 'Manual', '# gears', '# carbs')
dendromat(mt.dend, mt.mat, gap = 0.5, border = 'gray', las = 2,
  ylab = 'Euclidean distance',
  main = 'mtcars, clustered by performance')
legend('topright', legend = 0:8, fill = 0:8)

## US state data, with color keys
us.dend <- hclust(dist(scale(state.x77)))

income <- state.x77[, 'Income']
frost <- state.x77[, 'Frost']
murder <- state.x77[, 'Murder']

income.cmap <- makecmap(income, n = 5, colFn = colorRampPalette(c('black', 'green')))
frost.cmap <- makecmap(frost, n = 5, colFn = colorRampPalette(c('black', 'blue')))
murder.cmap <- makecmap(murder, n = 5, colFn = colorRampPalette(c('black', 'red')))

us.mat <- data.frame(Frost = cmap(frost, frost.cmap),
  Murder = cmap(murder, murder.cmap),
  Income = cmap(income, income.cmap))

par(mar = c(5,4,4,3)+0.1)
dendromat(us.dend, us.mat,
  ylab = 'Distance', main = 'US states')

vkey(frost.cmap, 'Frost')
vkey(murder.cmap, 'Murder', y = 0.3)
vkey(income.cmap, 'Income', y = 0.7)
```

---

diamond	<i>Draw diamonds</i>
---------	----------------------

---

### Description

Draw diamonds on the graphics device.

### Usage

```
diamond(x, y = NULL, radius, ...)
```

### Arguments

x, y	Position(s) of the centers of the diamonds.
radius	Distances from the center to the vertex.
...	Further arguments passed to <a href="#">polygon</a> (e.g. col, border).

### Details

x and y can be passed to diamond in any form recognized by [xy.coords](#) (e.g. individual vectors, list, data frame, formula).

Only “square” (equilateral) diamonds are implemented here.

### See Also

[rect](#)

### Examples

```
plot(1:10)
diamond(1:10, rep(3, 10), radius = 0.4)
diamond(3, 8, 1, border = 3)
diamond(1:10, rep(5, 10), radius = seq(0.1, 1, length = 10), col = 1:10)
```

---

distogram	<i>Draw a color-coded triangular distance matrix</i>
-----------	--

---

### Description

This function draws a color-coded, rotated triangular matrix indicating the “distance” between every pair of items.

### Usage

```
distogram(x, map,
  n = 10, base = NA, colFn = heat,
  key = TRUE, title = NA, ...)
```

**Arguments**

x	A <a href="#">dist</a> object, or a square numeric matrix.
map	A color map, as generated by <a href="#">makecmap</a> (optional).
n, base, colFn	Arguments passed to <a href="#">makecmap</a> , if map is omitted.
key	Add a color key?
title	Title for the color key.
...	Further arguments passed to <a href="#">trianglegram</a> , (e.g. labels).

**Details**

If the input x is a matrix, the lower triangle is extracted by default (but see the arguments for [trianglegram](#)).

**Value**

The color map, invisibly.

**See Also**

[corrogram](#)

**Examples**

```
## Distances between European cities
distogram(eurodist, title = 'Distance (km)')

## Some variations
map <- distogram(eurodist, key = FALSE, colFn = jet, right = TRUE)
vkey(map, title = 'Distance (km)', x = -8)
```

---

hist2

*Bivariate histogram*


---

**Description**

Calculate data for a bivariate histogram and (optionally) plot it as a colorgram.

**Usage**

```
hist2(x, y = NULL,
      nx = 50, ny = nx,
      xlim = NULL, ylim = NULL,
      xbreaks = NULL, ybreaks = NULL,
      plot = TRUE,
      xlab = NULL, ylab = NULL, zlab = "Counts",
      colFn = heat, breaks = prettyInt, ...)
```

### Arguments

<code>x, y</code>	Numeric vectors.
<code>nx, ny</code>	Approximate number of intervals along x and y axes.
<code>xlim, ylim</code>	Limit the range of data points considered.
<code>xbreaks, ybreaks</code>	Breakpoints between bins along x and y axes.
<code>plot</code>	Plot the histogram?
<code>xlab, ylab</code>	Axis labels.
<code>zlab</code>	Label for the color key.
<code>colFn, breaks</code>	Color key parameters; see <a href="#">makecmap</a> .
<code>...</code>	Further arguments passed to <a href="#">colorgram</a> .

### Details

Data can be passed to `hist2` in any form recognized by [xy.coords](#) (e.g. individual vectors, list, data frame, formula).

### Value

Invisibly, a list with components:

<code>x</code>	Vector of breakpoints along the x-axis.
<code>y</code>	Vector of breakpoints along the y-axis.
<code>z</code>	Matrix of counts.
<code>xlab</code>	A label for the x-axis.
<code>ylab</code>	A label for the y-axis.
<code>zlab</code>	A label for the color key.

### See Also

[hist](#), for a standard (univariate) histogram.

`hist2d` in the **gplots** package for another implementation.

The **hexbin** package, for a hexagonal implementation.

### Examples

```
set.seed(123)
x <- rnorm(10000)
y <- rnorm(10000) + x
hist2(x, y)

## pseudo-log-scale color breaks:
hist2(x, y, breaks = prettyLog, key.args = list(stretch = 4))

## log-scale color breaks; the old way using 'base'
```

```
## (notice box removal to make space for the vertical color key)
hist2(x, y, base = 2, key = vkey, nz = 5, bty = 'l')
```

hkey

*Add a color key to a plot*

## Description

Add a horizontal or vertical color key to a plot

## Usage

```
hkey(map, title = NA, side = 1, stretch = 1.4, x, y, skip, wh)
vkey(map, title = NA, side = 2, stretch = 1.4, x, y, skip, wh)
```

## Arguments

map	A list, as generated by <a href="#">makecmap</a> .
title	Title for the key.
side	Where to place the labels. (1 or 3 for hkey, 2 or 4 for vkey)
stretch	Aspect ratio of the color rectangles.
x, y	Position of lower left corner of the color rectangles. If missing, the key will be placed automatically in the lower-left (hkey) or lower-right (vkey) corner of the figure region.
skip	Omit every skip labels (optional).
wh	Integer indices indicating which labels to include (optional).

## Details

This functions tries to label as many breakpoints as possible, but if the labels would overlap a subset of labels is chosen automatically. If this doesn't look right, the subset of labels can be specified with either skip or wh.

Clipping is turned off, so the key can be placed anywhere in the figure region, including the margins.

## Examples

```
attach(iris)
map <- makecmap(Petal.Length)
pl.color <- cmap(Petal.Length, map = map)

plot(Sepal.Length, Sepal.Width, col = pl.color, pch = 16)
hkey(map, title = 'Petal length (hkey default)')
hkey(map, title = 'Another hkey', x = 3.8, y = 4.7, stretch = 3)
```

```
## looks bad with default margins
vkey(map, title = 'vkey default')

vkey(map, title = 'Small vkey', x = 7.8, y = 4, stretch = 0.3)
```

makecmap

*Generate a color map from numeric values to colors***Description**

Generate a color map from numeric values to a contiguous set of colors.

**Usage**

```
makecmap(x, n = 10, breaks = pretty,
  symm = FALSE, base = NA,
  colFn = jet, col.na = NA,
  right = FALSE, include.lowest = FALSE, ...)
```

**Arguments**

x	A vector of numbers (only the finite range is used).
n	Approximate number of color levels desired.
breaks	A function to generate breakpoints, or the breakpoints themselves.
symm	Extend the mapping domain to be symmetric around zero?
base	Base for log scale, or NA to use a linear scale.
colFn	A function that generates contiguous colors.
col.na	Color to use for missing values.
right	Logical; if TRUE, the intervals will be closed on the right (and open on the left).
include.lowest	Logical, indicating if an $x[i]$ equal to the lowest (or highest, for <code>right = FALSE</code> ) breaks value should be included.
...	Further arguments to breaks.

**Details**

The general point of this function is to automatically generate a mapping that can be used in combination with [cmap](#) to represent numeric data with colors in a consistent way.

`colFn` should be a function that returns a vector of colors of specified length, such as [rainbow](#), [greyscale](#). Custom functions of this type can be generated with [colorRampPalette](#).

The breakpoints can be specified explicitly by setting `breaks` to a vector of numbers, in which case `x` is ignored. Otherwise, the breakpoints are chosen to be nice, relatively round values (using [pretty](#), or another function passed to `breaks`) covering the finite range of `x`.

If `symm` is TRUE, the map domain is extended such that it is symmetric around zero. This can be useful when using divergent color palettes to ensure that the zero point is a neutral color.

If `base` is specified, the breakpoints are generated using log-transformed data. However, setting `breaks = prettyLog` might be preferable.



**Value**

A list with the following components:

breaks	Breakpoints (numeric vector).
colors	Colors (character or numeric vector).
base	(as supplied in arguments)
col.na	(as supplied in arguments)
right	(as supplied in arguments)
include.lowest	(as supplied in arguments)

**See Also**

[cmap](#) and [colorgram](#) use the mappings generated by this function.

[hkey](#) plots a color key.

Consider setting breaks = [prettyInt](#) or breaks = [prettyLog](#)

**Examples**

```
attach(iris)
map1 <- makecmap(Petal.Length)
myColors <- cmap(Petal.Length, map = map1)
plot(Sepal.Length, Sepal.Width, col = myColors, pch = 16)
hkey(map1, title = 'Petal.Length')

## Compare the 'breaks' element in the following:
x <- rnorm(100) * 1000
str(makecmap(x))
str(makecmap(x, breaks = c(-Inf, -1000, 0, 1000, Inf)))
str(makecmap(x, breaks = prettyLog))
```

---

matapply

*Apply a function over z coordinates, binned by their x, y coordinates*

---

**Description**

Divide the range of x and y into intervals, thus forming a matrix of bins, and apply an arbitrary function to the z values corresponding to each bin.

**Usage**

```
matapply(x, y = NULL, z = NULL, FUN,
         nx = 50, ny = nx,
         xlim = NULL, ylim = NULL,
         xbreaks = NULL, ybreaks = NULL,
         right = FALSE, include.lowest = TRUE, ...)
```

**Arguments**

<code>x, y, z</code>	Numeric vectors, or possibly a matrix.
<code>FUN</code>	Function to summarize <code>z</code> values.
<code>nx, ny</code>	Approximate number of bins along <code>x</code> and <code>y</code> axis.
<code>xlim, ylim</code>	Limit the range of data points considered.
<code>xbreaks, ybreaks</code>	Breakpoints between bins along <code>x</code> and <code>y</code> axes.
<code>right</code>	Logical; if <code>TRUE</code> , the intervals will be closed on the right (and open on the left).
<code>include.lowest</code>	Logical, indicating if an <code>x[i]</code> equal to the lowest (or highest, for <code>right = FALSE</code> ) breaks value should be included.
<code>...</code>	Further arguments to <code>FUN</code> .

**Details**

`x`, `y` and `z` values can be passed to `squash` in any form recognized by `xyz.coords` (e.g. individual vectors, list, data frame, formula).

Alternatively, data that is already in a matrix can be passed in any format recognized by `xyzmat.coords`.

`FUN` should accept a numeric vector and return a single numeric value (e.g. mean, median, min, max, sd).

If `xbreaks` is not specified, approximately `nx` breakpoints will be generated automatically to span the data; likewise for `ybreaks` and `ny`.

The output can be visualized with `cologram`, `image`, etc.

**Value**

A list with components

<code>x</code>	Vector of breakpoints along the <code>x</code> -axis.
<code>y</code>	Vector of breakpoints along the <code>y</code> -axis.
<code>z</code>	Matrix of values representing the summary for each bin.
<code>xlab</code>	A label for the <code>x</code> -axis.
<code>ylab</code>	A label for the <code>y</code> -axis.
<code>zlab</code>	A label for the <code>z</code> -axis.

**Note**

The defaults of `right` and `include.lowest` are opposite the defaults used in `cut`.

**See Also**

This function is essentially a souped-up version of `tapply`.  
[squashgram](#) has similar functionality but with graphical output.

**Examples**

```
## earthquake depths as a function of longitude, latitude
attach(quakes)
quakedepth <- matapply(depth ~ long + lat, FUN = mean)
colorgram(quakedepth)

## iris petal length vs. sepal length and width
ipl <- matapply(iris[,1:3], FUN = median, nx = 20, ny = 15 )
colorgram(ipl, main = 'iris')

## Example of matrix input; here used to downsample an image
colorgram(volcano, colFn = terrain.colors)
volcano2 <- matapply(volcano, FUN = mean, nx = 20)
colorgram(volcano2, colFn = terrain.colors)
```

prettyInt

*Pretty breakpoints***Description**

Compute a sequence of around *n* values covering the range of *x*. These functions are variations of the standard R function [pretty](#).

**Usage**

```
prettyInt(x, n = 5, ...)
prettyLog(x, n = 5, small = NA, logrange = c(-100, 100))
```

**Arguments**

<i>x</i>	Numeric vector.
<i>n</i>	Approximate number of values to return.
<i>small</i>	Value below which distinction from zero is unimportant.
<i>logrange</i>	Log (base 10) of the range of values to consider as possible breakpoints.
<i>...</i>	Further arguments passed to <a href="#">pretty</a> .

**Details**

`prettyInt` returns integer values, even if this forces the number of values returned to be much lower than the requested number *n*. However, at least two values will be returned.

`prettyLog` returns values that are approximately evenly spaced on a log scale, such as (1, 3, 10, 30, ...) or (1, 2, 5, 10, 20, 50, ...) or (1, 10, 100, ...). Negative or zero values in *x* are accomodated by series such as (-100, -10, -1, 0, 1, 10, 100, ...). Setting the parameter *small* to a non-NA value will ignore *x* with absolute values below *small*.

**Value**

A numeric vector.

**See Also**

[pretty](#)

**Examples**

```
##
x1 <- 1:3
pretty(x1)
prettyInt(x1)
prettyLog(x1)

##
x2 <- pi ^ (1:8)
range(x2)
pretty(x2)
prettyLog(x2)
prettyLog(x2, n = 10)

##
x3 <- c(-x2, x2)
pretty(x3)
prettyLog(x3)
prettyLog(x3, small = 100)
```

---

savemat

---

*Save a matrix as a raster image file*


---

**Description**

Save a matrix as a PNG, TIFF, BMP, JPEG, or PDF image file, such that each pixel corresponds to exactly one element of the matrix.

**Usage**

```
savemat(x, filename, map = NULL, outlier = NULL,
        dev = c('png', 'pdf', 'bmp', 'tiff', 'jpeg'),
        do.dev.off = TRUE, ...)
```

**Arguments**

x	A matrix
filename	Filename
map	(Optional) a list, as generated by <a href="#">makecmap</a> .

outlier	(Optional) A color for outliers, if map is specified.
dev	Which graphics device to use.
...	Further arguments passed to the graphics device; see <a href="#">png</a> or <a href="#">pdf</a> .
do.dev.off	Close graphics device when finished?

### Details

This function is a relatively simple wrapper around the usual graphics device with the same name as dev. The idea is to provide an easy way of creating an image file from a matrix, without axes, plotting frame, labels, etc.

For all choices of dev except "pdf", the output image dimensions are set to match the matrix size, such that each pixel corresponds to an element of the matrix.

If map is NULL (the default), the matrix is interpreted as a matrix of colors.

If map is specified, it is used to translate the numeric matrix x into a matrix of colors, using [cmap](#).

### Value

None.

### See Also

[cimage](#) for drawing a matrix on the screen.

### Examples

```
## Not run:
big.color.matrix <- matrix(rep(colors()[1:625], 16), nrow = 100)

## save as a PNG
savemat(big.color.matrix, file = 'test.png')

## End(Not run)
```

---

squashgram

---

*Visualize a function of z coordinates, binned by x, y coordinates*


---

### Description

This is a convenience function combining `matapply` and `colorgram`. 3-dimensional data is summarized in 2-dimensional bins and represented as a color matrix. Optionally, the number of observations in each bin is indicated by relative size of the matrix elements.

**Usage**

```
squashgram(x, y = NULL, z = NULL, FUN,
           nx = 50, ny = nx, xlim = NULL, ylim = NULL,
           xbreaks = NULL, ybreaks = NULL,
           xlab = NULL, ylab = NULL, zlab = NULL,
           shrink = 0, ...)
```

**Arguments**

x, y, z	Numeric vectors; see Details.
FUN	Function to summarize z values.
nx, ny	Approximate number of bins along x and y axis.
xlim, ylim	Limit the range of data points considered.
xbreaks, ybreaks	Breakpoints between bins along x and y axes.
xlab, ylab	Axis labels.
zlab	Label for color key.
shrink	Rectangle shrinkage cutoff.
...	Further arguments passed to <a href="#">colorgram</a> .

**Details**

This function may be useful for visualizing the dependence of a variable (z) on two other variables (x and y).

x, y and z values can be passed to squash in any form recognized by [xyz.coords](#) (e.g. individual vectors, list, data frame, formula).

This function calls [matapply](#) and plots the result along with a color key.

If non-zero, the shrink parameter reduces the size of rectangles for the bins in which the number of samples is smaller than shrink. This may be useful to reduce the visual impact of less reliable observations.

**Value**

None.

**See Also**

The lower-level functions [matapply](#) and [colorgram](#).

**Examples**

```
## earthquake depths in Fiji
attach(quakes)
squashgram(depth ~ long + lat, FUN = mean)

## iris measurements
```

```

attach(iris)
squashgram(Sepal.Length, Sepal.Width, Petal.Length,
  FUN = median, nx = 20, ny = 15)

## Here indicate sample size by size of rectangles
squashgram(iris[,1:3], FUN = median,
  nx = 20, ny = 15, shrink = 5)

## What is the trend in my noisy 3-dimensional data?
set.seed(123)
x <- rnorm(10000)
y <- rnorm(10000)
z <- rnorm(10000) + cos(x) + abs(y / 4)
squashgram(x, y, z, median, colFn = bluered, shrink = 5)

```

---

trianglegram

---

*Draw a color-coded triangular matrix*


---

## Description

This function is called by [distogram](#), and probably isn't very useful by itself.

## Usage

```

trianglegram(x, labels = rownames(x),
  lower = TRUE, diag = FALSE, right = FALSE,
  add = FALSE, xpos = 0, ypos = 0, xlim, ylim, ...)

```

## Arguments

x	A square matrix containing color values.
labels	Labels.
lower	If TRUE, use <a href="#">lower.tri</a> , else use <a href="#">upper.tri</a> .
diag	Include the diagonal elements of x?
right	Should triangle point to the right or left?
add	Add to an existing plot?
xpos, ypos	Location of bottom point of the triangle.
xlim, ylim	Plotting limits.
...	Further arguments passed to <a href="#">plot</a> .

## Details

The input must be a (square) matrix; however, only part of the matrix (the upper or lower triangle) is displayed.

**Value**

none.

**See Also**

[distogram](#), [corrogram](#)

**Examples**

```
m <- matrix(jet(40), nrow = 20, ncol = 20)
trianglegram(m)

## just for fun
trianglegram(m, labels = NA, right = TRUE, add = TRUE, xpos = 1)
```

---

xyzmat.coords

*Extract (x, y, z) coordinates, where z is a matrix*

---

**Description**

Extract (x, y, z) plotting coordinates, where z is a matrix.

**Usage**

```
xyzmat.coords(x = NULL, y = NULL, z = NULL,
             xlab = NULL, ylab = NULL, zlab = NULL,
             xds = NULL, yds = NULL, zds = NULL)
```

**Arguments**

x, y	Numeric vectors.
z	A matrix
xlab, ylab, zlab	Labels
xds, yds, zds	Results from <code>deparse(substitute(x))</code> (etc.); see below.

**Details**

This function is similar to [xyz.coords](#), except that this function accepts a matrix for z.

If x is the same length as `nrow(z)`, x will be taken as the points at which the z values were sampled. If x is the length of `nrow(z) + 1`, x is taken as the breakpoints between bins. If x is missing, the matrix indices (`1:nrow(z)`) will be used. Similarly for y and the columns of z.

For convenience, the matrix can be supplied as the x argument. Or, x can be a list with elements including {x, y, z, xlab, ylab, zlab}.

When this function is used inside a higher-level plotting function, the arguments xds, yds, and zds should be set to `deparse(substitute(x))` (etc.) so that the function can generate informative default axis labels. For example, see the code for [colorgram](#).



**Value**

A list with the following components:

x	X coordinates
y	Y coordinates
z	Z matrix
xlab	Label for X axis
ylab	Label for Y axis
zlab	Label for Z axis

**Examples**

```
##
str(volcano)
volcano.xyzmat <- xyzmat.coords(volcano)
str(volcano.xyzmat)
```

---

xyzmat2xyz

---

*Convert (x, y, zmat) coordinates to (x, y, z) coordinates*


---

**Description**

Convert a matrix of Z coordinates into (x, y, z) triples.

**Usage**

```
xyzmat2xyz(...)
```

**Arguments**

... Arguments passed to [xyzmat.coords](#)

**Details**

The input is based on [xyzmat.coords](#).

The output is as returned by [xyz.coords](#)

**Value**

A list; see [xyz.coords](#).

**Examples**

```
##
str(volcano)
volcano.xyz <- xyzmat2xyz(volcano)
str(volcano.xyz)
```

# Index

- \* **aplot**
  - diamond, [12](#)
  - hkey, [15](#)
  - trianglegram, [23](#)
- \* **color**
  - cmap, [4](#)
  - ColorPalettes, [7](#)
  - hkey, [15](#)
  - makecmap, [16](#)
- \* **dplot**
  - prettyInt, [19](#)
- \* **hplot**
  - cimage, [2](#)
  - colorgram, [5](#)
  - corrogram, [9](#)
  - dendromat, [10](#)
  - distogram, [12](#)
  - hist2, [13](#)
  - squashgram, [21](#)
- \* **manip**
  - xyzmat.coords, [24](#)
  - xyzmat2xyz, [25](#)
- \* **misc**
  - matapply, [17](#)
  - savemat, [20](#)
- as.raster, [5](#)
- blueorange, [9](#)
- blueorange (ColorPalettes), [7](#)
- bluered (ColorPalettes), [7](#)
- cimage, [2](#), [6](#), [21](#)
- cmap, [4](#), [16](#), [17](#), [21](#)
- colorgram, [3](#), [5](#), [14](#), [17](#), [18](#), [22](#), [24](#)
- ColorPalettes, [7](#)
- colorRamp, [8](#)
- colorRampPalette, [16](#)
- coolheat (ColorPalettes), [7](#)
- corrogram, [9](#), [13](#), [24](#)
- cut, [18](#)
- darkbluered (ColorPalettes), [7](#)
- dendrogram, [10](#)
- dendromat, [10](#)
- diamond, [12](#)
- dist, [13](#)
- distogram, [9](#), [12](#), [23](#), [24](#)
- filled.contour, [6](#)
- grayscale (ColorPalettes), [7](#)
- greyscale, [16](#)
- greyscale (ColorPalettes), [7](#)
- hclust, [10](#)
- heat (ColorPalettes), [7](#)
- heat.colors, [8](#)
- heatmap, [11](#)
- hist, [14](#)
- hist2, [13](#)
- hkey, [6](#), [15](#), [17](#)
- image, [3](#), [6](#), [18](#)
- jet (ColorPalettes), [7](#)
- layout, [10](#)
- level.colors, [5](#)
- levelplot, [6](#)
- lower.tri, [23](#)
- makecmap, [4–6](#), [9](#), [13–15](#), [16](#), [20](#)
- matapply, [17](#), [22](#)
- order.dendrogram, [10](#)
- par, [10](#)
- pdf, [21](#)
- plot, [2](#), [23](#)
- plot.dendrogram, [10](#)

png, [21](#)  
polygon, [12](#)  
pretty, [16](#), [19](#), [20](#)  
prettyInt, [17](#), [19](#)  
prettyLog, [17](#)  
prettyLog (prettyInt), [19](#)  
  
rainbow, [8](#), [16](#)  
rainbow2 (ColorPalettes), [7](#)  
rasterImage, [2](#), [3](#)  
rect, [12](#)  
  
savemat, [20](#)  
squashgram, [18](#), [21](#)  
  
tapply, [18](#)  
trianglegram, [13](#), [23](#)  
  
upper.tri, [23](#)  
  
vkey, [6](#)  
vkey (hkey), [15](#)  
  
xy.coords, [12](#), [14](#)  
xyz.coords, [18](#), [22](#), [24](#), [25](#)  
xyzmat.coords, [3](#), [6](#), [18](#), [24](#), [25](#)  
xyzmat2xyz, [25](#)