

# Package ‘IDPmisc’

January 2, 2012

**Type** Package

**Title** Utilities of Institute of Data Analyses and Process Design ([www.idp.zhaw.ch](http://www.idp.zhaw.ch))

**Version** 1.1.16

**Date** 2011-11-09

**Author** Rene Locher, Andreas Ruckstuhl et al.

**Maintainer** Rene Locher <[rene.locher@zhaw.ch](mailto:rene.locher@zhaw.ch)>

**Description** The IDPmisc package contains different high-level graphics functions for displaying large datasets, displaying circular data in a very flexible way, finding local maxima, brewing color ramps, drawing nice arrows, zooming 2D-plots, creating figures with differently colored margin and plot region. In addition, the package contains auxiliary functions for data manipulation like omitting observations with irregular values or selecting data by logical vectors, which include NAs. Other functions are especially useful in spectroscopy and analyses of environmental data: robust baseline fitting, finding peaks in spectra.

**LazyLoad** yes

**LazyData** yes

**Depends** R(>= 2.13.1), methods, grid(>= 2.13.1), lattice(>= 0.19-26)

**Suggests** SwissAir(>= 1.0.9), PROcess

**License** GPL (>= 3)

**Repository** CRAN

**Date/Publication** 2011-11-09 15:08:35

## R topics documented:

IDPmisc-package . . . . .	2
Arrows . . . . .	3
cart2clock . . . . .	5
clock2cart . . . . .	6



Package: IDPmisc  
 Type: Package  
 Version: 1.1.16  
 Date: 2011-11-09  
 License: GPL (>= 3) (GNU Public Licence)

### Author(s)

Rene Locher, Andreas Ruckstuhl et al.

Maintainer: Rene Locher <rene.locher@zhaw.ch>

---

Arrows

*Pretty Open or Closed Arrows*

---

### Description

Draws a set of open or closed arrows which can be shaped by many arguments. Arrows is an extended version of [p.arrows](#).

### Usage

```
Arrows(x1, y1, x2, y2, size = 1, width = 1.2/4/cin, open = TRUE,
       sh.adj = 0.1, sh.lwd = 1, sh.col = if (is.R()) par("fg") else 1,
       sh.lty = 1,
       h.col = sh.col, h.col.bo = sh.col, h.lwd = sh.lwd, h.lty = sh.lty,
       verbose = FALSE)
```

### Arguments

<code>x1, y1</code>	Coordinates of points <b>from</b> which to draw.
<code>x2, y2</code>	Coordinates of points <b>to</b> which to draw.
<code>size</code>	Head size as a fraction of a character height.
<code>width</code>	Width of the arrow head. See argument <code>verbose</code> .
<code>open</code>	Defines if arrows are open or closed.
<code>sh.adj</code>	Defines gap between end of the shaft and the top of the head of the arrow (see details)
<code>sh.lwd</code>	Thickness of shaft. cf. <a href="#">par</a>
<code>sh.col</code>	Color of shaft. cf. <a href="#">par</a>
<code>sh.lty</code>	Line type of shaft. cf. <a href="#">par</a>
<code>h.col</code>	Color of head.

<code>h.col.bo</code>	Color of border of head.
<code>h.lwd</code>	Line width of border of head.
<code>h.lty</code>	Line type of border of head.
<code>verbose</code>	When TRUE, the width used is returned.

### Details

Definition of `sh.adj`:

- `=0` End of shaft at top of head,
- `=1` End of shaft at bottom of head,
- `>1` Gap between shaft and head,
- `<0` Head is on the shaft.

This function is based on [graphics](#)

### Value

A value is only returned, when `verbose == TRUE`.

### Note

The plotting device should not be resized manually after plotting as this changes in general the aspect ratio of the plot and deforms hereby the plotted arrows. The beauty of the arrows depends from the resolution of the device. The higher resolutions give better results.

### Author(s)

Andreas Ruckstuhl, refined by Rene Locher

### See Also

[p.arrows](#), [arrows](#)

### Examples

```
## a bunch of different arrows
plot(c(0,10), c(0,10), type="n")
Arrows(5, 5, 5,10, size=3,
       sh.lwd=5, sh.lty=2,
       h.lwd=5)
Arrows(5, 5, 7.5, 9, size=3, open=TRUE,
       sh.adj=0.7, sh.lwd=5, sh.lty=2,
       h.col.bo="red",h.lwd=5)
Arrows(5, 5, 9, 7.5, size=3, open=FALSE,
       sh.adj=1, sh.lwd=5, sh.col="blue",
       h.col.bo="red",h.lwd=2)
Arrows(5, 5, 10, 5, size=2.5, width=1.5, open=FALSE,
       sh.adj=1, sh.lwd=7, sh.col="blue")
Arrows(5, 5, 9, 2.5, size=4, open=FALSE,
       sh.lty=0,
```

```
      h.col.bo="black",h.lwd=5)
Arrows(5, 5, 7.5, 1)
Arrows(5, 5, 5, 0, size=2)
Arrows(5, 5, 2.5, 1, size=2, width=1)

## vector field
x<- runif( 20)
y<- runif( 20)
u<- 0.1+0.02*rnorm(20)
v<- 0.1+0.02*rnorm(20)
plot(x,y,xlim=range(c(x,x+u)),ylim=range(c(y,y+v)),type="n")
Arrows(x,y,x+u,y+v,sh.col="blue")
```

---

cart2clock

*Convert Cartesian Coordinates to Clock Coordinates*

---

### Description

Converts cartesian coordinates (x, y to clock coordinates (rho, phi)

### Usage

```
cart2clock(x, y, circle)
```

### Arguments

x, y	Cartesian coordinates.
circle	Defines the full circle in the units of phi.

### Details

Be aware that in clock coordinates and polar coordinate the angle phi is differently defined!

### Value

Data frame with

rho	Distance of point from center of coordinate system.
phi	Angle between North (12 o' clock), center and the point, measured clockwise.

### Author(s)

Rene Locher

### See Also

[clock2cart](#), [rose-class](#)

**Examples**

```
## convert clock coordinates to cartesian coordinates
xy <- clock2cart(rho=rep(1,33),phi=seq(0,to=360,length.out=33),circle=360)

## convert the cartesian coordinates back to clock coordinates
rhophi <- cart2clock(xy$x,xy$y,circle=360)

round(clock2cart(rhophi,circle=360)-xy)
## QED
```

---

clock2cart

*Convert Clock Coordinates to Cartesian Coordinates*


---

**Description**

Converts clock coordinates (rho, phi) to cartesian coordinates (x, y).

**Usage**

```
clock2cart(rho, phi, circle)
```

**Arguments**

rho	Distance of point from center of coordinate system.
phi	Angle between North (12 o' clock), center and the point, measured clockwise.
circle	Defines the full circle in the units of phi.

**Details**

Be aware that in clock coordinates and polar coordinate the angle phi is differently defined!

**Value**

Data frame with

x, y	Cartesian coordinates.
------	------------------------

**Author(s)**

Rene Locher

**See Also**

[cart2clock](#), [rose-class](#)

**Examples**

```
## an easy way to plot a circle
xy <- clock2cart(rho=rep(1,33),phi=seq(0,to=360,length.out=33),circle=360)
plot(xy)
```

---

col2hsv	<i>Convert Color to hsv Code</i>
---------	----------------------------------

---

**Description**

Converts color names or values to hsv code.

**Usage**

```
col2hsv(col)
```

**Arguments**

col	Vector of color code.
-----	-----------------------

**Value**

Matrix with the rows

h	Hue of hsv-Signal: 0=red, 1/3=green, 2/3=blue, 1=red.
s	Saturation of hsv-Signal: 0=white, 1=full color.
v	Value of hsv-Signal: 0=black, 1=full color.

**Author(s)**

Rene Locher

**See Also**

[col2rgb](#), [rgb2hsv](#), [hsv](#)

**Examples**

```
col2hsv(c("blue", "green", "red"))
```

---

`data.sheet`*Coerce a list to a data.frame*

---

**Description**

Coerces a list with vectors of different length into a `data.frame`. Fills the shorter vectors with NA.

**Usage**

```
data.sheet(x)
```

**Arguments**

`x` List to be converted.

**Details**

This function is convenient for comparing / controlling data in Lists whose components should have approximately the same length.

**Value**

`data.frame`

**Author(s)**

Thomas Unternaehrer

**Examples**

```
data.sheet(list(a=1:5,b=1:4))
```

---

`draw.leg`*Produce a Legend or Key (Grid Function)*

---

**Description**

Produces (and possibly draws) a Grid frame grob which is a legend that can be placed in other Grid plots. `draw.leg` is a slightly enhanced version of `draw.key` in package **lattice** V 0.12-3.

**Usage**

```
draw.leg(key, draw=FALSE, vp=NULL)
```

## Arguments

key	A list determining the key. See details below and the documentation for <a href="#">xyplot</a> .
draw	logical, whether the grob is to be drawn.
vp	viewport

## Details

Three new components are added to the list key of the original code in package **lattice** V 0.12-3: `between.rows`, `between.title`, `adj.title`. They allow to format the legend in a more flexible way. To ease the use of `draw.leg`, the full description of `draw.key` and the here interesting part of `xyplot` are also included:

The key essentially consists of a number of columns, possibly divided into blocks, each containing some rows. The contents of the key are determined by (possibly repeated) components named “rectangles”, “lines”, “points” or “text”. Each of these must be lists with relevant graphical parameters (see later) controlling their appearance. The key list itself can contain graphical parameters, these would be used if relevant graphical components are omitted from the other components.

The length (number of rows) of each such column (except “text”s) is taken to be the largest of the lengths of the graphical components, including the ones specified outside (see the entry for `rep` below for details on this). The “text” component has to have a character or expression vector as its first component, and the length of this vector determines the number of rows.

The graphical components that can be included in key (and also in the components named “text”, “lines”, “points” and “rectangles” as appropriate) are:

- `cex=1`
- `col="black"`
- `lty=1`
- `lwd=1`
- `font=1`
- `fontface`
- `fontfamily`
- `pch=8`
- `adj=0`
- `type="l"`
- `size=5`
- `angle=0`
- `density=-1`

`adj`, `angle` and `density` are currently unimplemented. `size` determines the width of columns of rectangles and lines in character widths. `type` is relevant for lines; “l” denotes a line, “p” denotes a point, and “b” and “o” both denote both together.

Other possible components of key are:

`between` numeric vector giving the amount of space (character widths) surrounding each column (split equally on both sides),

**title** string or expression giving a title for the key  
**rep** logical, defaults to TRUE. By default, it's assumed that all columns in the key (except the "text"s) will have the same number of rows, and all components are replicated to be as long as the longest. This can be suppressed by specifying `rep=FALSE`, in which case the length of each column will be determined by components of that column alone.  
**cex.title** cex for the title  
**background** background color, defaults to default background  
**border** either a color for the border, or a logical. In the latter case, the border color is black if border is TRUE, and no border is drawn if it is FALSE (the default)  
**transparent=FALSE** logical, whether key area should have a transparent background  
**columns** the number of columns column-blocks the key is to be divided into, which are drawn side by side.  
**between.columns** Space between column blocks, in addition to between.  
**between.rows** New argument: Space between rows.  
**between.title** New argument: Space between top row and title of legend.  
**adj.title** New argument: Adjustment of title in respect to body of legend.  
     **adj.title = 0** adjusts the title to the left,  
     **adj.title = 1** adjusts the title to the right and  
     **adj.title = 0.5 (=default)** centers the title.  
**divide** Number of point symbols to divide each line when type is "b" or "o" in lines.

**Value**

A Grid frame object (that inherits from "grob")

**Author(s)**

Deepayan Sarkar, modified by Rene Locher

**See Also**

[xyplot](#)

**Examples**

```

## Legend 2 cm below and 1 cm left of the upper right corner
grid.newpage()
key <- draw.leg(key = list(rectangles = list(col=1:3,
      size = 2,
      lwd = 0.5),
      text=list(LETTERS[1:3]),
      cex=1.2,
      between=2,
      between.rows=0.5,
      between.title=0.7,
      title = "component",
      cex.title = 1.4,

```

```

        transparent = TRUE))

vp.key <- viewport(x = convertX(unit(1,"npc")-unit(1,"cm"),"cm"),
                  y = convertY(unit(1,"npc")-unit(2,"cm"),"cm"),
                  width=grobWidth(key),
                  height=grobHeight(key),
                  just=c("right","top"))

pushViewport(vp.key)
grid.draw(key)
popViewport()

##-----
## Legend 1 cm above the lower left corner
key <- draw.leg(key = list(rectangles = list(col=1:3,
                                           size = 4,
                                           lwd = 0.5),
                          text=list(letters[1:3]),
                          lines=list(col=1:3),
                          cex=2,
                          between=2,
                          between.rows=0.5,
                          between.title=0.7,
                          title = "component",
                          adj.title = 0,
                          cex.title = 2.4,
                          transparent = TRUE))

vp.key <- viewport(x = 0,
                  y = unit(1,"cm"),
                  width=grobWidth(key),
                  height=grobHeight(key),
                  just=c("left","bottom"))

grid.newpage()
pushViewport(vp.key)
grid.draw(key)
popViewport()

```

---

general.control

*Auxiliary for Controlling the General Appearance of a Rose Plot*


---

### Description

Defines the General Appearance of rose plot.

### Usage

```

general.control(stacked = FALSE,
               rose.rad = NULL, rose.x = NULL, rose.y = NULL,

```

```
mar = rep(0.3, 4),
rev.col = FALSE,
shift = 0,
cex = 1, col = NULL, lty = 1:3, lwd = 1, type = "s")
```

### Arguments

stacked	<p>stacked = FALSE: For each point (<math>x@rho[i,j]</math>, <math>x@cyclVar[i]</math>) with <math>x</math> equal to a <a href="#">rose</a> object, the radius is <math>x@rho[i,j] - ray.lim[1]</math> as defined in <a href="#">grid.control</a>. Points with identical <math>j</math> are connected by a colored line.</p> <p>stacked = TRUE: For each point (<math>x@rho[i,j]</math>, <math>x@cyclVar[i]</math>), the radius is <math>\text{sum}(x@rho[1:i,j])</math>. No negative values are allowed in the stacked case as this feature makes sense only for variables like proportions, counts or concentrations. Areas between two adjacent <math>j</math> are filled by color.</p> <p>NA values in <math>x@rho</math> are interpreted as 0 and a warning is issued.</p>
rose.rad, rose.x, rose.y	Approximate length of radius, x- and y-position of rose. Default units are mm but any valid grid unit might be chosen (cf. <a href="#">unit</a> ). If one or more of these arguments are defined, the user of the plot function must make sure that the size of the viewport is large enough to show the complete rose and the legend. These arguments are especially useful when a series of plots of exactly the same size and position of the rose on the viewport has to be produced.
mar	Margin around the plotting area. Default units are <code>grid.control(cyclVar.cex)</code> . Other units can be defined by <a href="#">unit</a> .
rev.col	= TRUE: Ordering of the columns of $x@rho$ is reversed.
shift	Shifts the plot in clockwise direction by this angle. Units of shift must be identical with units of $cyclVar$ in <code>link{plot.rose}</code>
cex	Basic size of characters in the plot.
col	Colors of lines when stacked = FALSE or colors of stacked areas else. In the first case colors are by default as distinct as possible; in the latter case, all colors are matched by default to the range green (=center of rose) to blue (outside of rose).
lty, lwd	Line type and line width. When stacked = TRUE and lwd = 0 the colored areas are drawn without black borders.
type	1-character string giving the type of plot desired: "s" plots observations as segments. This option is the proper way to display rose data. "l" plots a line joining the data points. This option is a nicer to look at when data are smoothly distributed in all directions.

### Value

Returns the arguments conveniently packaged up in a list to supply the general arguments for [plot.rose](#).

### Author(s)

Rene Locher

**See Also**

[plot.rose](#), [grid.control](#)

**Examples**

```
general.control()
```

---

getXY

*Easy and Flexible Input for One- and Two Dimensional Data*

---

**Description**

The function accepts one- or two dimensional data, checks them for compatibility and gives a dataframe back.

**Usage**

```
getXY(x, y = NULL, unidim.allowed = TRUE)
```

**Arguments**

- |                |   |
|----------------|---|
| x              | Vector, matrix, dataframe or list. If x is a vector, the second dimension may be defined by argument y. If x is a matrix, dataframe or list and a second column or element exists, this second element is used instead of y.  |
| y              | Optional vector of the same length as x.<br>When argument x is onedimensional, argument y does not exist <i>and</i> unidim.allowed is TRUE, argument x is coerced to a vector and returned as y component where the resulting x is just the index vector 1:n.   |
| unidim.allowed | Logical.<br>When unidim.allowed is TRUE (=default), one and two dimensional input is accepted for any reasonable combination of x and y. In this mode getXY behaves very similar to <a href="#">xy.coords</a><br>An error message is returned, when unidim.allowed is FALSE, argument x is onedimensional and y does not exist. |

**Details**

Input is checked for compatibility: When x is a list, the first two elements must have identical length. When y is defined, x must be one dimensional and must have the same length as argument y. When onedim.allowed is FALSE, input must be twodimensional.

When input is one dimensional and unidim is TRUE, y gives the numbers of elements back.

Function works similar to [xy.coords](#)

**Value**

dataframe with the components x and y

**Author(s)**

Rene Locher

**Examples**

```

getXY(3:4, 1:2)
getXY(matrix(1:4,ncol=2))
getXY(as.data.frame(matrix(1:4,ncol=2)))
getXY(4:1)
getXY(list(a=1:2,b=9:10))

```

---

grid.control

*Auxiliary for Controlling the Grid Appearance of a Rose Plot*


---

**Description**

Defines the appearance of the guiding elements of rose plots such as circles, rays and labels.

**Usage**

```

grid.control(circ.n = 4, circ.r = NULL,
             circ.col = "gray30", circ.lwd = 0.5,
             circ.cex = 0.8, circ.between = 0.3,
             circ.dir = pi/16 * 9,
             circ.sub.n = NULL, circ.sub.r = NULL,
             circ.sub.col = "gray70", circ.sub.lwd = 0.5,
             cyclVar.lab = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"),
             cyclVar.cex = 1.2, cyclVar.between = 0,
             cyclVar.centered = TRUE,
             ray.lim = NULL, ray.n = 8)

```

**Arguments**

`circ.n`, `circ.r` Number of (main)circles or, alternatively, radius of circles to be drawn, which will be labeled.

`circ.col`, `circ.lwd` Color and line width of circles.

`circ.cex` Character size of labels of main circles in multiples of `cex` as defined in [general.control](#).

`circ.between` Distance between labels of the main circle and the circle itself in multiples of `circ.cex`.

`circ.dir` Direction along which the labels of the main circles should be drawn, measured clockwise as radian from North.

`circ.sub.n`, `circ.sub.r` Number of subcircle intervals between two main circles, or, alternatively, the radii of *all* subcircles.

<code>circ.sub.col</code> , <code>circ.sub.lwd</code>	Color and line width of subcircles.
<code>cyclVar.lab</code>	Labels of cyclic variable placed along the outmost circle.
<code>cyclVar.cex</code>	Character size of labels of cyclic variable in multiples of <code>cex</code> as defined in <a href="#">general.control</a>
<code>cyclVar.between</code>	Distance between labels of the cyclic variable and the outmost circle of the rose in multiples of <code>cyclVar.cex</code> .
<code>cyclVar.centered</code>	Labels are positioned relative to their center. <code>cyclVar.centered = FALSE</code> is used for very long labels like (unabbreviated) days of the week. In this case, the labels are positioned relative to the side of the label, which is closest to the circle.
<code>ray.lim</code>	Defines the values for the center and the maximum radius in user coordinates. Be careful with specifying the center different from 0 as this might result in misleading roses.
<code>ray.n</code>	Number of rays.

**Value**

Returns the arguments conveniently packaged up in a list to supply the arguments for the grid appearance of rose plot.

**Author(s)**

Rene Locher

**See Also**

[plot.rose](#), [general.control](#)

**Examples**

```
grid.control()
```

**Description**

Produces color ramps which change simultaneously hues, saturation and values as defined in the [hsv](#) modus. This allows to produce especially smooth transitions from one color to the next. The default color ramp starts with light blue, continues with green, yellow, red and ends with dark violet.

**Usage**

```
IDPcolorRamp(n,  
             colInt = data.frame(h = c(0.47, 0.28, 0.16, 0, 1, 0.8),  
                                 s = c(0.31, 0.55, 0.7, 0.8, 0.8, 1),  
                                 v = c(1, 1, 1, 1, 1, 0.4)),  
             fr = c(0.27, 0.27, 0.27, 0))
```

**Arguments**

n	Total number of different colors in color ramp.
colInt	Data.frame or matrix with the columns h,s & v which defines the Intervals for individual color subramps, with nrow(colInt) = nsr+1 with nsr>1. See details
fr	Fraction of the colors in each of the first nsr-1 subramps.

**Details**

The function distributes the number of colors in the subramps, given the fractions fr, as smoothly as possible. The default arguments are optimized to most distinct colors possible, also for very small n. There is at least one color in the first and the last subramp.

Definition of hsv code:

h Hue of hsv-Signal: 0=red, 1/3=green, 2/3=blue, 1=red.

s Saturation of hsv-Signal: 0=white, 1=full color.

v Value of hsv-Signal: 0=black, 1=full color.

**Value**

A vector of n colors.

**Note**

If there are subramps which are not adjacent in the color space (as is here the case for red and violet), you need a virtual subramp (here from h=0.00 to h=1.00) with corresponding fraction fr == 0.

**Author(s)**

Rene Locher

**See Also**

[showColors](#), [ColorBrewer](#)

**Examples**

```
IDPcolorRamp(10)

## Default IDPcolorRamp in 21 colors
n <- 21
showColors(IDPcolorRamp(n),border=FALSE)

## colorRamp optimized to return at equidistant indices the colors
## light blue, light green, yellow, orange, red, dark violet
## works fine with n > 7
cInt <- data.frame(h = c(0.47, 0.28, 0.16, 0, 1, 0.8),
                  s = c(0.31, 0.55, 0.7, 0.8, 0.8, 1),
                  v = c(1, 1, 1, 1, 1, 0.5))

fr <- c(0.15, 0.25, 0.45, 0.0)
ii <- seq(1,n,length.out=6)

## colors at equidistant indices
showColors(IDPcolorRamp(n, colInt = cInt, fr =fr)[ii], border=FALSE)

## Alternative ramp in 21 colors
showColors(IDPcolorRamp(n, colInt = cInt, fr =fr), border=FALSE)
```

---

ilagplot

*Image Lag Plot Matrix for Large Time Series*


---

**Description**

Produces an image lag plot matrix of large timeseries where the colors encode the density of the points in the lag plots.

**Usage**

```
ilagplot(x, set.lags = 1,
         pixs = 1, zmax = NULL, ztransf = function(x){x},
         colramp = IDPcolorRamp, mfrow=NULL, cex=par("cex"),
         main = NULL, d.main = 1, cex.main = 1.5*par("cex.main"),
         legend = TRUE, d.legend = 1,
         cex.axis = par("cex.axis"), las = 1,
         border=FALSE, mar = c(2,2,2,0), oma = rep(0,4)+0.1,
         mgp = c(2,0.5,0)*cex.axis, tcl = -0.3, ...)
```

**Arguments**

x	ts object or ordinary vector
set.lags	vector of lags to be displayed
pixs	Pixel size in mm

<code>zmax</code>	Maximum counts per Pixel to be plotted. When NULL each lag plot has its individual scale. If a number $\geq$ maximum number of counts per pixel is supplied, the scale will be identical for all lag plots. The maximum of the number per pixel is delivered by the return value. Beware: <code>zmax</code> has its meaning only for <code>ilagplots</code> with identical settings for <code>main</code> , <code>legend</code> , <code>mar</code> and <code>oma</code> !
<code>ztransf</code>	Function to transform the counts. The user has to make sure that the transformed counts lie in the range $[0, zmax]$ , where <code>zmax</code> is any positive number ( $\geq 2$ ).
<code>colramp</code>	Color ramp to encode the density of the points within a pixel
<code>mfrow</code>	See Argument <code>mfrow</code> in <a href="#">par</a>
<code>cex</code>	See Argument <code>cex</code> in <a href="#">par</a>
<code>main</code>	Title
<code>d.main</code>	Vertical distance between upper border of scatter plots and the title line in multiples of title height.
<code>cex.main</code>	Magnification used for title relative to the current setting of <code>cex</code> .
<code>legend</code>	Logical. When FALSE, no legend is plotted and space is saved in figure region.
<code>d.legend</code>	Horizontal distance between right border of scatter plots and legend in multiples of title height.
<code>cex.axis</code>	Magnification used for axis annotation relative to the current setting of <code>cex</code> .
<code>las</code>	Orientation of labels on axes.
<code>border</code>	Logical. When TRUE, a border is drawn around the individual colors in the legend.
<code>mar</code> , <code>oma</code>	Margin and outer margin respectively. Cf. <a href="#">par</a>
<code>mgp</code> , <code>tcl</code>	Cf. <a href="#">par</a>
<code>...</code>	Additional arguments to <a href="#">par</a>

### Details

Code is based on R function [lag.plot](#) V1.7. Tip: Legend looks better when `mar` is defined symmetrically.

### Value

Maximum number of counts per Pixel found.

### Note

When you get the error message "Zmax too small! Densiest aereas are out of range!" you must run the function with identical parameters but without specifying `zmax`. The value returned gives you the minimum value allowed for `zmax`.

### Author(s)

Andreas Ruckstuhl, refined by Rene Locher

**See Also**

[ipairs](#), [iplot](#), [Image](#)

**Examples**

```
if(require(SwissAir)) {
  data(AirQual)

  ## low correlation
  ilagplot(AirQual[,c("ad.03")],set.lags = 1:9,
           ztransf=function(x){x[x<1] <- 1; log2(x)})

  ## high correlation
  Ox <- AirQual[,c("ad.03","lu.03","sz.03")+
            AirQual[,c("ad.NOx","lu.NOx","sz.NOx")]-
            AirQual[,c("ad.NO","lu.NO","sz.NO")]
  names(Ox) <- c("ad","lu","sz")
  ilagplot(Ox$ad,set.lags = 1:9,
           ztransf=function(x){x[x<1] <- 1; log2(x)})

  ## cf. ?AirQual for the explanation of the physical
  ## and chemical background
} else print("Package SwissAir is not available")
```

---

Image

*Display the Density of Points in a Scatter Plot by Colors*

---

**Description**

The density of points in a scatter plot is encoded by color.

**Usage**

```
Image(x, y = NULL, pixs = 1, zmax = NULL, ztransf = function(x){x},
      colramp = IDPcolorRamp, factors = c(FALSE, FALSE),
      matrix = FALSE)
```

**Arguments**

<b>x, y</b>	Coordinates of points whose density is plotted. If x is a matrix or a data.frame, the first two column are used as x and y respectively. y must be in this case NULL. x and y may be numeric or factor variable.
<b>pixs</b>	Size of pixel in x- and y-direction in [mm] on the plotting device. When x and y are numeric, pixels are square. When x and y are factors or should be handled as factors (see argument factors), pixels are no longer square. The pixels are enlarged in the dimension in which the factors are displayed, so that the rectangular pixels are centered at the factor levels.

<code>zmax</code>	Maximum number of counts per pixel in the plot. When NULL, the density in the scatter plot is encoded from 0 to maximum number of counts per pixel observed. <code>zmax</code> must be equal or larger than maximum number of counts found. The maximum number of counts per pixel is delivered by the return value.
<code>ztransf</code>	Function to transform the number of counts per pixel, which will be mapped by the function in <code>colramp</code> to well defined colors. The user has to make sure that the transformed density lies in the range <code>[0,zmax]</code> , where <code>zmax</code> is any positive number ( $\geq 2$ ). For examples see <a href="#">ipairs</a> and <a href="#">ilagplot</a> .
<code>colramp</code>	Color ramp to encode the number of the counts within a pixel by color.
<code>factors</code>	Vector of logicals indicating whether x and / or y should be handled as factors independently of their class.
<code>matrix</code>	Boolean. Should all counts be returned in a xyz-matrix or just the maximum.

### Details

Before calling `Image` a plot must have been created by, e.g., calling `plot(x,y,type="n")`. This function ensures by default that the pixel has the same size in x- and y-direction. As a drawback, pixels may be unequally spaced, when there are only very few distinct (integer) values in x- or y-direction. When this is the case, the corresponding dimension should be declared as a factor. (cf. argument `factors`).

This function is based on [graphics](#)

### Value

Maximum number of counts per pixel found (`matrix = FALSE`) or the full matrix.

### Author(s)

Andreas Ruckstuhl, Rene Locher

### See Also

[ipairs](#), [ilagplot](#), [iplot](#), [image](#)

### Examples

```
plot.default(iris$Species, iris$Petal.Width, xlim=c(0.5,3.5),
             type="n", axes=FALSE)
axis(1, at=1:3, labels=levels(iris$Species))
axis(2)
Image(iris$Species, iris$Petal.Width, pixs=3)

##
x <- rnorm(10000)
y <- rnorm(10000, 10)
plot(x+y, y, type="n")
Image(x+y, y)
abline(a=0, b=1)
```

```
## The above can be merged to
iplot(x+y, y, legend=FALSE, oma=c(5, 4, 4, 2) + 0.1)
abline(a=0, b=1)
```

---

ipairs

*Image Scatter Plot Matrix for Large Datasets*


---

## Description

Produces an image scatter plot matrix of large datasets where the colors encode the density of the points in the scatter plots.

## Usage

```
ipairs(x,
      pixs = 1, zmax = NULL, ztransf=function(x){x},
      colramp = IDPcolorRamp, cex = par("cex"),
      lab.diag, cex.diag = NULL,
      main = NULL, d.main = 1.5, cex.main = 1.5*par("cex.main"),
      legend = TRUE, d.legend = 1.5, cex.axis = 0.8*par("cex.axis"),
      nlab.axis = 5, minL.axis = 2, las = 1, border = FALSE,
      mar = rep(0,4), oma = c(3,3,1,0), mgp = c(2,0.5,0)*cex.axis,
      tcl = -0.3, ...)
```

## Arguments

x	data.frame or matrix
pixs	Pixel size in mm on the plotting device.
zmax	Maximum number of counts per pixel in the plot. When NULL, each scatter plot has its individual scale. If a number $\geq$ maximum number of counts per pixel is supplied, the scale will be identical for all scatter plots. The maximum number of counts per pixel is delivered by the return value.
ztransf	Function to transform the counts per pixel, which will be mapped by the function in <code>colramp</code> to well defined colors. The user has to make sure that the transformed counts lie in the range $[0, zmax]$ , where <code>zmax</code> is any positive number ( $\geq 2$ ).
colramp	Color ramp to encode the number of counts within a pixel.
cex	See Argument <code>cex</code> in <a href="#">par</a>
lab.diag	Labels of columns, written into the diagonal of the matrix. When NULL, the names of <code>x</code> are used.
cex.diag	Magnification used for text in diagonal relative to the current setting of <code>cex</code> . When NULL, they are calculated automatically.
main	Titel. When NULL

d.main	Vertical distance between upper border of scatter plots and the title line in multiples of title height.
cex.main	Magnification used for title relative to the current setting of cex.
legend	Logical. When FALSE, no legend is plotted and space is saved in figure region.
d.legend	Horizontal distance between right border of scatter plots and legend in multiples of title height.
cex.axis	Magnification used for axis annotation relative to the current setting of cex.
nlab.axis	Approximate number of labels on axes.
minL.axis	The minimum length of the abbreviations of factor levels, used to label the axes ticks.
las	Orientation of labels on axes.
border	Logical. When TRUE, a border is drawn around the individual colors in the legend.
mar, oma	Margin and outer margin respectively. Cf. <a href="#">par</a>
mgp, tcl	Cf. <a href="#">par</a>
...	Additional arguments to <a href="#">par</a>

### Details

The idea is similar to [gplot.hexbin](#). The hexagons are better suited to reflect the density of points in a plane than the squares used here. Nevertheless squares are, contrary to hexagons, invariant to reflexions at the *x- and y-axis* and therefore better suited for scatter plot matrices and also for plotting factors.

The code is based on R function [pairs](#) V1.7.

### Value

Maximum number of counts per Pixel found. Additional elements are returned when `verbose == TRUE`.

### Note

When you get the error message "Zmax too small! Densiest aereas are out of range!" you must run the function with identical parameters but without specifying `zmax`. The value returned gives you the minimum value allowed for `zmax`.

### Author(s)

Andreas Ruckstuhl, Rene Locher

### See Also

[ilagplot](#), [iplot](#), [Image](#)

**Examples**

```

## Small numbers of different values are plotted nicer
## when converted to factors
AQ <- airquality
AQ$Month <- as.factor(AQ$Month)

zmax <- ipairs(AQ, pixs=2, main="Air Quality")
ipairs(AQ, pixs=2, zmax=zmax, main="Air Quality",border=TRUE)

## example with factors
ipairs(iris,pixs=2)

## a really huge dataset
if(require(SwissAir)) {
  data(AirQual)

  ## low correlation
  ipairs(AirQual[,c("ad.03", "lu.03", "sz.03")],
        ztransf=function(x){x[x<1] <- 1; log2(x)})

  ipairs(AirQual[,c("ad.NO", "lu.NO", "sz.NO")],
        ztransf=function(x){x[x<1] <- 1; log2(x)})

  ## high correlation
  Ox <- AirQual[,c("ad.03", "lu.03", "sz.03")] +
    AirQual[,c("ad.NOx", "lu.NOx", "sz.NOx")] -
    AirQual[,c("ad.NO", "lu.NO", "sz.NO")]
  names(Ox) <- c("ad", "lu", "sz")
  ipairs(Ox, ztransf=function(x){x[x<1] <- 1; log2(x)})

  ## cf. ?AirQual for the explanation of the physical and
  ## chemical background
} else print("Package SwissAir is not available")

```

---

ipanel.smooth

*Panelplot for itermplot*


---

**Description**

An example of a useful panel function for huge datasets

**Usage**

```

ipanel.smooth(x, y = NULL, pixs = 1, zmax = NULL,
             ztransf = function(x) {x},
             colramp = IDPcolorRamp, col = "black", lwd = 2,
             span = 2/3, iter = 3, ...)

```

**Arguments**

<code>x,y</code>	Numeric vectors of the same length.
<code>pixs</code>	Size of pixel in x- and y-direction in [mm] on the plotting device. When x and y are numeric, pixels are square. When x and y are factors, pixels are no longer square. The pixels are enlarged in x-direction.
<code>zmax</code>	Maximum number of counts per Pixel in the plot. When NULL, the density in the scatter plot is encoded from 0 pixel to maximum number of counts observed. <code>zmax</code> must be equal or larger than maximum number of counts found.
<code>ztransf</code>	Function to transform the number of counts per pixel, which will be mapped by the function in <code>colramp</code> to well defined colors. The user has to make sure that the transformed density lies in the range [0, <code>zmax</code> ], where <code>zmax</code> is any positive number ( $\geq 2$ ). For examples see <a href="#">ipairs</a> and <a href="#">ilagplot</a> .
<code>colramp</code>	Color ramp to encode the number of counts within a pixel by color.
<code>col,lwd</code>	Color and line width of the “smoothed curve”.
<code>span</code>	the smoother span. This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness.
<code>iter</code>	The number of robustifying iterations which should be performed. Using smaller values of <code>iter</code> will make lowess run faster.
<code>...</code>	Other graphical parameters as arguments to the <a href="#">lines</a> function.

**Author(s)**

Rene Locher

**See Also**

[lowess](#)

**Examples**

```
r.lm <- lm(Sepal.Length~Sepal.Width+Petal.Length+Petal.Width+Species,
          data=iris)
par(mfrow=c(2,2),pty="s")
itermplot(r.lm, se = TRUE, partial.res=TRUE, smooth=ipanel.smooth,
          lwd.smth = 3, pixs = 2, ask=FALSE)

if (require(SwissAir)) {
  par(mfrow=c(1,1))
  dat <- AirQual[,c("ad.O3","ad.NOx","ad.T","ad.Td","ad.WS")]
  dat$ad.O3 <- log(dat$ad.O3)
  dat$ad.NOx <- log(dat$ad.NOx)
  dat$ad.WS <- log(dat$ad.WS)
  pairs(dat,
        panel=ipanel.smooth)
} else print("Package SwissAir is not available")
```

iplot

*Image Scatter Plot for Large Datasets***Description**

Produces an image scatter plot of large datasets where the colors encode the density of the points in the scatter plot. Works also with factors.

**Usage**

```
iplot(x, y = NULL,
      pixs = 1, zmax = NULL, ztransf = function(x){x},
      colramp = IDPcolorRamp, cex = par("cex"),
      main = NULL, d.main = 1, cex.main = par("cex.main"),
      xlab = NULL, ylab = NULL, cex.lab = 1,
      legend = TRUE, d.legend = 1,
      cex.axis = par("cex.axis"), nlab.xaxis = 5, nlab.yaxis = 5,
      minL.axis = 3, las = 1, border = FALSE,
      oma = c(5,4,1,0)+0.1, mgp = c(2,0.5,0)*cex.axis, tcl = -0.3, ...
    )
```

**Arguments**

<code>x,y</code>	Coordinates of points whose density is to be plotted. If <code>x</code> is a matrix or a data.frame, the first two columns are used as <code>x</code> and <code>y</code> respectively. <code>y</code> must be in this case <code>NULL</code> . <code>x</code> and <code>y</code> may be numeric or factor variable.
<code>pixs</code>	Pixelsize in mm.
<code>zmax</code>	Maximum number of counts per pixel in the plot. When <code>NULL</code> , the density in the scatter plot is encoded from 0 to maximum number of counts per pixel observed. <code>zmax</code> must be equal or larger than maximum number of counts found. The maximum number of counts per pixel is delivered by the return value.
<code>ztransf</code>	Function to transform the number of counts per pixel, which will be mapped by the function in <code>colramp</code> to well defined colors. The user has to make sure that the transformed density lies in the range <code>[0,zmax]</code> , where <code>zmax</code> is any positive number ( $\geq 2$ ). For examples see <a href="#">ipairs</a> and <a href="#">ilagplot</a> .
<code>colramp</code>	Color ramp to encode the number of counts within a pixel.
<code>cex</code>	Magnification of text relative to default.
<code>main</code>	Title.
<code>d.main</code>	Vertical distance between upper border of scatter plots and the title line in multiples of title height.
<code>cex.main</code>	Magnification used for title relative to the current setting of <code>cex</code> .
<code>xlab</code>	Label for x-axis.
<code>ylab</code>	Label for y-axis.
<code>cex.lab</code>	Magnification used for text in axis labels relative to the current setting of <code>cex</code> .

legend	Logical. When FALSE, no legend is plotted and space is saved in figure region.
d.legend	Horizontal distance between right border of scatter plot and legend in multiples of title height.
cex.axis	Magnification used for axis annotation relative to the current setting of cex.
nlab.xaxis, nlab.yaxis	Approximate number of labels on x- and y-axes respectively.
minL.axis	The minimum length of the abbreviations of factor levels, used to label the axes ticks.
las	Orientation of labels on axes.
border	Logical. When TRUE, a border is drawn around the individual colors in the legend.
oma	Outer margin. Cf. <a href="#">par</a>
mgp, tcl	Cf. <a href="#">par</a>
...	Additional arguments to <a href="#">par</a>

### Details

The idea of this plot is similar to [gplot.hexbin](#). The hexagons are better suited to reflect the density of points in a plane than the squares used here. Nevertheless squares are, contrary to hexagons, invariant to reflexions at the x- *and* y-axis and therefore suited for scatter plot matrices and also for plotting factors.

### Value

Maximum number of counts per Pixel found.

### Note

When you get the error message "Zmax too small! Densiest aereas are out of range!" you must run the function again without specifying zmax. The value returned gives you the minimum value allowed for zmax.

### Author(s)

Andreas Ruckstuhl, Rene Locher

### See Also

[ilagplot](#), [ipairs](#), [Image](#)

### Examples

```
x <- rnorm(10000)
y <- atan(rnorm(10000, 0))
iplot(x, y)
iplot(x, pixs=2)

oma <- c(5,5,0,0)
```

```

iplot(iris[,1:2],pixs=4, oma=oma)
iplot(iris[,"Petal.Width"], iris[,"Species"], pixs=4, oma=oma)
iplot(x=iris[,"Species"], y=iris[,"Petal.Width"], pixs=10,border=TRUE,
      xlab="Species",
      ylab="Petal Width",
      main="Iris Species and Petal Width", oma=oma)

iplot(iris$Species, iris$Petal.Width,pixs=3, minL.axis=10,
      oma=c(3,6,0,0), mgp=c(4, 1, 0),
      cex.axis=2, cex.lab=2, cex.main= 2, main="Larger fonts")

```

---

iplotLegend

*Plots Legend for Color Ramp*


---

## Description

Plots legend for color ramp.

## Usage

```

iplotLegend(colramp, ncol = NULL,
            cex.axis = par("cex.axis"), border = FALSE,
            mar = c(0, 0, 0, 3), las = 1, ...)

```

## Arguments

colramp	Function defining color ramp, e.g. <a href="#">IDPcolorRamp</a>
ncol	Number of individual colors in color ramp. See details.
cex.axis	Magnification to be used for labels.
border	Logical. When TRUE, a border is drawn around the individual colors in the legend.
mar	Margin. See <a href="#">par</a> .
las	Orientation of labels on axis. See <a href="#">par</a> .
...	Additional arguments to par

## Details

A color ramp of `ncol+1` individual colors is plotted, starting with the background color (= color 0). When `ncol = NULL` a color ramp of 101 individual colors is plotted. The color at the bottom is labeled by '0', the color at the top by 'max'.

## Author(s)

Rene Locher

## Examples

```
## Not run:
par(oma=rep(2,4))
layout(matrix(c(1,2),ncol=2),
        width=c(lcm(par("csi")*5*2.54),1),
        heights=1)
## End(Not run)
iplotLegend(IDPcolorRamp,ncol=15)
```

---

iplot

*Plot Regression Terms for Huge Datasets*


---

## Description

Plots regression terms against their predictors, optionally with standard errors and partial residuals in a density plot.

## Usage

```
itermplot(model, data = NULL, envir = environment(formula(model)),
          partial.resid = FALSE, scale=0, pixs = 1,
          zmax=NULL, ztransf = function(x) {x}, colramp = IDPcolorRamp,
          terms = NULL, se = FALSE,
          xlabs = NULL, ylabs = NULL, main = NULL,
          col.term = "black", lwd.term = 2,
          col.se = "gray", lty.se = 2, lwd.se = 1,
          col.smth = "darkred", lty.smth = 2,
          lwd.smth = 2, span.smth = 2/3,
          ask = interactive() && nb.fig < n.tms &&
          .Device != "postscript",
          use.factor.levels = TRUE, smooth = NULL, ...)
```

## Arguments

model	Fitted model object
data	Data frame in which variables in model can be found
envir	Environment in which variables in model can be found
partial.resid	Logical; should partial residuals be plotted?
scale	A lower limit for the number of units covered by the limits on the 'y' for each plot. The default is <code>scale = 0</code> , in which case each plot uses the range of the functions being plotted to create their ylim. By setting <code>scale</code> to be the maximum value of <code>diff(ylim)</code> or above for all the plots, then all subsequent plots will be produced in the same vertical units. This is essential for comparing the importance of fitted terms in additive models.

<code>pixs</code>	Size of pixel in x- and y-direction in [mm] on the plotting device. When x and y are numeric, pixels are square. When x and y are factors, pixels are no longer square. The pixels are enlarged in x-direction.
<code>zmax</code>	Maximum number of counts per pixel in the plot. When NULL, each scatter plot has its individual scale. If a number $\geq$ maximum number of counts per pixel is supplied, the scale will be identical for all scatter plots. The maximum number of counts per pixel is delivered by the return value.
<code>ztransf</code>	Function to transform the number of counts per pixel. The user has to make sure that the transformed density lies in the range $[0, zmax]$ , where <code>zmax</code> is any positive number ( $\geq 2$ ). For examples see <a href="#">ipairs</a> and <a href="#">ilagplot</a> .
<code>colramp</code>	Color ramp to encode the number of counts within a pixel by color.
<code>terms</code>	Numeric. Which terms to plot (default NULL means all terms)
<code>se</code>	Logical. Plot pointwise standard errors?
<code>xlabs</code>	Vector of labels for the x axes
<code>ylabs</code>	Vector of labels for the y axes
<code>main</code>	Logical, or vector of main titles; if TRUE, the model's call is taken as main title, NULL or FALSE mean no titles.
<code>col.term, lwd.term</code>	Color and line width for the "term curve"
<code>col.se, lty.se, lwd.se</code>	Color, line type and line width for the "twice-standard-error curve" when <code>se = TRUE</code> .
<code>col.smth, lty.smth, lwd.smth</code>	Color, line type and line width for the smoothed curve
<code>span.smth</code>	Smoothing parameter <code>f</code> for <a href="#">lowess</a> .
<code>ask</code>	Logical. Should user be asked before each plot? cf. <a href="#">par</a> .
<code>use.factor.levels</code>	Logical. Should x-axis ticks use factor levels or numbers for factor terms?
<code>smooth</code>	NULL or a function with the same arguments as <a href="#">ipanel.smooth</a> to draw a smooth through the partial residuals for non-factor terms
<code>...</code>	Other graphical parameters

## Details

`itermplot` is a modified version of [termplot](#) of R V2.3.1. Partial residuals are displayed here as a density plot and is therefore especially suited for models of huge datasets. The model object must have a `predict` method that accepts `type=terms`, eg `glm` in the base package, `coxph` and `survreg` in the survival package.

For the `partial.resid=TRUE` option it must have a `residuals` method that accepts `type="partial"`, which `lm`, `glm` and `gam` do.

The `data` argument should rarely be needed, but in some cases `termplot` may be unable to reconstruct the original data frame. Using `na.action=na.exclude` makes these problems less likely.

Nothing sensible happens for interaction terms.

**Value**

Maximum number of counts per pixel found.

**Author(s)**

Rene Locher

**See Also**

[termplot](#).

**Examples**

```
r.lm <- lm(Sepal.Length~Sepal.Width+Petal.Length+Petal.Width+Species,
           data=iris)
par(mfrow=c(2,2),pty="s")
itermplot(r.lm, se = TRUE, partial.res=TRUE, lwd.term = 3,
          lwd.se = 2, pixs = 2)

if (require(SwissAir)) {
  data(AirQual)
  r.lm <- lm(log(ad.O3)~log(ad.NOx)+ad.T+ad.Td+ad.WS, data=AirQual)
  par(mfrow=c(2,2),pty="s")
  itermplot(r.lm, se = TRUE, partial.resid=TRUE, smooth=ipanel.smooth,
            lwd.smth = 3, pixs = 1, ask=FALSE)
} else print("Package SwissAir is not available")
```

---

key.control

*Auxiliary for Controlling the Appearance of the Legend of a Rose Plot*

---

**Description**

Defines the appearance of the legend of a Rose Plot.

**Usage**

```
key.control(plot = TRUE, lab = NULL, title = NULL, between = 0)
```

**Arguments**

plot	Defines whether legend should be drawn. When FALSE, the full width of the viewport is used for the rose itself.
lab, title	Labels and title of legend. When lab = NULL, labels are extracted from the names of the slot x@rho of <a href="#">rose</a> object.
between	Distance between label East and left edge of legend in cex as defined in <a href="#">general.control</a>

**Value**

Returns the arguments conveniently packaged up in a list to supply the arguments for the legend in [plot.rose](#)

**Author(s)**

Rene Locher

**See Also**

[plot](#)

**Examples**

```
key.control()
```

---

longtsPlot

*Plot Very Long Regular Time Series*

---

**Description**

Plot one or more regular time series in multiple figures on one or more pages.

**Usage**

```
longtsPlot(y1, y2 = NULL,
           names1 = NULL, names2 = NULL,
           startP = start(y1)[1], upf = 400, fpp = 4, overlap = 20,
           x.at = NULL, x.ann = NULL, x.tick = NULL,
           y1.at = NULL, y1.ann = NULL, y1.tick = NULL,
           y2.at = NULL, y2.ann = NULL, y2.tick = NULL,
           nx.ann = 10, ny.ann = 3, cex.ann = par("cex.axis"),
           xlab = "", y1lab = "", y2lab = "", las = 0,
           col.y1 = "black", col.y2 = col.y1,
           cex.lab = par("cex.lab"),
           ylim = range(y1, na.rm = TRUE, finite=TRUE),
           y2lim = range(y2, na.rm = TRUE, finite=TRUE),
           lty1 = 1, lty2 = 2, lwd1 = 1, lwd2 = lwd1,
           col1 = NULL, col2 = NULL,
           leg = TRUE, y1nam.leg = NULL, y2nam.leg = NULL,
           ncol.leg = NULL, cex.leg = par("cex"),
           h1 = NULL, h2 = NULL, col.h1 = "gray70", col.h2 = "gray70",
           main = NULL, cex.main = par("cex.main"),
           automain = is.null(main),
           mgp = c(2, 0.7, 0), mar = c(2,3,1,3)+.2,
           oma = if (automain!is.null(main))
                   c(0,0,2,0) else par("oma"),
```

```
xpd = par("xpd"), cex = par("cex"),
type1 = "s", type2 = type1,
pch1 = 46, pch2 = pch1, cex.pt1 = 2, cex.pt2 = cex.pt1,
slide = FALSE, each.fig = 1,
filename = NULL, extension = NULL, filetype = NULL, ...)
```

### Arguments

<code>y1, y2</code>	Regular time series, time series matrices or ordinary vectors, dataframes or matrices with values corresponding to regular time intervals. Corresponding axes are on the left (for <code>y1</code> ) and on the right (for <code>y2</code> ) respectively. <code>y2</code> is optional and may have a different frequency and a different start time.
<code>names1, names2</code>	Names for time series used for legend.
<code>startP</code>	Start time of plot.
<code>upf</code>	Number of time units plottet per figure.
<code>fpp</code>	Number of figures per page (screen).
<code>overlap</code>	Length of time series on the right end of the figures which is identical with the left start of the next figure.
<code>x.at</code>	Time points at which long ticks on x-axis and annotations are set.
<code>x.ann</code>	Time annotations (character vector) for long ticks on x-axis
<code>x.tick</code>	Time points at which short ticks are set on x-axis. If Null, no short ticks are set.
<code>y1.at, y1.ann, y1.tick, y2.at, y2.ann, y2.tick</code>	Arguments starting with 'y1' ('y2') correspond to arguments for left (right) axis. For details see arguments for x-axis.
<code>nx.ann, ny.ann</code>	Approximate number of annotations on x- and y-axes, respectively
<code>cex.ann</code>	The magnification to be used for axis annotation relative to the current setting of <code>cex</code> .
<code>xlab</code>	Label for x-axis.
<code>y1lab, y2lab</code>	Labels for y-axes on the left and on the right end of the figure.
<code>las</code>	Cf. <a href="#">par</a> .
<code>col.y1, col.y2</code>	Colors for left and right axis respectively.
<code>cex.lab</code>	<code>cex</code> of labels.
<code>y1lim, y2lim</code>	Limits for left and right axis respectively.
<code>lty1, lty2</code>	Vector of line types for each time series, possibly recycled.
<code>lwd1, lwd2</code>	Vector of line widths for each time series, possibly recycled.
<code>col1, col2</code>	Vector of color for each time series, possibly recycled.
<code>leg</code>	Logical. When TRUE, legend is drawn.
<code>y1nam.leg, y2nam.leg</code>	Name of y1- and y2-axis in legend.
<code>ncol.leg</code>	Number of columns in legend.
<code>cex.leg</code>	<code>cex</code> for legend.

h1, h2	Vector of y-positions of horizontal lines in the coordinate systems represented by the left and the right axis respectively.
col.h1, col.h2	Vector of colors for horizontal lines.
main	Title of plot.
cex.main	cex of title.
automain	Logical. When TRUE, the units of the start and the end on the page are printed in the title.
mgp, mar, oma	Cf. <a href="#">par</a> .
xpd	Define whether data points are clipped to the plot region (=FALSE) or not (=TRUE). Horizontal lines, defined by h1 or h2 are always clipped.
cex	Cf. <a href="#">par</a> .
type1, type2	For plotting lines or points. Cf. <a href="#">plot</a> .
pch1, pch2, cex.pt1, cex.pt2	Plotting symbols and there sizes when 'type1' or 'type2' = "p"
slide	Logical. When FALSE, the first page (screen) is plotted only. When TRUE, user may leaf through the pages interactively.
each.fig	For each.fig==1 all data are plotted. for each.fig==2 every 2nd page is plotted and so on.
filename	When filename is not NULL, the displayed screen(s) are saved to disk and there is no question whether the next page should be printed. When slide = TRUE, the filename of the plots is extended by a serial number.
extension	The extension is defined automatically on windows systems and must be defined manually on the other systems.
filetype	<i>On Windows:</i> The type of plot, Windows metafile, PNG, JPEG, BMP (Windows bitmap format), TIFF, PostScript or PDF. Defaults to Windows meta file, "wmf" , with the extension ".wmf". Cf. <a href="#">savePlot</a> . <i>On non Windows operating systems:</i> A device function (e.g., x11, postscript,...). The filetype defaults to postscript , with the extension ".ps". Cf. <a href="#">dev.print</a>
...	Additional arguments to <a href="#">savePlot</a> and <a href="#">dev.print</a> respectively.

### Details

For longer time-series, it is sometimes important to spread several time-series plots over several subplots or even over several pages with several subplots in each. Moreover, these series have often different ranges, frequencies and start times. There is sometimes also the need of a more flexible annotation of axes than `plot.ts` provides. `longtsPlot` provides the user with all these features for one or two matrices or regular time series (time series matrices).

### Side Effects

One or more pages of time series plots are drawn on the current graphic device and, optionally, saved in one or more files.

### Author(s)

Rene Locher

### Examples

```

## sunspots, y-axis only on the left
data(sunspots)
longtsPlot(sunspots, upf=ceiling((end(sunspots)-start(sunspots))[1]/5))

## air quality (left axis) and meteo data (right axis)
## use xpd=TRUE for time series with rare but large values
if (require(SwissAir)) {
  data(AirQual)
  st <- 6.5*30*48
  x.at <- seq(st, nrow(AirQual), 48)
  longtsPlot(y1=AirQual[,c("ad.O3", "ad.NOx")], y2 = AirQual$ad.T,
             names1=c("O3", "NOx"), names2 = "Temp",
             startP = st, upf=7*48,
             x.at = x.at, x.ann = substr(AirQual$start, 1, 6)[x.at],
             x.tick = seq(st, nrow(AirQual), 12),
             y1.at = c(0, 100), y1.tick = seq(0, 150, 50),
             y2.at = c(10, 30), y2.tick = seq(10, 30, 10),
             y1lab="ppb", y2lab="[C]",
             y1lim = c(0, 100), y2lim = c(10, 30), xpd=TRUE,
             col2 = "red", type1 = "l")
}

## Two time series with different frequencies and start times
## on the same figures
set.seed(13)
len <- 4*6*400
x <- sin((1:len)/200*pi)
d <- sin(cumsum(1+ rpois(len, lambda= 2.5)))

y1 <- ts(10*x, start=0, frequency=6)+d*rnorm(len)
y2 <- ts(100*x, start=100, frequency=13)+10*rnorm(len)
longtsPlot(y1, y2)

## plot your own legend
longtsPlot(sunspots, upf=ceiling((end(sunspots)-start(sunspots))[1]/5),
           fpp=1, leg=FALSE)
legend(1750, 260, legend="Monthly Sunspot Numbers", col="blue", lwd=1,
      bty="n")

```

---

MS

*Spectrum Measured by a SELDI TOF Mass Spectrometer*


---

### Description

The spectrum was taken from a sample of sheep blood. The instrument used was a so called SELDI TOF (Surface Enhanced Laser Desorption Ionisation, Time Of Flight) Mass Spectrometer.

### Usage

```
data(MS)
```

**Format**

A data frame with 45213 observations on the following 2 variables.

mz mass / charge

I Intensity

**Details**

The measured masses lie between  $m/z=1000$  and 200000. The intensities are raw output. Neither smoothing nor background subtraction was applied to the spectrum.

**Source**

Medical research project.

**Examples**

```
data(MS)

MS1 <- log10(MS[MS$mz>12000&MS$mz<1e5,])
P <- peaks(MS1, minPH=0.025, minPW=0.0015)

plot(MS1, ty="l", xlim=c(4.15,5))
points(P,col="red")
```

---

NaRV.omit

*Omit Observations with NA, NaN, Inf and -Inf Values*

---

**Description**

Omits observations with values which are not regular (=Not a Regular Value) when object is a vector, a factor, a data.frame or a matrix.

**Usage**

```
NaRV.omit(x)
```

**Arguments**

x                      Vector, data.frame or matrix

**Details**

Irregular values are defined as NA, NaN, Inf and -Inf Values in numerics and NA in factors and characters.

**Value**

Returns objects of class vector, factor, data.frame or matrix in the same way as `na.omit` does. Returns all other objects unchanged and prints a warning.

**Author(s)**

Rene Locher

**See Also**

[na.omit](#)

**Examples**

```
M <- matrix(c(NA,1:7,NA),nrow=3)
M
NaRV.omit(M)

DF <- iris[sample(1:nrow(iris),12),]
DF[1,1] <- NA
DF[10,5] <- NA
row.names(DF) <- 1:12
DF
NaRV.omit(DF)

NaRV.omit(c(NA,1:10,NA))

fac <- factor(c(NA,sample(c(1:9))))
NaRV.omit(fac)

fac <- factor(c(NA,sample(c(1:9))),exclude=NULL)
fac
NaRV.omit(fac)
```

---

ok

*Sets NAs in Logical Objects to FALSE*

---

**Description**

Sets NAs in logical vectors and matrices to FALSE. This is especially useful for conditional selections of data when the variables the condition is based on contain NAs.

**Usage**

```
ok(x)
```

**Arguments**

x                    Logical vector or matrix

**Value**

Logical vector or matrix, containing no NAs.

**Author(s)**

Rene Locher

**Examples**

```
ok(c(FALSE, TRUE, NA, TRUE))
```

---

peaks

*Finding Peaks in Raw Data*

---

**Description**

Returns position, signal height and approximate width at half maximum peak height.

**Usage**

```
peaks(x, y = NULL, minPH, minPW, thr, stepF = 0.49)
```

**Arguments**

x, y	Position and height of signal. Any reasonable way of defining the coordinates is acceptable. See function <code>link{getXY}</code> for details.
minPH	Minimum height of peak to be reported.
minPW	Minimum width of peak at half maximum to be reported.
thr	Threshold below which the signal is not processed.
stepF	StepF defines indirectly the accuracy of the selection criteria <code>minPH</code> and <code>minPW</code> and of the value of the calculated width: The smaller the more accurate and the slower the function. It must be $<0.5$

**Details**

The function is especially useful for signals in which both very broad and very narrow peaks are of interest. The peaks may lie very close to each other or might even be superpositioned on top of each other, e.g. peaks on broader shoulders. The algorithm is also very useful when the resolution of the signal is poor and the noise is small.

The function is looking for peaks without any preceding baseline subtraction or smoothing, which could distort the spectrum.

The selection criteria `minPH` and `minPW` and the values for the calculated peak widths are only approximate.

**Value**

dataframe consisting of

x	Position of peak
y	Signal height
w	Approximate width at half maximum of peak

**Note**

In the function, the main selection criterium for the peaks is the height of the peaks, the second optional criterium is the width of the peaks.

When the width of different peaks vary not too strongly and the distances between the peaks is not critical, you might use the faster function `peaks` of the bioconductor package `PROcess`.

**Author(s)**

Rene Locher

**Examples**

```
n <- 200
freq <- 1:n
theory <- sin(freq/n*4*pi)*cos(freq/n*3*pi)
spec <- theory + 0.1*rnorm(n)

plot(spec,type="b")
lines(theory,lwd=2)

pts <- peaks(spec, minPH=0.7)
points(pts,col="red",cex=1.2, pch=20)

## peaks after smoothing the spectrum
spec.sm <- loess.smooth(freq, spec, span=0.2,
                        degree = 2, evaluation = 100)
lines(spec.sm$x, spec.sm$y, col="steelblue", lwd=2)
pts <- peaks(spec.sm, minPH=0.4)
points(pts,col="green",cex=1.2,pch=20)

## Analyses of Mass Spectrum between 12000 and 100'000
## without smoothing, without baseline subtraction
data(MS)
MS1 <- log10(MS[MS$mz>12000&MS$mz<1e5,])

P <- peaks(MS1, minPH=0.02, minPW=0.001)
plot(MS1, type="l", xlab="log10(mz)", ylab="log10(I)")
points(P,col="blue",cex=1.6)

## compare to
if (require(PROcess)) {
  ii <- peaks(MS1$I,span=300)
  detach("package:PROcess")
}
```

```

    points(MS1$mz[ii],MS1$I[ii],col="darkgreen",cex=1.2)
  } else print("Package Process not available!")

```

---

plot.rose

*Plot Method for Class "rose" (Grid Graphics Function)*


---

## Description

Describes plot method for class "rose"

## Usage

```

## S4 method for signature 'rose,missing'
plot(
  x,
  transf = function(x) sqrt(x),
  subset.col = NULL,
  warn = TRUE,
  general = general.control(),
  grid = grid.control(),
  title = title.control(),
  key = key.control())

```

## Arguments

x	Object <a href="#">rose</a> .
transf	Transformation function for $x@rho$ . It defaults to the square root, resulting in equal area roses.
subset.col	Display only a subset of the columns of $x@rho$ . Subset might be declared by numerical values or by name.
warn	Logical. If FALSE, warnings are suppressed.
general	Settings for general appearance of plot, defined in <a href="#">general.control</a> .
grid	Settings for appearance of guiding elements of rose such as circles, rays and labels, defined in <a href="#">grid.control</a>
title	Settings for title, defined in <a href="#">title.control</a> .
key	Settings for appearance of the legend, defined in <a href="#">key.control</a> .

## Details

This function appeals especially to environmental specialists who often have response variables, which depend from cyclic variables like the direction of wind, the hour of the day, the month etc. All these variables are displayed usually clockwise, starting with 0 in the north (12 o' clock). We call this kind of coordinates 'clock coordinates', to distinct them from the polar coordinates as used in mathematical context. The [rose](#) object is displayed as the time on a clock, measuring the angle

defined in slot `cyclVar` in the clockwise direction from the north.

The eye takes the area of a graphical object as a measure of its size. This is why the default transformation of `x@rho` is chosen to be the square root. For equal distance roses use the transformation function `function(x) x`.

All labels, titles and line sizes are defined in multiples of `cex`.

This graphic function is based on package [grid](#): Viewport `vp.rose` which was used to draw the rose and viewport `vp.key` which was used to draw the key may be addressed by `pushviewport()` after having drawn the figure.

### Value

No value returned

### Note

The function is designed to use the area on the active viewport in an optimal way, but the plot is not scalable after having been drawn.

Furthermore this function is still experimental so that some features may change in future versions.

You will find another nice example for this plot method in [AirQual](#)

### Author(s)

Rene Locher

### See Also

[rose](#), [rose-class](#)

### Examples

```
hour <- rep(0:23,100)
WD <- c(rnorm(24*90, mean=sample(c(190,220,50),24*90,
                               replace = TRUE),sd=10),
        rnorm(24*10, mean=360, sd=180))%360
dat <- data.frame(A = (2*cos((hour+6)/6*pi)+
                    2*cos((WD+60)/180*pi)+rnorm(24*100,4))^2,
                 B = (2*cos((hour+4)/6*pi)+rnorm(24*100,1,8))^2)
dat$B[dat$B>1000] <- 1000

## two different response variables, scalar summary function
mean.dayrose <- rose(dat[,c("A","B")],
                    cyclVar = hour,
                    n.cyclVar = 24,
                    circle = 24,
                    FUN=mean, na.rm=TRUE)

## one response variable, vector summary function
quant.windrose <- rose(dat$A,
                      cyclVar = WD,
                      n.cyclVar = 16, circle = 360,
                      FUN=quantile, na.rm=TRUE)
```

```

## one response variable, second (non cyclic) explanatory variable,
## scalar summary function
windrose <- rose(dat[,c("A")],
                cyclVar = WD,
                n.cyclVar=8,
                circle = 360,
                cut = dat$B,
                breaks = c(0,10,100,1000),
                include.lowest = TRUE, dig.lab = 4,
                FUN = function(x) sum(!is.na(x)))

grid.newpage()
plot(mean.dayrose,
     general = general.control(
       mar = rep(1,4),
       stacked = FALSE,
       lwd = 3,
       lty = c(1:2)),
     grid = grid.control(
       circ.n = 2,
       circ.sub.n = 2,
       circ.lwd = 2,
       circ.sub.col = "black",
       ray.n = 12,
       cyclVar.lab = seq(0,by=2,to=22)),
     title = title.control(text = "unstacked dayrose"),
     key = key.control(title = "Mean",
                       between = 0))

grid.newpage()
plot(quant.windrose)

grid.newpage()
plot(windrose,
     general = general.control(
       stacked = TRUE,
       lwd = 3),
     grid = grid.control(
       circ.n = 2,
       circ.sub.n = 2),
     title = title.control(
       text = "Stacked windrose:\nCounts of A-Values"),
     key = key.control(title = "Value of B"))

if (require(SwissAir)){
  data(AirQual)
  dat <-
    data.frame(month =as.numeric(substr(AirQual$start,4,5)),
              hour = as.numeric(substr(AirQual$start,12,13)),
              WD = AirQual$ad.WD,
              NOx = AirQual$ad.NOx,

```

```

## NO2 = AirQual$ad.NOx-AirQual$ad.NO,
## NO  = AirQual$ad.NO,
O3    = AirQual$ad.O3,
Ox    = AirQual$ad.O3+AirQual$ad.NOx-AirQual$ad.NO)

## Windrose
windrose <- rose(dat$WD,
                 cyclVar = dat$WD, n.cyclVar = 32, circle = 360,
                 FUN = function(x) sum(!is.na(x)))

grid.newpage()
plot(windrose,
     general =
     general.control(lwd = 2),
     grid =
     grid.control(circ.n = 2,
                  circ.sub.n = 2))

## median of concentrations as a function of daytime
## from May to September
med.dayrose <- rose(dat[,c("NOx", "O3", "Ox")],
                   subset= dat$month>4 & dat$month<10,
                   cyclVar=dat$hour, n.cyclVar=24, circle=24,
                   FUN=median, na.rm=TRUE)

## line type version of rose
grid.newpage()
plot(med.dayrose,
     general = general.control(lwd=2, type="l"),
     grid =
     grid.control(ray.n = 12,
                  circ.n =2,
                  circ.sub.n = 2,
                  cyclVar.lab = seq(0,by=2,to=22)),
     title = title.control(text =
                           "Day Rose of Medians\nduring summer"))

## quantiles of concentrations as a function of daytime
## from May to September
quant.dayrose <- rose(dat$NOx,
                     subset= dat$month>4 & dat$month<10,
                     cyclVar=dat$hour, n.cyclVar=24, circle=24,
                     FUN=quantile, na.rm=TRUE)

grid.newpage()
plot(quant.dayrose,
     general =
     general.control(mar = c(0.3, 0.3, 0.3, 2),
                    lwd = 2),
     grid =
     grid.control(ray.n = 12,
                  cyclVar.lab = seq(0,by=2,to=22)),
     title = title.control(text = "Concentration of NOx [ppb]\nduring summer"),

```

```

    key = key.control(title = "Quantiles"))
} else print("Package SwissAir is not available")

```

---

poster.plot

*Convenient xyplot with Differently Colored Margin and Plot Region*


---

### Description

Convenient xyplot with Colored Background. Background of margin may be chosen independently from background in plot region.

### Usage

```

poster.plot(x, y = NULL, type = "p",
            col = col.fg, col.axis = col.fg, col.lab = col.fg,
            col.fg = "blue", col.bg = "lavender", col.box = "cornsilk",
            xlim = NULL, ylim = NULL, xlab = "", ylab = "",
            main = "", cex = 1.2, axes = TRUE, ...)

```

### Arguments

x	A vector, data.frame or matrix. When x is data.frame or matrix only first two columns are used.
y	A vector or NULL
type	See Argument type in <a href="#">par</a>
col	Color of points. If length(col) > 1, colors are recycled.
col.axis	Color of axis.
col.lab	Color of labels on axis.
col.fg	Color of foreground.
col.bg	Color of background outside of figure.
col.box	Color of background inside of figure
xlim	Limits of x-axis.
ylim	Limits of y-axis
xlab	Label of x-axis
ylab	Label of y-axis
main	Titel.
cex	Size of characters.
axes	Should axis be plotted?
...	Additional arguments to <a href="#">par</a> .

**Details**

Side effect: par options will remain changed so that other graphic elements can be added comfortably.

**Author(s)**

Andreas Ruckstuhl, refined by Rene Locher

**Examples**

```
poster.plot(iris[,1],iris[,2],
            xlab="Sepal.Length", ylab="Sepal.Width")

poster.plot(iris[,1], col="red", col.box="grey95", ylab="Sepal.Length")

## plotting lines
n <- 200
freq <- 1:n
y <- sin(freq/n*4*pi)*cos(freq/n*3*pi) + 0.1*rnorm(n)
poster.plot(y,col.fg="grey30",type="l")
```

---

 rfbaseline

*Robust Fitting of Baselines*


---

**Description**

Robust fitting of local regression models for estimating a baseline or a background signal

**Usage**

```
rfbaseline(x, y, span = 2/3, NoXP = NULL,
           maxit = c(2, 2), b = 3.5, weight = NULL,
           Scale = function(r) median(abs(r))/0.6745,
           delta = NULL, SORT = TRUE, DOT = FALSE, init = NULL)
```

**Arguments**

x, y	Abscissa and ordinate of the points on the scatterplot.
span	Specifies the amount of smoothing; span is the fraction of points used to compute each fitted value; as span increases the output becomes smoother.
NoXP	Another way of specifying the amount of smoothing; NoXP is the Number of X Points used to compute each fitted value; it must be larger than 3.
maxit	The number of iterations in the robust fit; if maxit=c(0,0), the nonrobust fit is returned; the first entry specifies the number of iterations using an asymmetric biweight function, whereas the second entry specifies the number of iterations using the usual (symmetric) biweight function.
b	Tuning constant in the biweight function.

weight	Optional weights to be given to individual observations.
Scale	function specifying how to calculate the scale of the residuals.
delta	Nonnegative parameter which may be used to save computation. By default, if $\text{length}(x) \leq 100$ , delta is set equal to 0; if $\text{length}(x) > 100$ set to 1/100th of the range of $x$ .
SORT	Boolean variable indicating whether $x$ data must be sorted. Change it only when the $x$ are sorted and you want to save computer time.
DOT	If TRUE disregard outliers totally; that is, observations with weight 0 are disregarded even when the neighbourhood is determined.
init	Values of an initial fit.

**Value**

List containing components

$x$	Sorted input vector $x$ with duplicate points removed
$y$	Corresponding input vector $y$
fit	Fitted values at $x$
rw	Robust weights of $(x,y)$ -Points used in last iteration of fit
scale	Scale used in last iteration of fit

**Author(s)**

Andreas Ruckstuhl

**References**

Ruckstuhl, Andreas F., Matthew P. Jacobson, Robert W. Field and James A. Dodd (2001); Baseline Subtraction Using Robust Local Regression Estimation; Journal of Quantitative Spectroscopy and Radiative Transfer **68**: 179 – 193

Ruckstuhl, Andreas F., et al.; Estimation of background concentrations of atmospheric trace gases using robust local regression; to be published

**See Also**

See Also as [loess](#) and [lowess](#)

**Examples**

```
data(MS)
MS1 <- log10(MS[MS$mz>12000&MS$mz<1e5,])

MS1.rfb2 <- rfbaseline(x=MS1$mz, y=MS1$I, NoXP=2200, maxit=c(5,0))
plot(x=MS1$mz, y=MS1$I, type="l",
      xlab="log(mass/charge)", ylab="log(intensity)")
lines(MS1.rfb2$x, MS1.rfb2$fit, col="orange", lwd=3)

MS1.rfb3 <- rfbaseline(x=MS1$mz, y=MS1$I, NoXP=1100, maxit=c(5,0),
```

```
                                DOT=TRUE, Scale=function(x) mad(x, center=0))
plot(x=MS1$mz, y=MS1$I, type="l",
     xlab="log(mass/charge)", ylab="log(intensity)")
lines(MS1.rfb3$x, MS1.rfb3$fit, col="orange", lwd=3)

## 'delta=0' needs much more computer time
## Not run:
MS1.rfb4 <- rfbaseline(x=MS1$mz, y=MS1$I, NoXP=2200,
                      delta=0, maxit=c(5,0))
plot(x=MS1$mz, y=MS1$I, ty="l",
     xlab="log(mass/charge)", ylab="log(intensity)")
lines(MS1.rfb4$x, MS1.rfb4$fit, col="orange", lwd=3)

## End(Not run)
```

---

rfbaselineScale

*Estimation of the Scale Parameter*

---

## Description

Estimation of the scale parameter based on data smaller than its first mode. Mainly used in rfbaseline.

## Usage

```
rfbaselineScale(r)
```

## Arguments

r                    residuals

## Value

Estimated scale.

## Author(s)

Andreas Ruckstuhl

## See Also

See also [mad](#)

---

rose	<i>Creates a rose object out of circular data</i>
------	---

---

### Description

rose splits data into subsets according to one or two grouping elements, computes summary statistics for each, and returns the result in a rose object.

### Usage

```
rose(x, subset = NULL,
     cyclVar = NULL, circle = NULL, n.cyclVar = 8,
     cut = NULL, labels = NULL,
     breaks = NULL, include.lowest = FALSE, right = TRUE, dig.lab = 2,
     warn = TRUE, FUN = mean, ...)
```

### Arguments

x	Vector, data frame or matrix containing the response.
subset	An optional vector specifying a subset of observations to be used in the aggregating process.
cyclVar	Cyclic variable as first grouping element. cyclVar must be a numeric vector whose length is equal to the number of rows in x with $0 \leq \text{cyclVar} < \text{circle}$ . Observations where cyclic variables are NA are automatically excluded from the rose object.
circle	Defines the value of a full circle with no default.
n.cyclVar	Defines the number of equally spaced intervals of the cyclic variable, into which the data are split. The first interval is labeled with 0 and is always centered around 0.
cut	Vector of numerics, logicals or factors as second grouping elements. Its length is equal to the number of rows in x. cut is used to group the observations similar to function <code>cut</code> .
labels	Labels for the corresponding intervals. When cut is a logical, labels has to be named in the order: FALSE, TRUE.
breaks, include.lowest, right, dig.lab	These arguments are only active when cut is numeric and are used in exactly the same way as in <code>cut</code> . breaks defines the break points. include.lowest = TRUE indicates that the lowest (or highest, for right = FALSE) breaks value should be also included. right = TRUE indicates that the intervals should be closed on the right (and open on the left) or vice versa for right = FALSE. dig.lab number of digits for breaks labeling when labels are <i>not</i> given explicitly.

warn	Logical, indicating if warnings should be issued for NAs in <code>cyclVar</code> and / or <code>x</code> -values outside of breaks range.
FUN	Summary function, returning a scalar or vector.
...	Additional arguments for summary function.

### Details

The first grouping element, `cyclVar`, for the summary statistics must be circular and numeric. The second grouping element, `cut`, can be numeric, logical or a factor.

Not all combinations of arguments are allowed:

Argument `cut` can only be defined when summary consists of a scalar and `x` consists of 1 column.

When `x` contains only one column and `cut` is *not* defined, the summary function may also be a vector with the restriction, that the summary of each subset, defined by the cyclic variable, must have the same number of elements.

When `x` is a data frame or matrix with more than 1 column, the summary function must be scalar.

### Value

Object of class `rose`

### Author(s)

Rene Locher

### See Also

[rose-class](#), [plot.rose](#), [cart2clock](#), [clock2cart](#)

### Examples

```
## artificial example:
## concentration of A and B as function of
## hour of day (hour) and wind direction (WD)
hour <- rep(0:23,100)
dat <- data.frame(hour = hour,
                  A = (2*cos((hour+6)/6*pi)+rnorm(24*100,1))^2,
                  B = (2*cos((hour+4)/6*pi)+rnorm(24*100,1,2))^2,
                  WD = rnorm(24*100,
                             mean=sample(c(190,220,50),24*100, replace = TRUE),
                             sd=30)%%360)

## two different responses, scalar summary function
mean.windrose <- rose(dat[,c("A","B")],
                     cyclVar=dat$WD,
                     circle=360,
                     FUN=mean, na.rm=TRUE)

mean.windrose

## one response, vectorial summary function
quant.dayrose <- rose(dat$A,
                     cyclVar=dat$hour,
```

```

                                n.cyclVar=24, circle=24,
                                FUN=quantile, na.rm=TRUE)
quant.dayrose

mean.windroseB <- rose(dat[,c("A")],
                      cyclVar=dat$WD,
                      circle=360,
                      cut=dat$B,
                      breaks=c(0,30,100),
                      dig.lab=3,
                      FUN=mean, na.rm=TRUE)
mean.windroseB

```

---

 rose-class

*rose-class*


---

## Description

Summary statistics of cyclic data.

## Objects from the Class

Objects can be created by calls of the form `rose(x, cyclVar = NULL, circle = NULL, n.cyclVar = 8, cut = NULL, breaks = NULL, labels = NULL, dig.lab = 2, include.lowest = FALSE, subset = NULL, na.warning = TRUE, FUN = mean, ...)`

## Slots

**rho:** Object of class `matrix`. `rho[i, ]` contains the summary values of all data within the interval defined by the cyclic Variable `cyclVar[i]`. Column and row names are mandatory. The different columns of `rho` correspond to different responses when the summary statistics is a scalar *or* to the different elements of a vector summary of one response *or* to the different subsets of the second grouping element.

**cyclVar:** Object of class `numeric` containing the center of the interval of the cyclic variable. The values are sorted by increasing values, are unique and cannot contain NA values.

**circle:** Scalar of class `numeric`, defining the full circle.

## Author(s)

Rene Locher

## See Also

For the details of how to create and plot a [rose](#) object see [rose](#) and [plot.rose](#).

---

showColors                      *Displays vectors of colors*

---

### Description

Displays colors produced by a color vector `col` and labels them by the corresponding number of the element of `col`.

### Usage

```
showColors(col = IDPcolorRamp(20),
           ntm = min(length(col), 20),
           border = TRUE, mar = rep(0,4))
```

### Arguments

<code>col</code>	Color vector
<code>ntm</code>	Approximate number of labels printed
<code>border</code>	Shall border be drawn between the colors in the legend: TRUE / FALSE
<code>mar</code>	Margin. cf <a href="#">par</a>

### Author(s)

Rene Locher

### See Also

[IDPcolorRamp](#), [show.colors](#), [ColorBrewer](#)

### Examples

```
showColors(IDPcolorRamp(5))
showColors(IDPcolorRamp(200),border=FALSE)
showColors(IDPcolorRamp(200),border=FALSE,ntm=5)

showColors(IDPcolorRamp(4,
                      colInt = data.frame(
                        h = c(0.47, 0.28, 0.22, 0.2, 0.00),
                        s = c(0.3, 0.55, 0.75, 0.75, 0.75),
                        v = c(1, 1, 1, 1, 1)),
                      fr      = c(0.2,0.2,0.2))
           ,border=FALSE)

showColors(IDPcolorRamp(200,
                      t(col2hsv(c("darkviolet","blue",
                                  "green","yellow","red"))),
                      fr=rep(0.25,3)),border=FALSE)
```

---

title.control	<i>Auxiliary for Controlling the Title of a Rose Plot</i>
---------------	---

---

## Description

Auxiliary for controlling the title of a rose plot.

## Usage

```
title.control(text = NULL, cex = 1.5,  
             between = if (is.null(text)) 0 else 1)
```

## Arguments

text	Title.
cex	Size of characters in title in multiples of cex as defined in <a href="#">general.control</a> .
between	Distance between title and label North. Default units are <code>grid.control(cyclVar.cex)</code> . Other units can be defined by <a href="#">unit</a> .

## Value

Returns the arguments conveniently packaged up in a list to supply the arguments for the title of [plot.rose](#).

## Author(s)

Rene Locher

## See Also

[plot.rose](#), [general.control](#)

## Examples

```
title.control()
```

---

`zoom`*Zooming in and out in a 2d-Plot*

---

**Description**

Function to zoom in and out by mouse click in a 2D-plot.

**Usage**

```
zoom(fun = plot, zoom.col = "red", delay = 3, ...)
```

**Arguments**

<code>fun</code>	2D-plotting function
<code>zoom.col</code>	Color of clicked points
<code>delay</code>	Number of sec during which the 2 zooming points are shown on the plot before zooming
<code>...</code>	Arguments to plotting function

**Details**

When the clicked points lay within the plot region range, the points define the new plotting limits. When the clicked points lay in the margin, the plotting limits will be moved into the corresponding direction by 1/3 out of the actual range. There is no special sequential order for the zooming points required. The zooming function is stopped by right clicking and choosing the menu item "stop".

**Value**

No value returned.

**Author(s)**

Rene Locher

**Examples**

```
i <- 1:100
y <- i*sin(i*(pi/16))
y <- c(rev(y),y)

## Not run:
zoom(fun=plot, zoom.col="red", x=1:200, y=y, type="l", xlab="index")
## End( Not run)
```

# Index

- \*Topic **aplot**
    - Arrows, 3
    - Image, 19
  - \*Topic **category**
    - rose, 47
  - \*Topic **classes**
    - rose-class, 49
  - \*Topic **color**
    - col2hsv, 7
    - IDPcolorRamp, 15
    - showColors, 50
  - \*Topic **datasets**
    - MS, 34
  - \*Topic **dplot**
    - cart2clock, 5
    - clock2cart, 6
    - col2hsv, 7
    - draw.leg, 8
    - general.control, 11
    - grid.control, 14
    - IDPmisc-package, 2
    - key.control, 30
    - peaks, 37
    - rose, 47
    - title.control, 51
  - \*Topic **dynamic**
    - zoom, 52
  - \*Topic **error**
    - getXY, 13
  - \*Topic **hplot**
    - IDPmisc-package, 2
    - ilagplot, 17
    - ipairs, 21
    - ipanel.smooth, 23
    - iplot, 25
    - iplotLegend, 27
    - itermplot, 28
    - longtsPlot, 31
    - plot.rose, 39
    - poster.plot, 43
    - showColors, 50
  - \*Topic **iplot**
    - longtsPlot, 31
    - zoom, 52
  - \*Topic **manip**
    - cart2clock, 5
    - clock2cart, 6
    - data.sheet, 8
    - getXY, 13
    - IDPmisc-package, 2
    - NaRV.omit, 35
    - ok, 36
    - peaks, 37
  - \*Topic **methods**
    - rose-class, 49
  - \*Topic **multivariate**
    - IDPmisc-package, 2
    - longtsPlot, 31
  - \*Topic **package**
    - IDPmisc-package, 2
  - \*Topic **regression**
    - rfbaseline, 44
  - \*Topic **robust**
    - rfbaseline, 44
  - \*Topic **smooth**
    - rfbaseline, 44
  - \*Topic **ts**
    - IDPmisc-package, 2
    - longtsPlot, 31
  - \*Topic **univar**
    - rfbaselineScale, 46
  - \*Topic **utilities**
    - cart2clock, 5
    - clock2cart, 6
    - IDPmisc-package, 2
- AirQual, 40  
Arrows, 3  
arrows, 4

- cart2clock, [5](#), [6](#), [48](#)
- clock2cart, [5](#), [6](#), [48](#)
- col2hsv, [7](#)
- col2rgb, [7](#)
- ColorBrewer, [16](#), [50](#)
- cut, [47](#)
  
- data.sheet, [8](#)
- dev.print, [33](#)
- draw.key, [8](#), [9](#)
- draw.leg, [8](#)
  
- gam, [29](#)
- general.control, [11](#), [14](#), [15](#), [30](#), [39](#), [51](#)
- getXY, [13](#)
- glm, [29](#)
- gplot.hexbin, [22](#), [26](#)
- graphics, [4](#), [20](#)
- grid, [40](#)
- grid.control, [12](#), [13](#), [14](#), [39](#)
  
- hsv, [7](#), [15](#)
  
- IDPcolorRamp, [15](#), [27](#), [50](#)
- IDPmisc (IDPmisc-package), [2](#)
- IDPmisc-package, [2](#)
- ilagplot, [17](#), [20](#), [22](#), [24–26](#), [29](#)
- Image, [19](#), [19](#), [22](#), [26](#)
- image, [20](#)
- ipairs, [19](#), [20](#), [21](#), [24–26](#), [29](#)
- ipanel.smooth, [23](#), [29](#)
- iplot, [19](#), [20](#), [22](#), [25](#)
- iplotLegend, [27](#)
- itermplot, [28](#)
  
- key.control, [30](#), [39](#)
  
- lag.plot, [18](#)
- lines, [24](#)
- lm, [29](#)
- loess, [45](#)
- longtsPlot, [31](#)
- lowess, [24](#), [29](#), [45](#)
  
- mad, [46](#)
- MS, [34](#)
  
- na.omit, [36](#)
- NaRV.omit, [35](#)
  
- ok, [36](#)
  
- p.arrows, [3](#), [4](#)
- pairs, [22](#)
- par, [3](#), [18](#), [21](#), [22](#), [26](#), [27](#), [29](#), [32](#), [33](#), [43](#), [50](#)
- peaks, [37](#), [38](#)
- plot, [31](#), [33](#)
- plot (plot.rose), [39](#)
- plot,rose,missing-method (plot.rose), [39](#)
- plot.rose, [12](#), [13](#), [15](#), [31](#), [39](#), [48](#), [49](#), [51](#)
- poster.plot, [43](#)
  
- rfbaseline, [44](#)
- rfbaselineScale, [46](#)
- rgb2hsv, [7](#)
- rose, [12](#), [30](#), [39](#), [40](#), [47](#), [49](#)
- rose-class, [5](#), [6](#), [40](#)
- rose-class, [48](#), [49](#)
  
- savePlot, [33](#)
- show.colors, [50](#)
- showColors, [16](#), [50](#)
  
- termplot, [29](#), [30](#)
- title.control, [39](#), [51](#)
  
- unit, [12](#), [51](#)
  
- xy.coords, [13](#)
- xyplot, [9](#), [10](#)
  
- zoom, [52](#)