Package ‘plot3D’

December 18, 2019

Version 1.3
Title Plotting Multi-Dimensional Data
Author Karline Soetaert <karline.soetaert@nioz.nl>
Maintainer Karline Soetaert <karline.soetaert@nioz.nl>
Depends R (>= 2.15)
Imports misc3d, stats, graphics, grDevices, methods
Description Functions for viewing 2-D and 3-D data, including perspective plots, slice plots, surface plots, scatter plots, etc. Includes data sets from oceanography.
License GPL (>= 3.0)
LazyData yes
Repository CRAN
Repository/R-Forge/Project plot3d
Repository/R-Forge/Revision 124
Repository/R-Forge/DateTimeStamp 2019-12-18 08:14:15
Date/Publication 2019-12-18 10:50:02 UTC
NeedsCompilation no

R topics documented:

  plot3D-package .............................................. 2
  2-D data set ................................................. 4
  2D image and contour plots .................................. 6
  3-D arrows, segments, polygons, boxes, rectangles .......... 13
  3-D contours .................................................. 20
  3-D data set .................................................. 23
  3-D perspectives ............................................. 25
  3-D surfaces .................................................. 34
  3-D volume visualisation .................................... 40
  Color key legend ............................................. 47
  Colors ......................................................... 51
  Composite plots ............................................... 55
plot3D-package

Plotting multi-dimensional data.

Description

Functions for visualising 2-D and 3-D data.
Many of the functions are extensions of R’s persp or image function.
Other packages that provide visualisation of 3-D data (and which might be better suited) are: rgl, scatterplot3D, misc3D.

Note

This package is dedicated to Carlo.

Note

Some of the functions based on persp will not work properly for all values of phi (which turns the plots upside-down). This is because an assumption is made as to how the perspective plots are viewed.

Author(s)

Karline Soetaert

References

http://www.rforscience.com/rpackages/visualisation/oceanview/
http://www.rforscience.com/rpackages/visualisation/plot3d/

See Also

Functions that are based on the persp function:

- persp3D: an extended version of persp.
- ribbon3D: a perspective plot as ribbons.
- hist3D: 3-D histograms.
- scatter3D, points3D, lines3D: colored points, lines, ... in 3-D.
- slice3D, slicecont3D: slices from a full 3-D data set.
- isosurf3D: isosurfaces from a full 3-D data set as triangles.
• voxel3D: isosurfaces from a full 3-D data set as points.
• surf3D, spheresurf3D: 3-D shapes or surfaces.
• arrows3D: arrows in 3-D.
• segments3D: line segments in 3-D.
• polygon3D: 3-D polygons.
• box3D, border3D, rect3D: boxes and rectangles in 3-D.
• text3D: labels in 3-D.

Functions defined on the image function:
• image2D, for an image function to visualise 2-D or 3-D data.
• ImageOcean: an image of the ocean's bathymetry.

Other plotting functions:
• contour2D, for a contour function to visualise 2-D data and that have a color key.
• scatter2D: colored points, lines, ... in 2-D.
• text2D, arrows2D, segments2D, rect2D, polygon2D for other 2D functions that have a color key.

Colors and colorkey:
• colkey: adds a color legend.
• jet.col, ramp.col, gg.col, alpha.col: suitable colors, shade and lighting.

Utility functions:
• mesh: to generate rectangular (x, y) or (x, y, z) meshes.

Data sets:
• Oxsat: 3-D data set with the ocean's oxygen saturation values.
• Hypsometry: 2-D data set with the worlds elevation and ocean's bathymetry.

Examples
# run all examples
## Not run:
  example(persp3D)
  example(surf3D)
  example(slice3D)
  example(scatter3D)
  example(segments3D)
  example(image2D)
  example(image3D)
  example(contour3D)
  example(colkey)
  example(jet.col)
  example(perspbox)
  example(mesh)
2-D data set

The earths hypsometry (land elevation) and the ocean’s bathymetry

Description

Hypsometry is a relatively crude data set of the earths land elevation (positive) and ocean depth (negative), at 1 dg intervals.

ImageOcean plots the ocean’s bathymetry.

Usage

ImageOcean (...)  
Hypsometry

Arguments

... arguments passed to function image2D.

Format

A list with the bathymetry (depth) and hypsometry (altitude) of the world. It contains:

x the latitude,
y the longitude,
z the height (m).

Details

Hypsometry is based on dataset Bathymetry from the R-package marelac.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

image2D, for the image function that visualises the bathymetry
Examples

# save plotting parameters
pm <- par("mfrow")
mar <- par("mar")

## =======================================================================
## Images of the hypsometry
## =======================================================================
par(mfrow = c(2, 2))
image2D(Hypsometry, asp = TRUE, xlab = expression(degree*E),
        ylab = expression(degree*N), contour = TRUE)

# remove ocean
zz <- Hypsometry$z
zz[zz < 0] <- NA
image2D(zz, x = Hypsometry$x, y = Hypsometry$y, NAcol = "black")

## =======================================================================
## A short version for plotting the Ocean's bathymetry
## =======================================================================
ImageOcean(asp = TRUE, contour = TRUE)
ImageOcean(col = "white",
          contour = list(levels = seq(-6000, 0, by = 2000)))

## =======================================================================
## A complex image of part of the ocean
## =======================================================================
# elaborate version
ii <- which(Hypsometry$x > -50 & Hypsometry$x < -20)
jj <- which(Hypsometry$y > 10 & Hypsometry$y < 40)

# Draw empty persp box
zlim <- c(-10000, 0)
pmat <- perspbox(z = Hypsometry$z[ii, jj],
                 xlab = "longitude", ylab = "latitude", zlab = "depth",
                 expand = 0.5, d = 2, zlim = zlim, phi = 20, theta = 30,
                 colkey = list(side = 1))

# A function that makes a black panel with grey edge:
panelfunc <- function(x, y, z) {
    XY <- trans3D(x, y, z, pmat = pmat)
    polygon(XY$x, XY$y, col = "black", border = "grey")
}

# left panel
panelfunc(x = c(0, 0, 0, 0), y = c(0, 0, 1, 1),
          z = c(zlim[1], zlim[2], zlim[2], zlim[1]))
2D image and contour plots

Extended image and contour plots for 2-D (and 3-D) data.

Description

image2D extends R’s image function. Input can be a matrix (2-D) or an array (3-D) or a list.
contour2D extends R’s contour function.

Usage

image2D (z, ...)  
contour2D (z, x = seq(0, 1, length.out = nrow(z)),  
y = seq(0, 1, length.out = ncol(z)), ...,  
col = NULL, NACol = NULL,  
colkey = NULL, resfac = 1,  
clab = NULL, add = FALSE, plot = TRUE)

## S3 method for class 'matrix'
image2D(z, x = seq(0, 1, length.out = nrow(z)),  
y = seq(0, 1, length.out = ncol(z)), ...,
2D image and contour plots

```r
y = seq(0, 1, length.out = ncol(z)), colvar = z, 
col = NULL, NAcol = "white", breaks = NULL,
border = NA, facets = TRUE, contour = FALSE,
colkey = NULL, resfac = 1, clab = NULL,
lighting = FALSE, shade = NA, ltheta = -135, lphi = 0,
theta = 0, rasterImage = FALSE,
add = FALSE, plot = TRUE)
```

## S3 method for class 'array'
image2D(z, margin = c(1, 2), subset, ask = NULL, ...)
## S3 method for class 'list'
image2D(z, ...)

### Arguments

- **z**
  - Matrix (2-D) or array (3-D) or a list with matrices or arrays, with z-values. By default colvar is equal to z, hence z also defines the variable used to color the image. Only when shade or lighting is toggled on does it make sense to use z different from colvar.

- **x, y**
  - Vectors or matrix with x and y values. If a vector x should be of length equal to nrow(z) and y should be of length equal to ncol(z). If a matrix (only for `image2D`), they should have the same dimension as z or be of dimension = dim(z)+1.

- **colvar**
  - Only used when shade or lighting is toggled on. The variable used to color the image.

- **col**
  - Color palette to be used for the image function or for the contours. See details.

- **NAcol**
  - Color to be used for NA values of z; for `image2D`, the default is “white”, for `contour2D`, the default is to do nothing.

- **breaks**
  - A set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.

- **contour**
  - If TRUE, then a contour plot will be added to the image plot, unless x, y are a matrix. Also allowed is to pass a list with arguments for the contour function.

- **colkey**
  - A logical, NULL (default), or a list with parameters for the color key (legend).
  - List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab, and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col.
  - The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = NULL, line.clab = NULL, cex.clab = NULL, font.clab = NULL) See colkey.
  - The default is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. If colkey = FALSE, no color key legend will be added.

- **clab**
  - Only if colkey is not NULL or FALSE, the label to be written on top of the color key. The label will be written at the same level as the main title. To lower it, clab can be made a vector, with the first values empty strings.
resfac  Resolution factor, one value or a vector of two numbers, for the x and y-values respectively. A value > 1 will increase the resolution. For instance, if resfac equals 3 then for each adjacent pair of x- and y-values, z will be interpolated to two intermediary points. This uses simple linear interpolation. If resfac is one number then the resolution will be increased similarly in x and y-direction.

lighting  If not FALSE the facets will be illuminated, and colors may appear more bright. To switch on lighting, the argument lighting should be either set to TRUE (using default settings) or it can be a list with specifications of one of the following: ambient, diffuse, specular, exponent, sr and alpha.

Will overrule shade not equal to NA.

See examples in jet.col.

shade  the degree of shading of the surface facets. Values of shade close to one yield shading similar to a point light source model and values close to zero produce no shading. Values in the range 0.5 to 0.75 provide an approximation to daylight illumination. See persp.

ltheta, lphi  if finite values are specified for ltheta and lphi, the surface is shaded as though it was being illuminated from the direction specified by azimuth ltheta and colatitude lphi. See persp.

theta  The angle defining the azimuthal direction. Implemented for consistency with the other functions based on persp.

border  The color of the lines drawn around the surface facets. The default, NA, will disable the drawing of borders.

facets  If TRUE, then col denotes the color of the surface facets. If FALSE, then the surface facets are colored “white” and the border will be colored as specified by col. If NA then the facets will be transparent. It is usually faster to draw with facets = FALSE.

rasterImage  If TRUE, the function rasterImage will be used for plotting rather than image or polygon. This requires the x and y to be a vector with equally spaced elements. Note that by default, rasterImage linearly interpolates the image, so it will appear smoother.

add  Logical. If TRUE, then the points will be added to the current plot. If FALSE a new plot is started.

plot  Logical. If TRUE (default), a plot is created, otherwise (for 3D plots) the viewing transformation matrix is returned (as invisible).

margin  A vector giving the subscripts which the image function will be applied over. The image function will loop over the index that is not in margin. For instance, c(1, 2), indicates to plot rows(x) and columns(y) and to loop over index 3; c(2, 1) will do the same but the image will be transposed. margin should be a vector with two numbers inbetween 1, and 3.

ask  A logical; if TRUE, the user is asked before each plot, if NULL the user is only asked if more than one page of plots is necessary and the current graphics device is set interactive, see par(ask) and dev.interactive.

subset  Either a logical expression indicating over which elements to loop, or a vector or integers denoting the indices of the elements over which to loop. Missing values are taken as FALSE.
additional arguments passed to the plotting methods image, rasterImage, polygon and contour.
alpha can be given a value inbetween 0 and 1 to make colors transparent.
The arguments after ... must be matched exactly.

Details
image2D is an extension to the default image plot that has the possibility to add a color key and contourlines, and to increase the resolution in order to make smoother images. It also uses a different color scheme, it can deal with decreasing x- and y- values and x and y can be a matrix. In the latter case, the image will be drawn as a set of polygons; if x and y are a vector, either R-function image or rasterImage will be used.
image2D.array and image2D.list are versions that accept a 3 dimensional array respectively a list with z-matrices as their first argument to produce multiple plots.
For argument col of the image2D function, both NA and NULL are allowed, in which case the color will be white, and no color key will be drawn.
To set the ranges of the z-variable, both arguments zlim (as in image) and clim (as in the other plot3D functions) are accepted.
Upon returning from the image2D and contour2D functions, the figure coordinates are defined by the main figure (excluding the color key). Thus, one can safely add other plotting elements.

Value
Returns nothing.

Note
The first argument, z generally determines the color variable. For consistency with the other functions, another variable, colvar is also defined and set by default equal to z. colvar will only be used if shade or lighting are toggled on. In this case, z will be used to define the shading (orientation of each facet), while colvar will define the color.
When x and y is a vector, the function uses R-function image. This means that the x- and y- axis will extend the x- and y- values with half a grid cell.
In contrast, when x and y are a matrix, the axis will not extend the x- or y- values. See first example.

Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

See Also
jet.col, ImageOcean, Oxsat, persp3D, scatter2D for other examples where image2D is used.
image and contour for the original R functions.
plot.image from the fields package.
Examples

```r
# save plotting parameters
pm <- par("mfrow")

## =======================================================================
## Difference between x or y a vector/matrix and rasterImage
## =======================================================================

par(mfrow = c(2, 2))
x <- y <- 1:3
z <- matrix(nrow = 3, ncol = 3, data = 1:9)
image2D(z, x, y, border = "black")
image2D(z, x, y, rasterImage = TRUE, border = "black")
image2D(z, x = matrix(nrow = 3, ncol = 3, data = x),
       y, border = "black")
image2D(z, x, y, border = "black", theta = 45)

## =======================================================================
## shading, light, adding contours, points and lines
## =======================================================================

par(mfrow = c(2, 2))
nr <- nrow(volcano)
nc <- ncol(volcano)
image2D(volcano, x = 1:nr, y = 1:nc, lighting = TRUE,
        main = "volcano", clab = "height, m")
abline(v = seq(10, 80, by = 10))
abline(h = seq(10, 60, by = 10))
points(50, 30, pch = 3, cex = 5, lwd = 3, col = "white")
image2D(z = volcano, x = 1:nr, y = 1:nc, lwd = 2, shade = 0.2,
        main = "volcano", clab = "height, m")
image2D(volcano, x = 1:nr, y = 1:nc, contour = TRUE, shade = 0.5, lphi = 0,
        col = "lightblue", main = "volcano")
breaks <- seq(90, 200, by = 10)
image2D(volcano, x = 1:nr, y = 1:nc, col = jet.col(length(breaks)-1),
        main = "volcano", clab = "height, m", breaks = breaks)

## =======================================================================
## Contour plots
## =======================================================================

par(mfrow = c(2, 2))
V <- volcano - 150
image2D(z = V, colkey = FALSE, lwd = 2)

# default, no color key
contour2D(z = V, colkey = FALSE, lwd = 2)
```
# imposed levels
contour2D(z = V, lwd = 2, levels = seq(-40, 40, by = 20))

# negative levels dashed
contour2D(z = V, col = "black", lwd = 2,
          levels = seq(0, 40, by = 20))
contour2D(z = V, col = "black", lwd = 2, lty = 2,
          levels = seq(-40, -20, by = 20), add = TRUE)

# no labels, imposed number of levels, colorkey
contour2D(z = V, lwd = 2, nlevels = 20, drawlabels = FALSE,
          colkey = list(at = seq(-40, 40, by = 20)))

## =======================================================================
## A large data set, input is an array
## =======================================================================

par(mfrow = c(1, 1))

image2D(z = Oxsat$val[, , 1], x = Oxsat$lon, y = Oxsat$lat,
        main = "surface oxygen saturation data 2005", NAcol = "black",
        clab = c("","","\%"))

# images at first 9 depths - use subset to select them
image2D(z = Oxsat$val, subset = 1:9, x = Oxsat$lon, y = Oxsat$lat,
         margin = c(1, 2), NAcol = "black",
         clab = "longitude", ylab = "latitude",
         zlim = c(0, 115),
         main = paste("depth ", Oxsat$depth[1:9], " m"),
         mfrow = c(3, 3))

# images at latitude - depth section - increase resolution
z <- Oxsat$val[, Oxsat$lat > - 5 & Oxsat$lat < 5, ]
image2D(z = z, x = Oxsat$lon, y = Oxsat$depth,
         margin = c(1, 3), NAcol = "black",
         resfac = 3, ylim = c(5000, 0))

# show position of transects
image2D(z = Oxsat$val[, ,],
         x = Oxsat$lon, y = Oxsat$lat,
         NAcol = "black")
aline(h = Oxsat$lat[Oxsat$lat > - 5 & Oxsat$lat < 5])

## =======================================================================
## Image of a list of matrices
## =======================================================================

listvolcano <- list(volcano = volcano, logvolcano = log(volcano))

image2D(listvolcano, x = 1:nr, y = 1:nc, contour = TRUE,
        main = c("volcano", "log(volcano)")
        clab = list("height, m", "log(m)"),
        zlim = list(c(80, 200), c(4.4, 5.5)))
# Image of a list of arrays

# Not run:
# crude conversion from oxsat to oxygen
listoxygen <- list(Oxsat$val, Oxsat$val/100 * 360)

image2D(z = listoxygen,
        x = Oxsat$lon, y = Oxsat$lat,
        margin = c(1, 2), NAcol = "black",
        main = c("Oxygen saturation ", " Oxygen concentration"),
        mtext = paste("depth ", Oxsat$depth, " m")
)

# End(Not run)

'x', 'y' and 'z' are matrices

par(mfrow = c(2, 1))

# tilted x- and y-coordinates of 'volcano'
volcx <- matrix(nrow = 87, ncol = 61, data = 1:87)
volcx <- volcx + matrix(nrow = 87, ncol = 61,
                        byrow = TRUE, data = seq(0., 15, length.out = 61))

volcy <- matrix(nrow = 87, ncol = 61, data = 1:61)
volcy <- t(volcy + matrix(ncol = 87, nrow = 61,
                          byrow = TRUE, data = seq(0., 25, length.out = 87)))

image2D(volcano, x = volcx, y = volcy)

# x and y can also be of dimension dim(z)+1:
# Not run:
# tilted x- and y-coordinates of 'volcano'
volcx <- matrix(nrow = 88, ncol = 62, data = 1:88)
volcx <- volcx + matrix(nrow = 88, ncol = 62,
                        byrow = TRUE, data = seq(0., 15, length.out = 62))

volcy <- matrix(ncol = 88, nrow = 62, data = 1:62)
volcy <- t(volcy + matrix(ncol = 88, nrow = 62,
                          byrow = TRUE, data = seq(0., 25, length.out = 88)))

image2D(volcano, x = volcx, y = volcy)

# use of panel function
image2D(volcano, x = volcx, y = volcy, NAcol = "black",
        panel.first = substitute(box(col = "lightgrey", lwd = 30)))
3-D arrows, segments, polygons, boxes, rectangles

```r
## Image with NAs and logs

par(mfrow = c(2, 2))
# normal volcano
image2D(volcano, clab = c("height", "m"))

# logarithmic z-axis
image2D(volcano, log = "z", clab = c("height", "m"),
main = "log='z'")

# Including NAs
VOLC <- volcano - 110
VOLC[VOLC <= 0] <- NA
image2D(VOLC, main = "including NAs and rescaled")

# both
image2D(VOLC, NAcol = "black", log = "z", zlim = c(1, 100),
main = "NAs and log='z'")

## Image with contour specification (alpha sets the transparency)

par(mfrow = c(1, 1))
image2D(volcano, shade = 0.2, rasterImage = TRUE,
contour = list(col = "white", labcex = 0.8, lwd = 3, alpha = 0.5))

# same:
## Not run:
image2D(z = volcano, shade = 0.2, rasterImage = TRUE)
contour2D(z = volcano, col = "white", labcex = 0.8,
lwd = 3, alpha = 0.5, add = TRUE)

## End(Not run)
# reset plotting parameters
par(mfrow = pm)
```

### Description

Functions arrows3D and segments3D draw arrows and line segments between pairs of points.
Functions box3D and border3D draw boxes between pairs of points.
polygon3D draws polygons; rect3D draws rectangles.
The 2D functions arrows2D, segments2D, rect2D and polygon2D are included for their side effect of having a color key.
Usage

arrows3D(x0, y0, z0, x1 = x0, y1 = y0, z1 = z0, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", type = "triangle", add = FALSE, plot = TRUE)

segments3D(x0, y0, z0, x1 = x0, y1 = y0, z1 = z0, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", add = FALSE, plot = TRUE)

box3D(x0, y0, z0, x1, y1, z1, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, border = NA, facets = TRUE, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", add = FALSE, plot = TRUE)

border3D(x0, y0, z0, x1, y1, z1, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", add = FALSE, plot = TRUE)

rect3D(x0, y0, z0, x1 = NULL, y1 = NULL, z1 = NULL, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, border = NA, facets = TRUE, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", add = FALSE, plot = TRUE)

polygon3D(x, y, z, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, border = NA, facets = TRUE, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", add = FALSE, plot = TRUE)

arrows2D(x0, y0, x1 = x0, y1 = y0, ..., colvar = NULL, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, clim = NULL, clab = NULL, type = "triangle", add = FALSE, plot = TRUE)
segments2D (x0, y0, x1 = x0, y1 = y0, ..., colvar = NULL, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, clim = NULL, clab = NULL, add = FALSE, plot = TRUE)

rect2D (x0, y0, x1 = x0, y1 = y0, ..., colvar = NULL, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, clim = NULL, clab = NULL, add = FALSE, plot = TRUE)

polygon2D (x, y, ..., colvar = NULL, col = NULL, NAcol = "white", breaks = NULL, border = NA, facets = TRUE, colkey = NULL, clim = NULL, clab = NULL, add = FALSE, plot = TRUE)

Arguments

- **x0, y0, z0** coordinates of points from which to draw.
- **x1, y1, z1** coordinates of points to which to draw. For arrows3D and segments3D, at least one must be supplied. For rect3D exactly one must be NULL.
- **x, y, z** coordinates of the vertices of the polygon. The polygon will be closed by joining the last point to the first point. The coordinates can contain missing values (NA). These NA values break the polygon into several complete polygons.
- **colvar** The variable used for coloring. It need not be present, but if specified, it should be a vector of dimension equal to the coordinates or to the number of polygons. Values of NULL, NA, or FALSE will toggle off coloration according to colvar.
- **theta, phi** the angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude. See persp.
- **col** Color palette to be used for coloring the arrows or segments as specified by the colvar variable. If col is NULL and colvar is specified, then a red-yellow-blue colorscheme (jet.col) will be used. If col is NULL and colvar is not specified, then col will be "black".
- **NAcol** Colors to be used for colvar values that are NA.
- **breaks** a set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.
- **colkey** A logical, NULL (default), or a list with parameters for the color key (legend). List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col. The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL. See colkey. The defaults is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. If colkey = FALSE, no color key legend will be added.
border The color of the lines drawn around the surface facets. The default, NA, will disable the drawing of borders.

facets If TRUE, then col denotes the color of the surface facets. If FALSE, then the surface facets are colored “white” and the border (if NA) will be colored as specified by col. If NA then the facets will be transparent. It is usually faster to draw with facets = FALSE.

panel.first A function to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful e.g. for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix, e.g. it should be defined as function(pmat). See example of persp3D and last example of voxel3D.

clab Only if colkey is not NULL or FALSE, the label to be written on top of the color key. The label will be written at the same level as the main title. To lower it, clab can be made a vector, with the first values empty strings.

clim Only if colvar is specified, the range of the color variable, used for the color key. Values of colvar that extend the range will be put to NA.

bty The type of the perspective box, the default draws only the back panels. Only effective if the persp argument (box) equals TRUE (this is the default). See perspbox.

type The type of the arrow head, one of "simple" (which uses R-function arrows), "curved" or "triangle" and "cone". The latter two are the same in plot3D (but differ in package plot3Drgl).

add Logical. If TRUE, then the arrows, segments, ... will be added to the current plot. If FALSE a new plot is started.

plot Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

... additional arguments passed to the plotting methods.

The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, scale, expand, box, axes, ntabs. The arguments xlim, ylim, zlim only affect the axes for 3D plots. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.

shade and lighting arguments will have no effect.

alpha can be given a value inbetween 0 and 1 to make colors transparent.

In addition, the perspbox arguments col.axis, col.panel, lwd.panel, col.grid, lwd.grid can also be given a value.

For arrows3D, the following arrows arguments can be specified: length, code, angle.

For polygon3D, the following polygon arguments can be specified: border.

For all the functions, arguments lty, lwd can be specified.

The arguments after ... must be matched exactly.

Value

Returns the viewing transformation matrix.

See trans3D.
See Also

`arrows` for the 2-D arrows function on which `arrows3D` is based.
`segments` for the 2-D arrows function on which `segments3D` is based.

Examples

```r
# save plotting parameters
pm <- par("mfrow")

## arrows, points, segments, box

# Create a grid of x, y, and z values
xx <- yy <- seq(-0.8, 0.8, by = 0.2)
zz <- seq(-0.8, 0.8, by = 0.8)
M <- mesh(xx, yy, zz)
x0 <- M$x; y0 <- M$y; z0 <- M$z
x1 <- x0 + 0.1
Col <- c("red", "blue", "green")
arrows3D(x0, y0, z0, x1 = x1, colvar = z0, lwd = 2,
d = 2, clab = "z-value", col = Col, length = 0.1,
xlim = c(-0.8, 0.8), ylim = c(-0.8, 0.8),
main = "arrows3D, points3D, segments3D, border3D")

# add starting point of arrows
points3D(x0, y0, z0, add = TRUE, colvar = z0,
colkey = FALSE, pch = ".", cex = 3, col = Col)

# use segments to add section
x0 <- c(-0.8, 0.8, 0.8, -0.8)
x1 <- c( 0.8, 0.8, -0.8, -0.8)
y0 <- c(-0.8, -0.8, 0.8, -0.8)
y1 <- c(-0.8, 0.8, 0.8, 0.8)
z0 <- c(0., 0., 0., 0.)
segments3D(x0, y0, z0, x1, y1, z1 = z0,
add = TRUE, col = "black", lwd = 2)

# add a box
border3D(-0.8, -0.8, -0.8, 0.8, 0.8, 0.8,
col = "orange", add = TRUE, lwd = 3)

## boxes, cubes

# borders are boxes without facets
border3D(x0 = seq(-0.8, -0.1, by = 0.1),
```

3-D arrows, segments, polygons, boxes, rectangles
3-D arrows, segments, polygons, boxes, rectangles

```r
y0 = seq(-0.8, -0.1, by = 0.1),
z0 = seq(-0.8, -0.1, by = 0.1),
x1 = seq(0.8, 0.1, by = -0.1),
y1 = seq(0.8, 0.1, by = -0.1),
z1 = seq(0.8, 0.1, by = -0.1),
col = gg.col(8), lty = 2,
lwd = c(1, 4), phi = 20, main = "border3D")

box3D(x0 = -0.8, y0 = -0.8, z0 = -0.8,
x1 = 0.8, y1 = 0.8, z1 = 0.8,
border = "black", lwd = 2,
col = gg.col(1, alpha = 0.8),
main = "box3D")

box3D(x0 = seq(-0.8, -0.1, by = 0.1),
y0 = seq(-0.8, -0.1, by = 0.1),
z0 = seq(-0.8, -0.1, by = 0.1),
x1 = seq(0.8, 0.1, by = -0.1),
y1 = seq(0.8, 0.1, by = -0.1),
z1 = seq(0.8, 0.1, by = -0.1),
col = rainbow(n = 8, alpha = 0.1),
border = "black", lwd = 2, phi = 20)

# here the perspective does not always work
# use alpha.col to set the transparency of a vector of colors
box3D(x0 = runif(3), y0 = runif(3), z0 = runif(3),
x1 = runif(3), y1 = runif(3), z1 = runif(3),
col = c("red", "lightblue", "orange"), alpha = 0.5,
border = "black", lwd = 2)
```

## rectangles

```r
rect3D(x0 = seq(-0.8, -0.1, by = 0.1),
y0 = seq(-0.8, -0.1, by = 0.1),
z0 = seq(-0.8, -0.1, by = 0.1),
x1 = seq(0.8, 0.1, by = -0.1),
y1 = seq(0.8, 0.1, by = -0.1),
z1 = seq(0.8, 0.1, by = -0.1),
col = gg.col(8), border = "black",
bty = "g", lwd = 2, phi = 20, main = "rect3D")
```

## arrows added to a persp plot

```r
rect3D(x0 = 0, y0 = 0, z0 = 0, x1 = 1, y1 = 1,
ylim = c(0, 1), facets = NA, border = "red",
bty = "g", lwd = 2, phi = 20)
```

# add rect at constant z, with colored facet
rect3D(x0 = 0, y0 = 0, z0 = 0, x1 = 1, y1 = 1,
border = "red", add = TRUE)
```
## 3-D arrows, segments, polygons, boxes, rectangles

```r
## ========================================================================
x <- y <- seq(-10, 10, length = 30)
z <- outer(x, y, FUN = function(x,y) x^2 + y^2)

persp3D(x, y, z, theta = 30, phi = 30,
        col = "lightblue", ltheta = 120, shade = 0.75,
        ticktype = "detailed", xlab = "X",
ylab = "Y", zlab = "x^2+y^2" )

# Points where to put the arrows
x <- y <- seq(-10, 10, len = 6)
X0 <- outer(x, y, FUN = function (x,y) x)
Y0 <- outer(x, y, FUN = function (x,y) y)
Z0 <- outer(x, y, FUN = function (x,y) x^2 + y^2)

X1 <- X0 + 1
Y1 <- Y0 + 1
Z1 <- Z0 + 10

arrows3D(X0, Y0, Z0, X1, Y1, Z1, lwd = 2,
         add = TRUE, type = "curved", col = "red")

dsegments3D(X0, Y0, Z0, X0, Y0, rep(0, length(X0)), lwd = 2,
            add = TRUE, col = "green")

## ========================================================================

## polygon3D

x <- runif(10)
y <- runif(10)
z <- runif(10)

polygon3D(x, y, z)

# several polygons, separated by NAs
x <- runif(39)
y <- runif(39)
z <- runif(39)
ii <- seq(4, 36, by = 4)
x[ii] <- y[ii] <- z[ii] <- NA

# transparent colors (alpha)
polygon3D(x, y, z, border = "black", lwd = 3,
          col = gg.col(length(ii) + 1, alpha = 0.8),
          main = "polygon3D")

## ========================================================================

## 2D examples, with color key

arrows2D(x0 = runif(10), y0 = runif(10),
```
3-D contours

Contours in 3-D plots.

Description

contour3D adds a contour in a 3-D plot.

Usage

contour3D (x = NULL, y = NULL, z = NULL, 
... , colvar = NULL, phi = 40, theta = 40, 
col = NULL, colkey = NULL,
3-D contours

\begin{verbatim}
panel.first = NULL, clim = NULL, clab = NULL, bty = "b",
dDepth = 1e-1, addbox = TRUE, add = FALSE, plot = TRUE)
\end{verbatim}

Arguments

\textbf{x, y, z} \quad \text{Matrix (2-D), vector, or one value containing the values where the image is to be plotted. At least one of them should be one number, as this will determine where the image is plotted, parallel to the (y-z) plane (x one number), to the (x-z) plane (y one number) or to the (z-y) plane (z one number). If two are vectors, the first vector should be of length equal to nrow(colvar) and the second should be of length equal to ncol(colvar).}

\textbf{colvar} \quad \text{The variable used for coloring. Values of colvar equal to NULL, NA, or FALSE will toggle off coloration according to colvar. This gives good results only if border is given a color, or when shade is >0 (see \texttt{persp}).}

\textbf{col} \quad \text{Color palette to be used for the colvar variable. If col is NULL and colvar is specified, then a red-yellow-blue colorscheme (jet.col) will be used. If col is NULL and colvar is not specified, then col will be "black".}

\textbf{colkey} \quad \text{A logical, NULL (default), or a list with parameters for the color key (legend). List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab, and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col.axis, col.ticks, xpd, adj, cex.axis, mgp, tck, tcl, las. The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL) See \texttt{colkey}. The default is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. if colkey = FALSE, no color key legend will be added.}

\textbf{clab} \quad \text{Only if colkey = TRUE, the label to be written on top of the color key. The label will be written at the same level as the main title. to lower it, clab can be made a vector, with the first values empty strings.}

\textbf{clim} \quad \text{Only if colvar is specified, the range of the color variable, used for the color key. Values of colvar that extend the range will be put to NA.}

\textbf{theta, phi} \quad \text{The angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude. see \texttt{persp}.}

\textbf{panel.first} \quad \text{A function to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix, e.g. it should be defined as function(pmat). See example of \texttt{persp3D} and last example of \texttt{voxel3D}.}

\textbf{bty} \quad \text{The type of the box, the default only drawing background panels. Only effective if the \texttt{persp} argument (box) equals TRUE (this is the default). See \texttt{perspbox}.}

\textbf{dDepth} \quad \text{When a contour is added on an image, the image polygons may hide some contour segments. To avoid that, the viewing depth of the segments can be artificially decreased with the factor dDepth times the \texttt{persp} argument expand (usually = 1), to make them appear in front of the polygons. Too large values of dDepth may create visible artifacts.}
addbox  If TRUE will draw a box around the plot.
add  Logical. If TRUE, then the contours will be added to the current plot. If FALSE a new plot is started.
plot  Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

... additional arguments passed to the plotting methods.
The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, sc, shade, lighting, alpha.
The arguments xlim, ylim, zlim only affect the axes. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.
In addition, the perspbox arguments col.axis, col.panel, lwd.panel, col.grid, lwd.grid can also be given a value.
The arguments lty, lwd can also be specified.
shade and lighting arguments will have no effect.
alpha can be given a value inbetween 0 and 1 to make colors transparent.
The arguments after ... must be matched exactly.

Value

Returns the viewing transformation matrix. See trans3D.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

contour for R’s 2-D contour function.

Examples

# save plotting parameters
pm <- par("mfrow")

# Contours
contour3D(x = r, y = c, z = 100, colvar = volcano, zlim = c(0, 150),
  clab = c("height", "m"))
contour3D(x = 100, y = r, z = c, colvar = volcano, clab = c("height", "m"))
contour3D(z = volcano, colvar = volcano, lwd = 2,
  nlevels = 20, clab = c("height", "m"), colkey = FALSE)
3-D data set

contour3D(y = volcano, colvar = volcano, lwd = 2,
    nlevels = 10, clab = c("height", "m"))

# Composite images and contours in 3D
persp3D(z = volcano, zlim = c(90, 300), col = "white",
    shade = 0.1, d = 2, plot = FALSE)
contour3D(z = volcano, colvar = volcano, lwd = 2, add = TRUE,
    nlevels = 20, clab = c("height", "m"), plot = FALSE,
    colkey = list(at = seq(90, 190, length.out = 5))
contour3D(z = 300, colvar = volcano, lwd = 2, col = "grey",
    add = TRUE, nlevels = 5)

# Viewing depth of contours (dDepth)
#
# too low
persp3D(z = volcano, col = "white", shade = 0.1, plot = FALSE)
contour3D(z = volcano, colvar = volcano, lwd = 2,
    add = TRUE, dDepth = 0, col = "black")

# default
persp3D(z = volcano, col = "white", shade = 0.1, plot = FALSE)
contour3D(z = volcano, colvar = volcano, lwd = 2,
    add = TRUE, dDepth = 0.1, col = "black")

# too high
persp3D(z = volcano, col = "white", shade = 0.1, plot = FALSE)
contour3D(z = volcano, colvar = volcano, lwd = 1,
    add = TRUE, dDepth = 0.5, col = "black")

# reset plotting parameters
par(mfrow = pm)


Description

Percentage Oxygen Saturation from the NODC World Ocean Atlas 2005 (WOA05).
The values are gridded in 2dg * 2dg longitude - latitude sets, and there are 33 depth intervals.

Usage

data(Oxsat)
3-D data set

Format

list with

- 1on, the longitude (dg E), at 2 dg resolution, 180 values.
- 1lat, the latitude (dg N), at 2 dg resolution, 90 values.
- depth, the water depth (m), 33 values.
- val, the saturation value (%). val is an array of dimension (180, 90, 33), (lon, lat, depth).
- name, the long name of the variable.
- units, the units of measurement.

Details

The “objectively analyzed climatology” has been used to extract these data.

The original data were averaged over the 4 seasons, and converted to half the resolution for latitude and longitude. The longitude was converted to the European view, i.e. the original data from (0, 360) was changed to (-180, 180).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References

http://www.nodc.noaa.gov

Originally made available by CSIRO:

See Also

image2D for plotting.

Examples

# save plotting parameters
pm <- par("mfrow")

### ========================================================================
### plot all surface data
### ========================================================================
par(mfrow = c(1, 1))
image2D(z = Oxsat$val[, , 1], x = Oxsat$lon, y = Oxsat$lat,
       main = "surface oxygen saturation (%) for 2005")

### ========================================================================
### plot a selection of latitude-depth profiles; input is an array
3-D perspectives

```
## ========================================================================
lon <- Oxsat$lon
image2D (z = Oxsat$val, margin = c(2, 3), x = Oxsat$lat,
        y = Oxsat$depth, subset = (lon > 18 & lon < 23),
        ylim = c(5500, 0), NAcol = "black", zlim = c(0, 110),
        xlab = "latitude", ylab = "depth, m")

ImageOcean()
abline ( v = lon[lon > 18 & lon < 23])

## ========================================================================
## plot with slices
## ========================================================================
par(mfrow = c(1, 1))
ii <- which (Oxsat$lon > -90 & Oxsat$lon < 90)
jj <- which (Oxsat$lat > 0 & Oxsat$lat < 90)

xs <- Oxsat$lon[ii[length(ii)]] # E boundary
ys <- Oxsat$lat[jj[1]] # S boundary

slice3D(colvar = Oxsat$val[ii,jj,], x = Oxsat$lon[ii],
        y = Oxsat$lat[jj], z = -Oxsat$depth,
        NAcol = "black", xs = xs, ys = ys, zs = 0,
        theta = 35, phi = 50, colkey = list(length = 0.5),
        expand = 0.5, ticktype = "detailed",
        clab = "\%", main = "Oxygen saturation",
        xlab = "longitude", ylab = "latitude", zlab = "depth")

# restore plotting parameters
par(mfrow = pm)
```

### Description

persp3D extends R’s persp function.

ribbon3D is similar to persp3D but has ribbon-like colored surfaces.

hist3D generates 3-D histograms.

### Usage

```
persp3D (x = seq(0, 1, length.out = nrow(z)),
         y = seq(0, 1, length.out = ncol(z)), z, ...
         colvar = z, phi = 40, theta = 40,
         col = NULL, NAcol = "white", breaks = NULL,
         border = NA, facets = TRUE, colkey = NULL, resfac = 1,
```
image = FALSE, contour = FALSE, panel.first = NULL, 
clim = NULL, clab = NULL, bty = "b", 
lighting = FALSE, shade = NA, ltheta = -135, lphi = 0, 
inttype = 1, curtain = FALSE, add = FALSE, plot = TRUE)

ribbon3D (x = seq(0, 1, length.out = nrow(z)),
y = seq(0, 1, length.out = ncol(z)), z, 
colvar = z, phi = 40, theta = 40, 
col = NULL, NAcol = "white", breaks = NULL, 
border = NA, facets = TRUE, colkey = NULL, resfac = 1, 
image = FALSE, contour = FALSE, panel.first = NULL, 
clim = NULL, clab = NULL, bty = "b", 
lighting = FALSE, shade = NA, ltheta = -135, lphi = 0, 
space = 0.4, along = "x", 
curtain = FALSE, add = FALSE, plot = TRUE)

hist3D (x = seq(0, 1, length.out = nrow(z)),
y = seq(0, 1, length.out = ncol(z)), z, 
colvar = z, phi = 40, theta = 40, 
col = NULL, NAcol = "white", breaks = NULL, 
border = NA, facets = TRUE, colkey = NULL, 
image = FALSE, contour = FALSE, panel.first = NULL, 
clim = NULL, clab = NULL, bty = "b", 
lighting = FALSE, shade = NA, ltheta = -135, lphi = 0, 
space = 0, opaque.top = FALSE, zmin = NULL, 
add = FALSE, plot = TRUE)

**Arguments**

**z**  
Matrix (2-D) containing the values to be plotted as a persp plot.

**x, y**  
Vectors or matrices with x and y values. If a vector, x should be of length equal to nrow(z) and y should be equal to ncol(z). If a matrix (only for persp3D), x and y should have the same dimension as z.

**colvar**  
The variable used for coloring. If present, it should have the same dimension as z. Values of NULL, NA, or FALSE will toggle off coloration according to colvar. This gives good results only if border is given a color, or when shade is > 0 or lighting is TRUE).

**col**  
Color palette to be used for the colvar variable. If col is NULL and colvar is specified, then a red-yellow-blue colorscheme (jet.col) will be used. If col is NULL and colvar is not specified, then col will be grey. Finally, to mimic the behavior of persp, set colvar = NULL and make col a matrix of colors with (nrow(z)-1) rows and (ncol(z)-1) columns.

**NAcol**  
Color to be used for NA values of colvar; default is "white".

**breaks**  
a set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.
colkey A logical, NULL (default), or a list with parameters for the color key (legend). List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col.axis.
The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL) See colkey. The default is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. if colkey = FALSE, no color key legend will be added.

clab Only if colkey = TRUE, the label to be written on top of the color key. The label will be written at the same level as the main title. to lower it, clab can be made a vector, with the first values empty strings.

clim Only if colvar is specified, the range of the color variable, used for the color key. Values of colvar that extend the range will be put to NA.

resfac Resolution factor, one value or a vector of two numbers, for the x and y-values respectively. A value > 1 will increase the resolution. For instance, if resfac equals 3 then for each adjacent pair of x- and y-values, z will be interpolated to two intermediary points. This uses simple linear interpolation. If resfac is one number then the resolution will be increased similarly in x and y-direction.

theta, phi The angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude. see persp.

border The color of the lines drawn around the surface facets. The default, NA, will disable the drawing of borders.

facets If TRUE, then col denotes the color of the surface facets. If FALSE, then the surface facets are colored “white” and the border (if NA) will be colored as specified by col. If NA then the facets will be transparent. It is usually faster to draw with facets = FALSE.

image If TRUE, an image will be plotted at the bottom. Also allowed is to pass a list with arguments for the image2D function. An optional parameter to this list is the side where the image should be plotted. Allowed values for side are a z-value, or side = "zmin", "zmax", for positioning at bottom or top respectively. The default is to put the image at the bottom.

contour If TRUE, a contour will be plotted at the bottom. Also allowed is to pass a list with arguments for the contour function. An optional parameter to this list is the side where the image should be plotted. Allowed values for side are a z-value, or side = "zmin", "zmax", for positioning at bottom or top respectively. The default is to put the image at the bottom.

panel.first A function to be evaluated after the plot axes are set up (and if applicable, images or contours drawn) but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix (pmat), e.g. it should be defined as function(pmat). See example.

along The direction along which the ribbons are drawn, one of "x", "y" or "xy", for ribbons parallel to the x- y- or both axes. In the latter case, the figure looks like a net.
curtain

If TRUE, the ribbon or persp edges will be draped till the bottom.

space

The amount of space (as a fraction of the average bar/ribbon width) left between bars/ribbons. A value in between [0, 0.9] (hist3D) or [0.1, 0.9] (ribbon3D). Either one number, or a two-valued vector, for the x- and y- direction.

bty

The type of the box, the default only drawing background panels. Only effective if the persp argument (box) equals TRUE (this is the default). See perspbox.

lighting

If not FALSE the facets will be illuminated, and colors may appear more bright. To switch on lighting, the argument lighting should be either set to TRUE (using default settings) or it can be a list with specifications of one of the following: ambient, diffuse, specular, exponent, sr and alpha.

Will overrule shade not equal to NA.

See examples in jet.col.

shade

the degree of shading of the surface facets. Values of shade close to one yield shading similar to a point light source model and values close to zero produce no shading. Values in the range 0.5 to 0.75 provide an approximation to daylight illumination. See persp.

ltheta, lphi

if finite values are specified for ltheta and lphi, the surface is shaded as though it was being illuminated from the direction specified by azimuth ltheta and colatitude lphi. See persp.

inttype

The interpolation type to create the polygons, either averaging the colvar (inttype = 1, 3 or extending the x, y, z values (inttype = 2) - see details.

opaque.top

Only used when alpha is set (transparency): if TRUE then the top of the bars is opaque.

zmin

The base of the histogram; if NULL then it extends to the minimum of the z-axis. Note: this was added from version 1.1.1 on; before that it was assumed that the base of the histogram was at z=0.

add

Logical. If TRUE, then the surfaces will be added to the current plot. If FALSE a new plot is started.

plot

Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

... additional arguments passed to the plotting methods. The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, scale, expand, box, axes, The arguments xlim, ylim, zlim only affect the axes. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.

In addition, the perspbox arguments col.axis, col.panel, lwd.panel, col.grid, lwd.grid can also be given a value.

alpha can be given a value inbetween 0 and 1 to make colors transparent.

For all functions, the arguments lty, lwd can be specified; this is only effective if border is not NA.

The arguments after ... must be matched exactly.
Details

persp3D is an extension to the default `persp` plot that has the possibility to add a color key, to increase the resolution in order to make smoother images, to toggle on or off facet coloration, ...

The perspective plots are drawn as filled polygons. Each polygon is defined by 4 corners and a color, defined in its centre. When facets are colored, there are three interpolation schemes as set by `inttype`.

The default (`inttype = 1`) is similar to R’s function `persp`, and assumes that the z-values define the points on the corners of each polygon. In case a `colvar` is defined, its values are to be recalculated to the middle of each polygon, i.e. the color values need to be of size (nx-1)(ny-1), and averages are taken from the original data (nx and ny are number of x and y points). This will make the colors (and/or shading) smoother. When `inttype = 1` then NA values in `colvar` will be used as such during the averaging. This will tend to make the NA region larger.

An alternative is to set `inttype = 3`, which is similar to `inttype = 1` except for the NA values, which will be removed during the averaging. This will tend to make the NA region smaller.

By setting `inttype = 2`, a second interpolation scheme is selected. This is mainly of use in case a `colvar` is defined, and it is not desirable that the colors are smoothened. In this scheme, it is assumed that the z values and `colvar` values are both defined in the centre of the polygons. To color the facets the x, y, z grid is extended (to a (nx+1)(ny+1) grid), while `colvar` is used as such. This will make the z-values (topography) smoother than the original data. This type of interpolation may be preferable for color variables that have NA values, as taking averages tends to increase the NA region.

Value

Returns, as invisible, the viewing transformation matrix.

See `trans3D`.

Note

To make a contour to appear on top of an image, i.e. when `side = "z"`, the viewing depth of the contour segments is artificially decreased. In some cases this may produce slight artifacts. The viewing depth can be adjusted with argument `dDepth`, e.g. `persp3D(z = volcano, contour = list(side = "z", dDepth = 0.))`

Parts of this help page come from the help pages of the R-core function `persp`.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References

The `persp` function on which this implementation is based:

See Also

`persp` for the function on which this is based.

Hypsometry for an example where axis-panels are colored.

`scatter3D` for a combination of a persp surface and data points.

`text3D` for annotating axes (hist3D).

`plotdev` for zooming, rescaling, rotating a plot.

Examples

```r
# save plotting parameters
pm <- par("mfrow")

## Ribbon, persp, color keys, facets

par(mfrow = c(2, 2))
# simple, no scaling, use breaks to set colors
persp3D(z = volcano, main = "volcano", clab = c("height", "m"),
breaks = seq(80, 200, by = 10))

# keep ratios between x- and y (scale = FALSE)
# change ratio between x- and z (expand)
persp3D(z = volcano, x = 1: nrow(volcano), y = 1:ncol(volcano),
expand = 0.3, main = "volcano", facets = FALSE, scale = FALSE,
clab = "height, m", colkey = list(side = 1, length = 0.5))

# ribbon, in x--direction
V <- volcano[, seq(1, ncol(volcano), by = 3)]  # lower resolution
ribbon3D(z = V, colkey = list(width = 0.5, length = 0.5,
cex.axis = 0.8, side = 2), clab = "m")

# ribbon, in y-direction
Vy <- volcano[seq(1, nrow(volcano), by = 3), ]
ribbon3D(z = Vy, expand = 0.3, space = 0.3, along = "y",
colkey = list(width = 0.5, length = 0.5, cex.axis = 0.8))

## Several ways to visualise 3-D data

x <- seq(-pi, pi, by = 0.2)
y <- seq(-pi, pi, by = 0.3)
grid <- mesh(x, y)

z <- with(grid, cos(x) * sin(y))
par(mfrow = c(2, 2))
persp3D(z = z, x = x, y = y)
```
persp3D(z = z, x = x, y = y, facets = FALSE, curtain = TRUE)

# ribbons in two directions and larger spaces
ribbon3D(z = z, x = x, y = y, along = "xy", space = 0.3)

hist3D(z = z, x = x, y = y, border = "black")

## =======================================================================
## Contours and images added
## =======================================================================

par(mfrow = c(2, 2))
x <- seq(1, nrow(volcano), by = 3)
y <- seq(1, ncol(volcano), by = 3)

Volcano <- volcano [x, y]
ribbon3D(z = Volcano, contour = TRUE, zlim= c(-100, 200),
         image = TRUE)
persp3D(z = Volcano, contour = TRUE, zlim= c(-200, 200), image = FALSE)
persp3D(z = Volcano, x = x, y = y, scale = FALSE,
        contour = list(nlevels = 20, col = "red"),
        zlim = c(-200, 200), expand = 0.2,
        image = list(col = grey (seq(0, 1, length.out = 100))))
persp3D(z = Volcano, contour = list(side = c("zmin", "z", "350")),
        zlim = c(-100, 400), phi = 20, image = list(side = 350))

## =======================================================================
## Use of inttype
## =======================================================================

par(mfrow = c(2, 2))
persp3D(z = Volcano, shade = 0.5, colkey = FALSE)
persp3D(z = Volcano, inttype = 2, shade = 0.5, colkey = FALSE)
x <- y <- seq(0, 2*pi, length.out = 10)
z <- with (mesh(x, y), cos(x) *sin(y)) + runif(100)
cv <- matrix(nrow = 10, 0.5*runif(100))
persp3D(x, y, z, colvar = cv) # takes averages of z
persp3D(x, y, z, colvar = cv, inttype = 2) # takes averages of colvar

## =======================================================================
## Use of inttype with NAs
## =======================================================================

par(mfrow = c(2, 2))
VV <- V2 <- volcano[10:15, 10:15]
V2[4, 5] <- NA
image2D(V2, border = "black") # shows true NA region

# averages of V2, including NAs, NA region larger
persp3D(z = VV, colvar = V2, inttype = 1, theta = 0,
       phi = 20, border = "black", main = "inttype = 1")

# extension of VV; NAs unaffected
persp3D(z = VV, colvar = V2, inttype = 2, theta = 0,
       phi = 20, border = "black", main = "inttype = 2")

# average of V2, ignoring NA; NA region smaller
persp3D(z = VV, colvar = V2, inttype = 3, theta = 0,
       phi = 20, border = "black", main = "inttype = 3")

## =======================================================================
## Use of panel.first
## =======================================================================

par(mfrow = c(1, 1))

# A function that is called after the axes were drawn
panelfirst <- function(trans) {
  zticks <- seq(100, 180, by = 20)
  len <- length(zticks)
  XY0 <- trans3D(x = rep(1, len), y = rep(1, len), z = zticks,
                  pmat = trans)
  XY1 <- trans3D(x = rep(1, len), y = rep(61, len), z = zticks,
                  pmat = trans)
  segments(XY0$x, XY0$y, XY1$x, XY1$y, lty = 2)
  rm <- rowMeans(volcano)
  XY <- trans3D(x = 1:87, y = rep(ncol(volcano), 87),
                z = rm, pmat = trans)
  lines(XY, col = "blue", lwd = 2)
}
persp3D(z = volcano, x = 1:87, y = 1:61, scale = FALSE, theta = 10,
       expand = 0.2, panel.first = panelfirst, colkey = FALSE)

## =======================================================================
## with / without colvar / facets
## =======================================================================

par(mfrow = c(2, 2))
persp3D(z = volcano, shade = 0.3, col = gg.col(100))

# shiny colors - set lphi for more brightness
persp3D(z = volcano, lighting = TRUE, lphi = 90)

persp3D(z = volcano, col = "lightblue", colvar = NULL,
       shade = 0.3, bty = "b2")

# this also works:
# persp3D(z = volcano, col = "grey", shade = 0.3)
# tilted x- and y-coordinates of 'volcano'
volcx <- matrix(nrow = 87, ncol = 61, data = 1:87)
volcx <- volcx + matrix(nrow = 87, ncol = 61,
    byrow = TRUE, data = seq(0., 15, length.out = 61))

volcy <- matrix(ncol = 87, nrow = 61, data = 1:61)
volcy <- t(volcy + matrix(ncol = 87, nrow = 61,
    byrow = TRUE, data = seq(0., 15, length.out = 87)))

persp3D(volcano, x = volcx, y = volcy, phi = 80)

## =======================================================================
## Several persps on one plot
## =======================================================================

par(mfrow = c(1, 1))
clim <- range(volcano)
persp3D(z = volcano, zlim = c(100, 600), clim = clim,
    box = FALSE, plot = FALSE)
persp3D(z = volcano + 200, clim = clim, colvar = volcano,
    add = TRUE, colkey = FALSE, plot = FALSE)
persp3D(z = volcano + 400, clim = clim, colvar = volcano,
    add = TRUE, colkey = FALSE)  # plot = TRUE by default

## =======================================================================
## hist3D
## =======================================================================

par(mfrow = c(2, 2))
VV <- volcano[seq(1, 87, 15), seq(1, 61, 15)]
hist3D(z = VV, scale = FALSE, expand = 0.01, border = "black")

# transparent colors
hist3D(z = VV, scale = FALSE, expand = 0.01,
    alpha = 0.5, opaque.top = TRUE, border = "black")
hist3D(z = VV, scale = FALSE, expand = 0.01, facets = FALSE, lwd = 2)
hist3D(z = VV, scale = FALSE, expand = 0.01, facets = NA)

## =======================================================================
## hist3D and ribbon3D with greyish background, rotated, rescaled,...
## =======================================================================

par(mfrow = c(2, 2))
hist3D(z = VV, scale = FALSE, expand = 0.01, bty = "g", phi = 20,
    col = "#0072B2", border = "black", shade = 0.2, ltheta = 90,
    space = 0.3, ticktype = "detailed", d = 2)

# extending the ranges
3-D surfaces

Functions for plotting 3 dimensional shapes

Description
surf3D plots a surface in 3-D with a color variable.

spheresurf3D plots a colored image on a sphere.

Usage

```
surf3D(x, y, z, ..., colvar = z, phi = 40, theta = 40, 
col = NULL, NAcol = "white", breaks = NULL, 
border = NA, facets = TRUE, colkey = NULL, 
panel.first = NULL, clim = NULL, clab = NULL, bty = "n", 
lighting = FALSE, shade = NA, ltheta = -135, lphi = 0, 
inttype = 1, add = FALSE, plot = TRUE)
```

```
spheresurf3D(colvar = matrix(nrow = 50, ncol = 50, data = 1:50, byrow = TRUE), 
..., phi = 0, theta = 0, 
col = NULL, NAcol = "white", breaks = NULL, 
border = NA, facets = TRUE, contour = FALSE, 
colkey = NULL, resfac = 1)
```
Arguments

\(x, y, z\)  
Matrices with \(x, y\) and \(z\)-values that define the surfaces to be colored. They should be of the same dimension as \(colvar\).

\(colvar\)  
The variable used for coloring. If a matrix, it should be of the same dimension as \(x, y, z\). Values of \(NULL\), \(NA\), or \(FALSE\) will toggle off coloration according to \(colvar\). This gives good results only if \(border\) is given a color or a shade is used.

\(\text{theta, phi}\)  
the angles defining the viewing direction. \(\text{theta}\) gives the azimuthal direction and \(\text{phi}\) the colatitude. see \text{persp}.

\(\text{col}\)  
Color palette to be used for coloring the \(colvar\) variable. If \(\text{col}\) is \(NULL\) and \(colvar\) is specified, then a red-yellow-blue colorscheme (\text{jet.col}) will be used. If \(\text{col}\) is \(NULL\) and \(colvar\) is not specified, then \(\text{col}\) will be "grey".

\(\text{NAcol}\)  
Colors to be used for \(colvar\) values that are \(NA\).

\(\text{breaks}\)  
a set of finite numeric breakpoints for the colors; must have one more breakpoint than \(\text{color}\) and be in increasing order. Unsorted vectors will be sorted, with a warning.

\(\text{border}\)  
The color of the lines drawn around the surface facets. The default, \(NA\), will disable the drawing of borders.

\(\text{facets}\)  
If \(TRUE\), then \(\text{col}\) denotes the color of the surface facets. If \(FALSE\), then the surface facets are colored "white" and the \(\text{border}\) (if \(\text{NA}\)) will be colored as specified by \(\text{col}\). If \(\text{NA}\) then the facets will be transparent. It is usually faster to draw with facets = \(FALSE\).

\(\text{contour}\)  
If \(TRUE\), then a \text{contour} plot will be added to the image plot, unless \(x, y\) are a matrix. Also allowed is to pass a list with arguments for the \text{contour} function.

\(\text{colkey}\)  
A logical, \(\text{NULL}\) (default), or a list with parameters for the color key (legend). List parameters should be one of \(\text{side, plot, length, width, dist, shift, addlines, col.clab, cex.clab}\) and the axis parameters \(\text{at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col}\). The defaults for the parameters are \(\text{side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = NULL, adj.clab = NULL, font.clab = NULL}\). See \text{colkey}.

The default is to draw the color key on side = 4, i.e. in the right margin. If \(\text{colkey = NULL}\) then a color key will be added only if \(\text{col}\) is a vector. Setting \(\text{colkey = list(plot = FALSE)}\) will create room for the color key without drawing it. If \(\text{colkey = FALSE}\), no color key legend will be added.

\(\text{resfac}\)  
Resolution factor, one value or a vector of two numbers, for the \(x\) and \(y\)-values respectively. A value > 1 will increase the resolution. For instance, if \(\text{resfac}\) equals 3 then for each adjacent pair of \(x\)- and \(y\)-values, \(z\) will be interpolated to two intermediary points. This uses simple linear interpolation. If \(\text{resfac}\) is one number then the resolution will be increased similarly in \(x\) and \(y\)-direction.
panel.first  A function to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix, e.g. it should be defined as function(pmat). See example of persp3D and last example of voxel3D.

clab  Only if colkey is not NULL or FALSE, the label to be written on top of the color key. The label will be written at the same level as the main title. To lower it, clab can be made a vector, with the first values empty strings.

clim  Only if colvar is specified, the range of the color variable, used for the color key. Values of colvar that extend the range will be put to NA.

bty  The type of the box, the default is to draw no box. Set bty = "f" or bty = "b" if you want a full box or the backpanel. See perspbox.

lighting  If not FALSE the facets will be illuminated, and colors may appear more bright. To switch on lighting, the argument lighting should be either set to TRUE (using default settings) or it can be a list with specifications of one of the following: ambient, diffuse, specular, exponent, sr and alpha. Will overrule shade not equal to NA. See examples in jet.col.

shade  the degree of shading of the surface facets. Values of shade close to one yield shading similar to a point light source model and values close to zero produce no shading. Values in the range 0.5 to 0.75 provide an approximation to daylight illumination. See persp.

ltheta, lphi  if finite values are specified for ltheta and lphi, the surface is shaded as though it was being illuminated from the direction specified by azimuth ltheta and colatitude lphi. See persp.

inttype  The interpolation type to create the polygons, either taking the mean of the colvar variable (inttype = 1,3) or extending the x,y,z values (inttype = 2). Values 1,3 differ in how they treat NAs in the colvar variable. For inttype = 3, NAs are removed before taking averages; this will tend to make the NA region smaller. NAs are included when inttype = 1. This will tend to make the NA region larger. See details and an example in persp3D.

full  Logical. If TRUE, the full sphere will be drawn, including the invisible part. If FALSE only the visible half will be drawn (faster).

add  Logical. If TRUE, then the surfaces will be added to the current plot. If FALSE a new plot is started.

plot  Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

...  Additional arguments passed to the plotting methods. The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, scale, expand, box, axes, n.ticks, ticktype. The arguments xlim, ylim, zlim only affect the axes. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.

In addition, the perspbox arguments col.axis, col.panel, lwd.panel, col.grid, lwd.grid can also be given a value. The arguments after ... must be matched exactly.
Details

Function `spheresurf3D` is a projection on a sphere with radius 1. This means that the x-, y- and z-axes range from [-1, 1].

Value

Returns the viewing transformation matrix, See `trans3D`.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

`persp` for the function on which this implementation is based.

`jet.col, plotdev` for other examples of `surf3D`.

`plotdev` for zooming, rescaling, rotating a plot.

Examples

```r
# save plotting parameters
pm <- par("mfrow")
pmar <- par("mar")

par(mar = c(1, 1, 1, 1))
## =======================================================================
## A three-dimensional shape
## (ala http://docs.enthought.com/mayavi/mayavi/mlab.html)
## =======================================================================
par(mfrow = c(2, 2))

# create grid matrices
X <- seq(0, pi, length.out = 50)
Y <- seq(0, 2*pi, length.out = 50)
M <- mesh(X, Y)
phi <- M$x
theta <- M$y

# x, y and z grids
r <- sin(4*phi)^3 + cos(2*phi)^3 + sin(6*theta)^2 + cos(6*theta)^4
x <- r * sin(phi) * cos(theta)
y <- r * cos(phi)
z <- r * sin(phi) * sin(theta)

# full colored image
surf3D(x, y, z, colvar = y, colkey = FALSE, shade = 0.5,
      box = FALSE, theta = 60)

# same, but just facets
surf3D(x, y, z, colvar = y, colkey = FALSE, box = FALSE,
       theta = 60)
```
theta = 60, facets = FALSE)

# with colors and border, AND increasing the size
# (by reducing the x- y and z- ranges
surf3D(x, y, z, colvar = y, colkey = FALSE, box = FALSE,
   theta = 60, border = "black", xlim = range(x)*0.8,
ylim = range(y)*0.8, zlim = range(z)*0.8)

# Now with one color and shading
surf3D(x, y, z, box = FALSE,
   theta = 60, col = "lightblue", shade = 0.9)

## Not run: # rotation
for (angle in seq(0, 360, by = 10))
  plotdev(theta = angle)

## End(Not run)

## =======================================================================
## Several other shapes
## http://xahlee.info/surface/gallery.html
## =======================================================================

par(mfrow = c(2, 2))
# Shape 1
M <- mesh(seq(0, 6*pi, length.out = 50),
   seq(pi/3, pi, length.out = 50))
  u <- M$x ; v <- M$y
  x <- u/2 * sin(v) * cos(u)
y <- u/2 * sin(v) * sin(u)
z <- u/2 * cos(v)
surf3D(x, y, z, colvar = z, colkey = FALSE, box = FALSE, phi = 50)

# Shape 2: add border
M <- mesh(seq(0, 2*pi, length.out = 50),
   seq(0, 2*pi, length.out = 50))
  u <- M$x ; v <- M$y
  x <- sin(u)
y <- sin(v)
z <- sin(u + v)
surf3D(x, y, z, colvar = z, border = "black",
   colkey = FALSE)

# shape 3: uses same mesh, other perspective (d >1)
x <- (3 + cos(v/2)*sin(u) - sin(v/2)*sin(2*u)) cos(v)
y <- (3 + cos(v/2)*sin(u) - sin(v/2)*sin(2*u)) sin(v)
z <- sin(v/2)*sin(u) + cos(v/2)*sin(2*u)
surf3D(x, y, z, colvar = z, colkey = FALSE, d = 2, facets = FALSE)

# shape 4: more complex colvar
M <- mesh(seq(-13.2, 13.2, length.out = 50),
  seq(-37.4, 37.4, length.out = 50))
  u <- M$x ; v <- M$y
b <- 0.4; r <- 1 - b^2; w <- sqrt(r)
D <- b*(w*cosh(b*u)^2 + (b*sin(w*v))^2)
x <- -u + (2*r*cosh(b*u)*sinh(b*u)) / D
y <- (2*w*cosh(b*u)*(-w*cos(v)*cos(w*v)) - sin(v)*sin(w*v))) / D
z <- (2*w*cosh(b*u)*(-w*sin(v)*cos(w*v)) + cos(v)*sin(w*v))) / D
surf3D(x, y, z, colvar = sqrt(x + 8.3), colkey = FALSE,
    theta = 10, border = "black", box = FALSE)

## =======================================================================
## A sphere, with box type with grid lines
## =======================================================================
par(mar = c(2, 2, 2, 2))
par(mfrow = c(1, 1))
M <- mesh(seq(0, 2*pi, length.out = 50),
  seq(0, pi, length.out = 50))
  u <- M$x ; v <- M$y
x <- cos(u)*sin(v)
y <- sin(u)*sin(v)
z <- cos(v)
colvar <- sin(u*6) * sin(v*6)
surf3D(x, y, z, colvar = colvar, phi = 0, bty = "b2",
    lighting = TRUE, ltheta = 40)

## =======================================================================
## Function spheresurf3D
## =======================================================================
par(mfrow = c(2, 2))
spheresurf3D()

# true ranges are [-1, 1]; set limits to [-0.8, 0.8] to make larger plots
lim <- c(-0.8, 0.8)
spheresurf3D(colkey = FALSE, xlim = lim, ylim = lim, zlim = lim)
spheresurf3D(bty = "b", ticktype = "detailed", phi = 50)
spheresurf3D(colvar = matrix(nrow = 30, data = runif(900)))

## =======================================================================
## Images on a sphere
## =======================================================================
3-D volume visualisation

Functions for plotting 3-D volumetric data.

Description

slice3D plots a 3-D dataset with a color variable as slices or on surfaces.
slicecont3D plots a 3-D dataset with a color variable as contours on slices.
isosurf3D plots isosurfaces from a 3-D dataset.
voxel3D plots isosurfaces as scatterpoints.
createisosurf create the isosurfaces (triangulations) from volumetric data. Its output can be plotted with triangle3D.
createvoxel creates voxels (x, y, z) points from volumetric data. Its output can be plotted with scatter3D.

Usage

```
slice3D (x, y, z, colvar, ..., phi = 40, theta = 40,
        xs = min(x), ys = max(y), zs = min(z),
        col = NULL, NAcol = "white", breaks = NULL,
        border = NA, facets = TRUE, colkey = NULL,
        panel.first = NULL, clim = NULL,
        clab = NULL, bty = "b",
        lighting = FALSE, shade = NA, ltheta = -135, lphi = 0,
        add = FALSE, plot = TRUE)
```

```
slicecont3D (x, y, z, colvar, ..., phi = 40, theta = 40,
             xs = NULL, ys = NULL, zs = NULL, level = NULL,
             col = NULL, NAcol = "white", breaks = NULL,
             border = NA, facets = TRUE,
             colkey = NULL, panel.first = NULL,
```
3-D volume visualisation

clim = NULL, clab = NULL, bty = "b",
dDepth = 0, add = FALSE, plot = TRUE)

isosurf3D (x, y, z, colvar, ..., phi = 40, theta = 40,
level = mean(colvar, na.rm = TRUE), isofunc = createisosurf,
col = NULL, border = NA, facets = TRUE,
colkey = NULL, panel.first = NULL,
clab = NULL, bty = "b",
lighting = FALSE, shade = 0.5, ltheta = -135, lphi = 0,
add = FALSE, plot = TRUE)

voxel3D (x, y, z, colvar, ..., phi = 40, theta = 40,
level = mean(colvar, na.rm = TRUE), eps = 0.01, operator = "=",
col = NULL, NAcol = "white", breaks = NULL, colkey = FALSE,
panel.first = NULL, bty = "b", add = FALSE, plot = TRUE)

triangle3D (tri, colvar = NULL, ..., phi = 40, theta = 40,
col = NULL, NAcol = "white", breaks = NULL,
border = NA, facets = TRUE,
colkey = NULL, panel.first = NULL,
lighting = FALSE, shade = 0.5, ltheta = -135, lphi = 0,
clim = NULL, clab = NULL,
bty = "b", add = FALSE, plot = TRUE)

createisosurf (x, y, z, colvar, level = mean(colvar, na.rm = TRUE))

createvoxel (x, y, z, colvar, level = mean(colvar, na.rm = TRUE), eps = 0.01,
operator = "=")

Arguments

x, y, z Vectors with x, y and z-values. They should be of length equal to the first,
second and third dimension of colvar respectively.

colvar The variable used for coloring. It should be an array of dimension equal to
c(length(x),length(y),length(z)). For triangle3D, colvar should be of
length = nrow(tri)/3. It must be present.

tri A three-columned matrix (x, y, z) with triangle coordinates. A triangle is defined
by three consecutive rows.

isofunc A function defined as function(x,y,z,colvar,level), and that returns the
three-columned matrix with triangle coordinates. The default, createisosurf
uses function computeContour3d from package misc3d.

theta, phi the angles defining the viewing direction. theta gives the azimuthal direction
and phi the colatitude. see persp.

col Colors to be used for coloring the colvar variable. If col is NULL then a red-
yellow-blue colorscheme (jet.col) will be used.

NAcol Colors to be used for colvar values that are NA.
a set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.

The color of the lines drawn around the surface facets. The default, NA, will disable the drawing of borders.

If TRUE, then col denotes the color of the surface facets. If FALSE, then the surface facets are colored “white” and the border (if NA) will be colored as specified by col. If NA then the facets will be transparent. It is usually faster to draw with facets = FALSE.

A logical, NULL (default), or a list with parameters for the color key (legend). List parameters should be one of side, plot, length, width, dist, shift, addlines, col.hlab, cex.hlab and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col. The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.hlab = NULL, cex.hlab = par("cex.lab"), side.hlab = NULL, line.hlab = NULL, adj.hlab = NULL, font.hlab = NULL. See colkey.

The default is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. If colkey = FALSE, no color key legend will be added.

A function to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix, e.g. it should be defined as function(pmat). See last example and example of persp3D.

Only if colkey is not NULL or FALSE, the label to be written on top of the color key. The label will be written at the same level as the main title. To lower it, clab can be made a vector, with the first values empty strings.

Only if colvar is specified, the range of the color variable, used for the color key. Values of colvar that extend the range will be put to NA.

Vectors or matrices. Vectors specify the positions in x, y or z where the slices (planes) are to be drawn. The values of colvar will be projected on these slices. Matrices specify a surface on which the colvar will be projected.

The level(s) at which the contour will be generated or the isosurfaces generated. There can be more than one level, but for slicecont3D too many will give a crowded view, and one is often best. For isosurf3D, the use of multiple values may need transparent colors to visualise. For voxel3D, level should either be one number (if operator equals '=' or '<>' or '>=' ) or two numbers (for operator = '<>').

If not FALSE the facets will be illuminated, and colors may appear more bright. To switch on lighting, the argument lighting should be either set to TRUE (using default settings) or it can be a list with specifications of one of the following: ambient, diffuse, specular, exponent, sr and alpha. Will overrule shade not equal to NA. See examples in jet.col.
shade

the degree of shading of the surface facets. Values of shade close to one yield shading similar to a point light source model and values close to zero produce no shading. Values in the range 0.5 to 0.75 provide an approximation to daylight illumination. See persp.

ltheta, lphi

if finite values are specified for ltheta and lphi, the surface is shaded as though it was being illuminated from the direction specified by azimuth ltheta and colatitude lphi. See persp.

bty

The type of the box, the default only draws background panels. Only effective if the persp argument (box) equals TRUE (this is the default). See perspbox.

eps

The voxel precision, only used when operator = "=". A point is selected if it closer than eps*diff(range(colvar)) to the required level.

operator

One of ",=','<','>','<>' for selection of points 'equal' (within precision), larger or smaller than the required level or to be within an interval.

dDepth

When a contour is added on an image, the image polygons may hide some contour segments. To avoid that, the viewing depth of the segments can be artificially decreased with the factor dDepth times the persp argument expand (usually = 1), to make them appear in front of the polygons. Too large values of dDepth may create visible artifacts. See contour3D.

add

Logical. If TRUE, then the slices, voxels or surfaces will be added to the current plot. If FALSE a new plot is started.

plot

Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

additional arguments passed to the plotting methods.

The following persp arguments can be specified: xlim,ylim,zlim,xlab,ylab,zlab,main,sub,r,d,scale,expand,box,axes,nticks,ticktype.

The arguments xlim,ylim,zlim only affect the axes. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.

In addition, the perspbox arguments col.axis,col.panel,lwd.panel,col.grid,lwd.grid can also be given a value.

alpha can be given a value inbetween 0 and 1 to make colors transparent.

For all functions, the arguments lty,lwd can be specified.

The arguments after ... must be matched exactly.

Value

The plotting functions return the viewing transformation matrix. See trans3D.

Function createisosurf returns a three-columned matrix (x, y, z) with triangle coordinates. One triangle is defined by three consecutive rows. It can be plotted with triangle3D.

Function createvoxel returns a list with the elements x, y, z defining the points that are at a distance of less than eps*diff(range(colvar)) from the required level. Its output can be plotted with scatter3D.
Note

The isosurf3D function uses function computeContour3d, from package misc3d, which is based on the marching cubes algorithm. Please cite the package misc3d (Feng & Tierney, 2008) when using isosurf3D.

For voxel3D, coloring is always according to the z-variable. A more flexible coloration can be achieved by using createvoxel, followed by scatter3D. See examples.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

References


See Also

Oxsat for another example of slice3D.

plotdev for zooming, rescaling, rotating a plot.

Examples

```r
# save plotting parameters
pm <- par("mfrow")
pmar <- par("mar")

# Simple slice3D examples
# ------------------------------

par(mfrow = c(2, 2))
x <- y <- z <- seq(-1, 1, by = 0.1)
grid <- mesh(x, y, z)
colvar <- with(grid, x*exp(-x^2 - y^2 - z^2))

# default is just the panels
slice3D (x, y, z, colvar = colvar, theta = 60)

# contour slices
slicecont3D (x, y, z, ys = seq(-1, 1, by = 0.5), colvar = colvar,
theta = 60, border = "black")

slice3D (x, y, z, xs = c(-1, -0.5, 0.5), ys = c(-1, 0, 1),
zs = c(-1, 0), colvar = colvar,
theta = 60, phi = 40)
```

```
3-D volume visualisation

## coloring on a surface

```r
XY <- mesh(x, y)
ZZ <- XY$x*XY$y
slice3D (x, y, z, xs = XY$x, ys = XY$y, zs = ZZ, colvar = colvar,
    lighting = TRUE, lphi = 90, ltheta = 0)
```

## Specifying transparent colors

```r
par(mfrow = c(1, 1))
x <- y <- z <- seq(-4, 4, by = 0.2)
M <- mesh(x, y, z)
R <- with (M, sqrt(x^2 + y^2 + z^2))
p <- sin(2*R) /(R+1e-3)

## Not run:
# This is very slow - alpha = 0.5 makes it transparent
slice3D(x, y, z, colvar = p, col = jet.col(alpha = 0.5),
    xs = 0, ys = c(-4, 0, 4), zs = NULL, d = 2)
## End(Not run)
```

## A section along a transect

```r
data(Oxsat)
Ox <- Oxsat$val[, Oxsat$lat > - 5 & Oxsat$lat < 5, ]
slice3D(x = Oxsat$lon, z = -Oxsat$depth, y = 1:5, colvar = Ox,
    ys = 1:5, zs = NULL, NAcol = "black",
    expand = 0.4, theta = 45, phi = 45)
```

## isosurf3D example - rather slow

```r
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
x <- y <- z <- seq(-2, 2, length.out = 15)
xyz <- mesh(x, y, z)
F <- with(xyz, log(x^2 + y^2 + z^2 +
    10*(x^2 + y^2) * (y^2 + z^2)^2))

# use shading for level = 1 - show triangulation with border
isosurf3D(x, y, z, F, level = 1, shade = 0.9,
    col = "yellow", border = "orange")
```
3-D volume visualisation

# lighting for level - 2
isosurf3D(x, y, z, F, level = 2, lighting = TRUE,
    lphi = 0, ltheta = 0, col = "blue", shade = NA)

# three levels, transparency added
isosurf3D(x, y, z, F, level = seq(0, 4, by = 2),
    col = c("red", "blue", "yellow"),
    clab = "F", alpha = 0.2, theta = 0, lighting = TRUE)

# transparency can also be added afterwards with plotdev()
## Not run:
    isosurf3D(x, y, z, F, level = seq(0, 4, by = 2),
        col = c("red", "blue", "yellow"),
        shade = NA, plot = FALSE, clab = "F")
    plotdev(lighting = TRUE, alpha = 0.2, theta = 0)
## End(Not run)

# use of creatisosurf
iso <- createisosurf(x, y, z, F, level = 2)
head(iso)

## Not run:
    triangle3D(iso, col = "green", shade = 0.3)
## End(Not run)

# higher resolution
x <- y <- z <- seq(-2, 2, length.out = 50)
xyz <- mesh(x, y, z)
F <- with(xyz, log(x^2 + y^2 + z^2 +
          10*(x^2 + y^2) * (y^2 + z^2)^2))

# three levels
isosurf3D(x, y, z, F, level = seq(0, 4, by = 2),
    col = c("red", "blue", "yellow"),
    shade = NA, plot = FALSE, clab = "F")
    plotdev(lighting = TRUE, alpha = 0.2, theta = 0)

## End(Not run)

## voxel3D example

# fast but needs high resolution grid
x <- y <- z <- seq(-2, 2, length.out = 70)
xyz <- mesh(x, y, z)
F <- with(xyz, log(x^2 + y^2 + z^2 +
          10*(x^2 + y^2) * (y^2 + z^2)^2))
voxel3D(x, y, z, F, level = 4, pch = ".", cex = 5)

## End(Not run)
## rotation
## =======================================================================
plotdev(theta = 45, phi = 0)
plotdev(theta = 90, phi = 10)

# same using createvoxel - more flexible for coloring
vox <- createvoxel(x, y, z, F, level = 4)
scatter3D(vox$x, vox$y, vox$z, colvar = vox$y,
      bty = "g", colkey = FALSE)

## =======================================================================
## voxel3D to show hypox sites
## =======================================================================
par(mfrow = c(1, 1), mar = c(2, 2, 2, 2))
Hypox <- createvoxel(Oxsat$lon, Oxsat$lat, Oxsat$depth[1:19],
      Oxsat$val[,1:19], level = 40, operator = "<")

panel <- function(pmat) { # an image at the bottom
  Nx <- length(Oxsat$lon)
  Ny <- length(Oxsat$lat)
  M <- mesh(Oxsat$lon, Oxsat$lat)
  xy <- trans3D(pmat = pmat, x = as.vector(M$x), y = as.vector(M$y),
      z = rep(-1000, length.out = Nx*Ny))
  x <- matrix(nrow = Nx, ncol = Ny, data = xy$x)
  y <- matrix(nrow = Nx, ncol = Ny, data = xy$y)
  Bat <- Oxsat$val[,1]; Bat[!is.na(Bat)] <- 1
  image2D(x = x, y = y, z = Bat, NAcol = "black", col = "grey",
      add = TRUE, colkey = FALSE)
}
scatter3D(Hypox$x, Hypox$y, -Hypox$z, colvar = Hypox$cv,
      panel.first = panel, pch = ".", bty = "b",
      theta = 30, phi = 20, ticktype = "detailed",
      zlim = c(-1000,0), x1im = range(Oxsat$lon),
      ylim = range(Oxsat$lat) )

# restore plotting parameters
par(mfrow = pm)
par(mar = pmar)

---

**Color key legend**

Plots a color legend

**Description**

colkey plots a color legend, either to an existing plot or starting a new plot.
Usage

```r
colkey (col = NULL, clim, clab = NULL, clog = FALSE, add = FALSE, 
cex.clab = NULL, col.clab = NULL, side.clab = NULL, 
line.clab = NULL, adj.clab = NULL, font.clab = NULL, 
side = 4, length = 1, width = 1, dist = 0, shift = 0, 
addlines = FALSE, breaks = NULL, at = NULL, labels = TRUE, tick = TRUE, 
line = NA, pos = NA, outer = FALSE, font = NA, lty = 1, lwd = 1, 
lwd.ticks = 1, col.axis = NULL, col.ticks = NULL, col.box = NULL, 
hadj = NA, padj = NA, cex.axis = par("cex.axis"), 
mgp = NULL, tck = NULL, tcl = NULL, las = NULL)
```

Arguments

- **col**: Colors to be used for the color key. If `col` is `NULL`, then a red-yellow-blue colorscheme (`jet.col`) will be used.
- **clim**: The range of the color values, used in the color key.
- **clab**: The label to be written on top of the color key. The label will be written at the same level as the main title. To lower it, either `clab` can be made a vector, with the first values empty strings. Alternatively, it can be lowered by argument `line.clab`.
- **clog**: If `TRUE`, then values of the color key will be log transformed.
- **add**: If `TRUE`, the color key will be added to the current plot and positioned in the margin. If `FALSE` a new plot will be started and the color key will be positioned in the centre.
- **cex.clab**: The size of the label written on top of the color key; default = same as axis labels.
- **col.clab**: The color of the label written on top of the color key; default = same as main title.
- **side.clab**: The side of the label written on top of the color key; default = same as main title, i.e. `side = 3`. Values of 1, 2, 4 will put the colorkey label at bottom, left and right of the key respectively.
- **line.clab**: The number of lines in the margin where the colorkey title is to be drawn. If unspecified, it is at `line.clab = 1.75`.
- **adj.clab**: The adjustment of the colorkey title, a number inbetween 0 (left) to 1 (right). The default is to put it central.
- **font.clab**: The font of the colorkey title, a number inbetween 0 (left) to 1 (right). The default is to put it central.
- **side**: Where to put the color key. 1 = bottom, 2 = left, 3 = top, 4 = right.
- **length**: Relative length of the color key; 1 = same length as the axis.
- **width**: Relative width of the color key.
- **dist**: Distance of the color key to the margin. Positive values are further into the margin, negative values cause the color key to be positioned closer to or within the main plot. Reasonable range is [-0.5, 0.05].
**Color key legend**

**shift**  
Shift relative to the centre. Positive values are upward when side = 2 or 4, and to the right for side = 1 or 3. It does not make sense to use this argument if length = 1. Reasonable range is [-0.2, 0.2].

**addlines**  
If TRUE, will draw lines inbetween the colors.

**breaks**  
a set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.

**at, labels, tick, line, pos, outer, font, lty, lwd**  
Additional parameters as from the `axis` command.

**lwd.ticks, hadj, padj, cex.axis, mgp, tck, tcl, las**  
Additional parameters as from the `axis` command.

**col.box, col.axis, col.ticks**  
Additional parameters to set the color of the color legend framing box, the axis label and the axis ticks.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**Examples**

```r
# save plotting parameters
pm <- par(mfrow = c(2, 2))
pmar <- par(mar = c(5.1, 4.1, 4.1, 2.1))

## =======================================================================
## colorkey as argument of a plot3D function
## =======================================================================
# default, colkey = NULL: adds colkey because multiple colors
image2D(z = volcano)

# default, colkey = NULL: no colkey because only one color
image2D(z = volcano, col = "grey", shade = 0.2, contour = TRUE)

# colkey = FALSE: no color key, no extra space foreseen
image2D(z = volcano, colkey = FALSE)

# colkey = list(plot = FALSE): no color key, extra space foreseen
image2D(z = volcano, colkey = list(plot = FALSE, side = 3))
colkey(side = 3, add = TRUE, clim = range(volcano))

## =======================================================================
## colorkey in new plot
## =======================================================================
colkey(side = 1, clim = c(0, 1), add = FALSE, clab = "z",
col.clab = "red", adj.clab = 0)
colkey(side = 2, clim = c(0, 1), clab = "z", length = 0.5, width = 0.5)
colkey(side = 3, clim = c(0, 1), lwd = 3, clab = c("a","b","c","d"),
```

---

**Color key legend**

- **shift**: Shift relative to the centre. Positive values are upward when side = 2 or 4, and to the right for side = 1 or 3. It does not make sense to use this argument if length = 1. Reasonable range is [-0.2, 0.2].
- **addlines**: If TRUE, will draw lines inbetween the colors.
- **breaks**: A set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.
- **at, labels, tick, line, pos, outer, font, lty, lwd**: Additional parameters as from the `axis` command.
- **lwd.ticks, hadj, padj, cex.axis, mgp, tck, tcl, las**: Additional parameters as from the `axis` command.
- **col.box, col.axis, col.ticks**: Additional parameters to set the color of the color legend framing box, the axis label and the axis ticks.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**Examples**

```r
# save plotting parameters
pm <- par(mfrow = c(2, 2))
pmar <- par(mar = c(5.1, 4.1, 4.1, 2.1))

## =======================================================================
## colorkey as argument of a plot3D function
## =======================================================================
# default, colkey = NULL: adds colkey because multiple colors
image2D(z = volcano)

# default, colkey = NULL: no colkey because only one color
image2D(z = volcano, col = "grey", shade = 0.2, contour = TRUE)

# colkey = FALSE: no color key, no extra space foreseen
image2D(z = volcano, colkey = FALSE)

# colkey = list(plot = FALSE): no color key, extra space foreseen
image2D(z = volcano, colkey = list(plot = FALSE, side = 3))
colkey(side = 3, add = TRUE, clim = range(volcano))

## =======================================================================
## colorkey in new plot
## =======================================================================
colkey(side = 1, clim = c(0, 1), add = FALSE, clab = "z",
col.clab = "red", adj.clab = 0)
colkey(side = 2, clim = c(0, 1), clab = "z", length = 0.5, width = 0.5)
colkey(side = 3, clim = c(0, 1), lwd = 3, clab = c("a","b","c","d"),
```
Color key legend

```r
line.clab = 5
colkey(side = 4, clim = c(1e-6, 1), clog = TRUE,
       lab = "a very long title in bold and close to the key",
       line.clab = 1, side.clab = 2, font.clab = 2)

## =======================================================================
## colorkey added to existing plot
## =======================================================================
par(mfrow = c(1, 1))
image2D(volcano, xlab = "", clab = "m",
        colkey = list(side = 1, length = 0.5, width = 0.5,
                      line.clab = 1))
colkey(side = 3, clim = range(volcano), add = TRUE)

# 'dist' to put colkey within the image
# 'shift' to position colkey to the right or upward
par(mfrow = c(1, 1))
image2D(volcano, colkey = FALSE)
colkey(clim = range(volcano), dist = -0.15, shift = 0.2,
       side = 3, add = TRUE, lab = "key 1", col.clab = "white",
       length = 0.5, width = 0.5, col.axis = "white",
       col.ticks = "white", cex.axis = 0.8)
colkey(clim = range(volcano), dist = -0.1, shift = -0.2,
       side = 4, add = TRUE, lab = "key 2", col.clab = "white",
       length = 0.3, width = 0.5, col.axis = "white",
       col.ticks = "white", col.box = "red", cex.axis = 0.8)
colkey(clim = range(volcano), dist = -0.3,
       side = 1, add = TRUE, lab = "key 3", col.clab = "white",
       length = 0.3, width = 0.5, col.axis = "white",
       col.ticks = "white", at = c(100, 140, 180),
       labels = c("a", "b", "c"), font = 2)
colkey(clim = range(volcano), dist = -0.3, shift = -0.2,
       side = 2, add = TRUE, lab = "key 4", col.clab = "white",
       length = 0.3, width = 0.5, col.axis = "white",
       col.ticks = "white", col.box = "red", cex.axis = 0.8,
       las = 3)

## =======================================================================
## colorkey in other plots
## =======================================================================
par(mfrow = c(1, 1))
par(mar = par("mar") + c(0, 0, -2, 0))
image2D(volcano, clab = "height, m",
        colkey = list(dist = -0.15, shift = 0.2,
                      side = 3, length = 0.5, width = 0.5, line.clab = 2.5,
                      cex.clab = 2, col.clab = "white", col.axis = "white",
```

## Colors

### jet.col generates the matlab-type colors.

### jet2.col is similar but lacks the deep blue colors

### gg.col and gg2.col generate gg-plot-like colors.

### ramp.col creates color schemes by interpolation.

### alpha.col creates transparent colors.

```r
col.ticks = "white", cex.axis = 0.8))

## Several color keys in composite plot

persp3D(z = volcano, zlim = c(-60, 200), phi = 20, bty = "b",
colkey = list(length = 0.2, width = 0.4, shift = 0.15,
cex.axis = 0.8, cex.clab = 0.85), lighting = TRUE, lphi = 90,
clab = c("height","m"), plot = FALSE)

# create gradient in x-direction
Vx <- volcano[-1,] - volcano[-nrow(volcano),]

# add as image with own color key, at bottom
image3D(z = -60, colvar = Vx/10, add = TRUE,
colkey = list(length = 0.2, width = 0.4, shift = -0.15,
cex.axis = 0.8, cex.clab = 0.85),
clab = c("gradient","m/m"), plot = TRUE)

## categorical colors; use addlines = TRUE to separate colors

with(iris, scatter3D(x = Sepal.Length, y = Sepal.Width,
z = Petal.Length, colvar = as.integer(Species),
col = c("orange", "green", "lightblue"), pch = 16, cex = 2,
clim = c(1, 3), ticktype = "detailed", phi = 20,
xlab = "Sepal Length", ylab = "Sepal Width",
zlab = "Petal Length", main = "iris",
colkey = list(at = c(1.33, 2, 2.66), side = 1,
addlines = TRUE, length = 0.5, width = 0.5,
labels = c("setosa", "versicolor", "virginica") )))

# reset plotting parameters
par(mfrow = pm)
par(mar = pmar)
```

---

**Colors**

**Colors, shading, lighting.**

**Description**

jet.col generates the matlab-type colors.

jet2.col is similar but lacks the deep blue colors

gg.col and gg2.col generate gg-plot-like colors.
ramp.col creates color schemes by interpolation.
alpha.col creates transparent colors.
Usage

jet.col (n = 100, alpha = 1)
jet2.col (n = 100, alpha = 1)

gg.col (n = 100, alpha = 1)
gg2.col (n = 100, alpha = 1)
ramp.col (col = c("grey", "black"), n = 100, alpha = 1)

Arguments

n Number of colors to generate.
alpha Value in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque). Transparency defined in the color palette is overruled when lighting or shading is switched on. To combine transparency with lighting or shading, pass argument alpha to the plotting functions directly.
col Colors to interpolate, change.

Details

In addition to the color functions described here, colors can also be adapted by shading and lighting, or made transparent. Shading will be overruled if lighting is not FALSE.

To make colors transparent, use argument alpha, with a value inbetween 0 and 1.

To switch on shading, the argument shade should be given a value inbetween 0 and 1.

To switch on lighting, the argument lighting should be either set to TRUE (in which case default settings will be used) or should be a list with specifications of one of the following: ambient, diffuse, specular, exponent, sr and alpha.

The defaults are: ambient = 0.3, diffuse = 0.6, specular = 1., exponent = 20, sr = 0, alpha = 1

Lighting is defined as the sum of ambient, diffuse and specular light. If N is the normal vector on the facets (3-values, x-, y-, z direction) and I is the light vector, then col = (ambient + Id + sr * Is) * col + (1 -sr) * Is, where Is = specular * abs(Light) ^ exponent, Id = diffuse * Light and Light = sum(N*I).

The lighting algorithm is very simple, i.e. it is flat shading, no interpolation.

Toggling on lighting or shading also requires the input of the angles of the light source, as ltheta and lphi, whose defaults are: ltheta = -135, lphi = 0. This usually works well for shading, but may not be optimal for lighting.

Value

A list with colors.
Colors

Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

References
The gg-plot type of colors gg.plot is a color-blind friendly palette from http://wiki.stdout.org/rcookbook/Graphs.

See Also
colorRamp and colorRampPalette for comparable (and more elaborate) R-functions.

Examples

# save plotting parameters
pm <- par("mfrow")
 pmar <- par("mar")

## =======================================================================
## Transparency and various color schemes
## =======================================================================

par(mfrow = c(3, 3))
for (alph in c(0.25, 0.75))
  image2D(volcano, alpha = alph,
          main = paste("jet.col, alpha = ", alph))
image2D(volcano, main = "jet.col")
image2D(volcano, col = jet2.col(100), main = "jet2.col")
image2D(volcano, col = gg.col(100), main = "gg.col")
image2D(volcano, col = gg2.col(100), main = "gg2.col")
image2D(volcano, col = rainbow(100), main = "rainbow")
image2D(volcano, col = terrain.colors(100), main = "terrain.colors")
image2D(volcano, col = ramp.col(c("blue", "yellow", "green", "red")),
         main = "ramp.col")

## =======================================================================
## Shading, lighting - one color
## =======================================================================

# create grid matrices
X <- seq(0, pi, length.out = 50)
Y <- seq(0, 2*pi, length.out = 50)
M <- mesh(X, Y)
phi <- M$x
theta <- M$y

# x, y and z grids
x <- sin(phi) * cos(theta)
y <- cos(phi)
z <- sin(phi) * sin(theta)

# these are the defaults
p <- list(ambient = 0.3, diffuse = 0.6, specular = 1.,
exponent = 20, sr = 0, alpha = 1)
par(mfrow = c(3, 3), mar = c(0, 0, 0, 0))
Col <- "red"
surf3D(x, y, z, box = FALSE, col = Col, shade = 0.9)
surf3D(x, y, z, box = FALSE, col = Col, lighting = TRUE)
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(ambient = 0))
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(diffuse = 0))
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(diffuse = 1))
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(specular = 0))
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(exponent = 5))
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(exponent = 50))
surf3D(x, y, z, box = FALSE, col = Col, lighting = list(sr = 1))

## =======================================================================
## Shading, lighting with default colors
## =======================================================================

x <- seq(-pi, pi, len = 100)
y <- seq(-pi, pi, len = 100)
grid <- mesh(x, y)

z <- with(grid, cos(x) * sin(y))
cv <- with(grid, -cos(y) * sin(x))

# lphi = 180, ltheta = -130 - good for shade
# lphi = 90, ltheta = 0 - good for lighting

par(mfrow = c(2, 2))
persp3D(z = z, x = x, y = y, colvar = cv, zlim = c(-3, 3), colkey = FALSE)
persp3D(z = z, x = x, y = y, colvar = cv, zlim = c(-3, 3),
lighting = TRUE, colkey = FALSE)
persp3D(z = z, x = x, y = y, colvar = cv, zlim = c(-3, 3),
shade = 0.25, colkey = FALSE)
persp3D(z = z, x = x, y = y, colvar = cv, zlim = c(-3, 3),
lighting = TRUE, lphi = 90, ltheta = 0, colkey = FALSE)

## =======================================================================
## transparency of a vector of colors
## =======================================================================

par(mfrow = c(1, 1))
x <- runif(19)
y <- runif(19)
z <- runif(19)

# split into 5 sections (polygons)
ii <- seq(4, 19, by = 4)
x[ii] <- y[ii] <- z[ii] <- NA

polygon3D(x, y, z, border = "black", lwd = 2,
col = alpha.col(c("red", "lightblue", "yellow", "green", "black"),
                alpha = 0.4))
# the same, now passing alpha as an argument to polygon3D:
## Not run:
polygon3D(x, y, z, border = "black", lwd = 2, 
col = c("red", "lightblue", "yellow", "green", "black"),
alpha = 0.4)
## End(Not run)

# reset plotting parameters
par(mfrow = pm)
par(mar = pmar)

### Description

S3 method `plot.plist` and function `plotdev` plot the plotting list to the current device. Changes can be made to the perspective view, to the lighting and shading, or to make colors transparent.

`getplist` and `setplist` retrieve and store information in the plotting list.

`selectplist` selects parts from the plotting list, based on a user-defined function.

### Usage

```r
getplist()
setplist(plist)
plotdev(...)
## S3 method for class 'plist'
plot(x, ...)
selectplist(plist, SS)
```

### Arguments

- **x, plist**  
The plotting list as generated (invisibly) by any of the 3D plotting functions.

- **SS**  
Function which tests points for inclusion in the plotting list. It should take as argument three vectors \((x, y, z)\) and return a vector of equal length that is either `TRUE` or `FALSE`, denoting whether the point should be selected or not.

- **...**  
Additional arguments to change the view or coloration. Supported arguments to change the view are: `theta`, `phi`, `xlim`, `ylim`, `zlim`, `d`, `r`, `scale`, `expand`. See `perspbox`, `persp`.

Supported arguments to change the lighting, or coloration are: `ltheta`, `lphi`, `shade`, `lighting`. See `jet.col`.  

---

**Composite plots**  
Handling and plotting plotting lists.
Details

All 3-D functions from package plot3D produce or update a plotting list that is local to the package. One can access this plotting list via getplist and setplist. The list is used to plot when, in a 3-D function, the argument plot is TRUE or via function plotdev.

When new 3-D objects are added to a plot, using the add argument of the plotting functions, then everything except the axes, is redrawn on top of what was already there. This means that several object will be drawn multiple times, and this may clutter the output. This may not be visible on your screen, but it may become apparent when exported. Use plotdev to create clean figures, where every object is drawn only once.

The plotting list can contain the following items:

- `mat`, the viewing transformation matrix, a 4 x 4 matrix suitable for projecting 3D coordinates \((x, y, z)\) into the 2D plane using homogeneous 4D coordinates \((x,y,z,v)\).
  It can be used to superimpose additional graphical elements on the 3D plot, by any function that is defined on `persp`.
  It can also be used to add lines, arrows or points, using the function `trans3D`.
- `plt`, with original `plt` parameters and the `plt` parameters used for the main frame.
- `persp`, with settings for the perspective box.
- `xlim,ylim,zlim`, with ranges.
- `scalefac`, the scaling factors in x, y and z direction, used e.g. for shading.
- `dot` other plotting parameters passed to `persp`.
- `colkey, numkey`, with settings for the color key(s).
- `poly, segm, pt, CIpt, labels, arr` the information for drawing polygons, segments, points, points with confidence intervals, labels and arrows, that are part of the plot.

For the item `poly` the elements are:

- `x,y,z` : A matrix with typically 4 or 5 rows, the first rows defining the x-, y- or z- values of each polygon, the last row contains `NA` (and which therefore terminates a polygon).
- `col` : a vector with the colors for the facets of each polygon.
- `lwd, lty, border` : a vector with the line widths, line type and colors for the border of each polygon. (note in R-function `polygon`, passing a vector of line widths is not implemented; therefore, only the first value of `lwd` will be used for all polygons).
  When `plot.plist` is called, the projection depth is calculated and used to sort the facets and function `polygon` used to draw them.

Value

Returns the updated plotting list.

Note

Once a 3D plot has been generated, a new device can be opened and `plotdev` used to plot also on this device.

`plotdev` and `plot(getplist())` are the same.

In an extension package, `plot3Drgl`, a similar function, `plotrgl`, plots the graphs to the device opened with `rgl`. This allows interactive zooming, rotating, etc...
Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>

Examples

# save plotting parameters
pm <- par("mfrow")
pmar <- par("mar")

## The volcano

# The volcano at lower resolution
x <- seq(1, nrow(volcano), by = 2)
y <- seq(1, ncol(volcano), by = 2)
V <- volcano[x, y]
persp3D(z = V)

# rotate
plotdev(theta = 0)

# light and transparency
plotdev(lighting = TRUE, lphi = 90, alpha = 0.6)

# zoom
plotdev(xlim = c(0.2, 0.6), ylim = c(0.2, 0.6), phi = 60)

## Two spheres

# create a sphere
M <- mesh(seq(0, 2*pi, length.out = 30),
          seq(0, pi, length.out = 30))
u <- M$x; v <- M$y

x <- cos(u)*sin(v)
y <- sin(u)*sin(v)
z <- cos(v)

surf3D(x = 2*x, y = 2*y, z = 2*z,
       colvar = NULL, lighting = TRUE, #plot = FALSE,
       facets = NA, col = "blue", lwd = 5)

surf3D(x, y, z, colvar = NULL, lighting = TRUE,
       col = "red", add = TRUE)


names(getplist())

# plot with different view:
plotdev(phi = 0)
## Not run: # will plot same 3-D graph to pdf
dev.off()

## End(Not run)

## Two spheres and two planes

par(mar = c(2, 2, 2, 2))

# equation of a sphere
M <- mesh(seq(0, 2*pi, length.out = 100),
          seq(0, pi, length.out = 100))

u <- M$x ; v <- M$y

x <- cos(u)*sin(v)
y <- sin(u)*sin(v)
z <- cos(v)

surf3D(x, y, z, colvar = z,
       theta = 45, phi = 20, bty = "b",
       xlim = c(-1.5, 1.5), ylim = c(-1, 2),
       zlim = c(-1.5, 1.5), plot = FALSE)

# add a second sphere, shifted 1 unit to the right on y-axis;
# no facets drawn for this sphere
surf3D (x, y+1, z, colvar = z, add = TRUE,
        facets = FALSE, plot = FALSE)

# define a plane at z = 0
Nx <- 100
Ny <- 100

x <- seq(-1.5, 1.5, length.out = Nx)
y <- seq(-1, 2, length.out = Ny)

image3D (x = x, y = y, z = 0, add = TRUE, colvar = NULL,
         col = "blue", facets = TRUE, plot = FALSE)

# another, small plane at y = 0 - here x and y have to be matrices!
x <- seq(-1., 1., length.out = 50)
z <- seq(-1., 1., length.out = 50)

image3D (x = x, y = 0, z = z, colvar = NULL,
         add = TRUE, col = NA, border = "blue",
         facets = TRUE, plot = FALSE)
facets = TRUE, plot = TRUE)

## Not run:  # rotate
for (angle in seq(0, 360, by = 10))
  plotdev(theta = angle)

## End(Not run)

## Zooming, rescaling, lighting,...

par(mfrow = c(2, 2))

# The volcano
x <- seq(1, nrow(volcano), by = 2)
y <- seq(1, ncol(volcano), by = 2)
V <- volcano[x, y]
# plot the volcano
persp3D(x, y, z = V, colvar = V, theta = 10, phi = 20,
  box = FALSE, scale = FALSE, expand = 0.3,
  clim = range(V), plot = FALSE)

# add a plane (image) at z = 170; jetcolored, transparant: only border
image3D(x, y, z = 170, add = TRUE, clim = range(V),
  colvar = V, facets = NA, plot = FALSE, colkey = FALSE)

# add a contour (image) at z = 170; jetcolored,
contour3D(x, y, z = 170, add = TRUE, clim = range(V),
  colvar = V, plot = FALSE, colkey = FALSE)

# plot it -
plot(getplist())  # same as plotdev()

# plot but with different expansion
plotdev(expand = 1)

# other perspective, and shading
plotdev(d = 2, r = 10, shade = 0.3)

# zoom and rotate
plotdev(xlim = c(10, 30), ylim = c(20, 30), phi = 50)

## Using setplist

polygon3D(runif(3), runif(3), runif(3))
# retrieve plotting list
plist <- getplist()
names(plist)
plist$poly

# change copy of plotting list
images in 3D frame  

```r
plist$poly$col <- "red"
# update internal plotting list
setplist(plist)
# plot updated list
plotdev()

## ========================================================================
## Using selectplist
## ========================================================================

polygon3D(runif(10), runif(10), runif(10), col = "red",
alpha = 0.2, plot = FALSE, ticktype = "detailed",
xlim = c(0, 1), ylim = c(0, 1), zlim = c(0, 1))
polygon3D(runif(10)*0.5, runif(10), runif(10), col = "yellow",
alpha = 0.2, plot = FALSE, add = TRUE)
polygon3D(runif(10)*0.5+0.5, runif(10), runif(10), col = "green",
alpha = 0.2, plot = FALSE, add = TRUE)
points3D(runif(10), runif(10), runif(10), col = "blue",
add = TRUE, plot = FALSE)
segments3D(x0 = runif(10), y0 = runif(10), z0 = runif(10),
x1 = runif(10), y1 = runif(10), z1 = runif(10),
colvar = 1:10, add = TRUE, lwd = 3)

# retrieve plotting list
plist <- getplist()

# selection function
SS <- function (x, y, z) {
  sel <- rep(TRUE, length.out = length(x))
  sel[x < 0.5] <- FALSE
  return(sel)
}
# The whole polygon will be removed or kept.
plot(x = selectplist(plist, SS),
  xlim = c(0, 1), ylim = c(0, 1), zlim = c(0, 1))

# restore plotting parameters
par(mfrow = pm)
par(mar = pmar)
```

Description

`image3D` adds an image in a 3-D plot.
Usage

image3D (x = NULL, y = NULL, z = NULL, ..., colvar = NULL,
phi = 40, theta = 40, col = NULL,
NAcol = "white", breaks = NULL, border = NA, facets = TRUE,
colkey = NULL, resfac = 1, panel.first = NULL,
clim = NULL, clab = NULL, bty = "b",
inttype = 1, add = FALSE, plot = TRUE)

Arguments

x, y, z
Matrix (2-D), vector, or one value containing the values where the image is to
be plotted. At least one of them should be one number, as this will determine
where the image is plotted, parallel to the (y-z) plane (x one number), to the
(x-z) plane (y one number) or to the (z-y) plane (z one number).
If two are vectors, the first vector should be of length equal to nrow(colvar)
and the second should be of length equal to ncol(colvar).

colvar
The variable used for coloring.

col
Color palette to be used for the colvar variable.

NAcol
Color to be used for NA values of colvar; default is "white".

breaks
a set of finite numeric breakpoints for the colors; must have one more breakpoint
than color and be in increasing order. Unsorted vectors will be sorted, with a
warning.

colkey
A logical, NULL (default), or a list with parameters for the color key (legend).
List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab
and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col.
The defaults for the parameters are side = 4, plot = TRUE, length = 1, width =
1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab =
= NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL) See colkey.
The default is to draw the color key on side = 4, i.e. in the right margin. If
colkey = NULL then a color key will be added only if col is a vector. Setting
colkey = list(plot = FALSE) will create room for the color key without draw-
ing it. if colkey = FALSE, no color key legend will be added.

clab
Only if colkey = TRUE, the label to be written on top of the color key. The label
will be written at the same level as the main title. to lower it, clab can be made
a vector, with the first values empty strings.

clim
Only if colvar is specified, the range of the color variable, used for the color
key. Values of colvar that extend the range will be put to NA.

resfac
Resolution factor, one value or a vector of two numbers, for the x and y- values
respectively. A value > 1 will increase the resolution. For instance, if resfac
equals 3 then for each adjacent pair of x- and y-values, z will be interpolated to
two intermediary points. This uses simple linear interpolation. If resfac is one
number then the resolution will be increased similarly in x and y-direction.

theta, phi
The angles defining the viewing direction. theta gives the azimuthal direction
and phi the colatitude. see persp.
border: The color of the lines drawn around the surface facets. The default, NA, will disable the drawing of borders.

facets: If TRUE, then col1 denotes the color of the surface facets. If FALSE, then the surface facets are colored “white” and the border (if NA) will be colored as specified by col1. If NA then the facets will be transparent. It is usually faster to draw with facets = FALSE.

panel.first: A function to be evaluated after the plot axes are set up (and if applicable, images or contours drawn) but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix, e.g., it should be defined as function(pmat).
See example of persp3D and last example of voxel3D.

bty: The type of the box, the default only drawing background panels. Only effective if the persp argument (box) equals TRUE (this is the default). See perspbox.

inttype: The interpolation type to create the polygons, either taking the mean of the colvar variable (inttype = 1, 3) or extending the x, y, z values (inttype = 2). Values 1, 3 differ in how they treat NAs in the colvar variable. For inttype = 3, NAs are removed before taking averages; this will tend to make the NA region smaller. NAs are included when inttype = 1. This will tend to make the NA region larger. See details and an example in persp3D.

add: Logical. If TRUE, then the image will be added to the current plot. If FALSE a new plot is started.

plot: Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

... additional arguments passed to the plotting methods.

The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, scale, expand, box, axes, nticks, ticktype.
The arguments xlim, ylim, zlim only affect the axes. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.

In addition, the perspbox arguments col.axis, col.panel, lwd.panel, col.grid, lwd.grid can also be given a value.
shade and lighting arguments will have no effect.
alpha can be given a value inbetween 0 and 1 to make colors transparent.
Also the arguments lty, lwd can be specified (when border is not NA).
The arguments after ... must be matched exactly.

Details
image3D calls the surf3D function. The x, y, and z values are expanded as a matrix.

Value
Returns the viewing transformation matrix. See trans3D.

Author(s)
Karline Soetaert <karline.soetaert@nioz.nl>
See Also

surf3D for the function on which image3D is based.
image2D for plot3D's 2-D image function.

Examples

# save plotting parameters
pm <- par("mfrow")

## images in x, y, z plane

par(mfrow = c(2, 2))

# images in x, y, z plane
# We use colkey = list(plot = FALSE) to create room for a color key
debug(image3D(y = seq(0, 1, 0.1), z = seq(0, 1, 0.1), x = 0.5,  
        col = "blue", xlab = c(0, 1), colkey = list(plot = FALSE)))
image3D(x = seq(0, 1, 0.1), z = seq(0, 1, 0.1), y = 0.5,  
        add = TRUE, col = "red", alpha = 0.2)  
        # alpha makes it transparent
image3D(x = seq(0, 1, 0.1), y = seq(0, 1, 0.1), z = 0.5,  
        add = TRUE, col = "green")
colkey(col = c("green", "red", "blue"), ylim = c(0.5, 3.5),  
        at = 1:3, labels = c("z", "y", "x"), add = TRUE)

# image3D(z = 100, colvar = volcano, zlim = c(0, 150),  
        clab = c("height", "m"))

# image3D( x = 0.5, colvar = volcano, xlim = c(0, 1),  
        ylim = c(0, 1), zlim = c(0, 1))

image3D( y = 0.5, colvar = volcano, add = TRUE)

# image3D( z = 1, colvar = volcano,  
        x = seq(0, 1, length.out = nrow(volcano)),  
        y = seq(0, 1, length.out = ncol(volcano)),  
        xlim = c(0, 2), ylim = c(0, 2), zlim = c(0, 2))
image3D(y = 2, colvar = volcano, add = TRUE,  
        shade = 0.2,  
        x = seq(0, 1, length.out = nrow(volcano)),  
        z = seq(1, 2, length.out = ncol(volcano)))
image3D(x = 2, colvar = NULL, col = "orange", add = TRUE,  
        y = seq(0, 1, length.out = nrow(volcano)),  
        z = seq(1, 2, length.out = ncol(volcano)))

# reset plotting parameters
par(mfrow = pm)
Mesh generation

Description

mesh creates a rectangular full 2-D or 3-D grid.

Usage

mesh (x, y, z = NULL)

Arguments

x, y, z

Vectors with x, y and z-values. They can be of arbitrary length.

Value

Function mesh returns a list with the expanded x- y- and z arrays (in case z is not NULL) or matrices (in case z = NULL). The dimensions of these list elements are the same and equal to c(length(x),length(y),length(z)).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

persp3D, arrows3D, slice3D, surf3D for other examples that use mesh.

Examples

```r
## 2-D mesh
x <- c(-1, 0, 1)
y <- 1:4
(M <- mesh(x, y))
V <- with(M, x/2 * sin(y))
V2 <- outer(x, y, FUN = function(x, y) x/2*sin(y))
```
## 3-D mesh

```r
x <- y <- z <- c(-1 , 0, 1)

# 3-D mesh
(M <- mesh(x, y, z))

# calculate with 3-D mesh
V <- with(M, x/2 * sin(y) * sqrt(z+2))

# plot result
scatter3D(M$x, M$y, M$z, V, pch = "+", cex = 3, colkey = FALSE)
```

---

**Description**

`perspbox` draws a box and labels, and makes space for a colorkey (if any).

**Usage**

```r
perspbox (x = seq(0, 1, length.out = nrow(z)), y = seq(0, 1, length.out = ncol(z)), z, bty = c("b", "b2", "f", "g", "bl", "bl2", "u", "n"), ..., col.axis = "black", col.panel = NULL, lwd.panel = 1, col.grid = NULL, lwd.grid = 1, phi = 40, theta = 40, col = NULL, colkey = NULL, plot = TRUE)
```

**Arguments**

- **x, y**
  - Vectors with x and y values. It is sufficient to pass the ranges of the x- and y-values, as they will not be drawn. If z is a matrix, it is required that `length(x) = nrow(z)` and `length(y) = ncol(z)`.

- **z**
  - Matrix or vector with z-values. If z is a matrix, it is sufficient to pass a diagonal matrix with the range of the z-values, as they will not be drawn.

- **bty**
  - The type of the box; only effective if the `persp` argument box equals `TRUE` (the default). Unless bty is equal to "u" then the arguments `col.axis`, `col.panel`, `lwd.panel`, `col.grid`, `lwd.grid` will be ignored. "f" is the full box, the default as from `persp`, "b" has only the back panels visible, when "b2" has back panels and grid lines, "g" has grey background with white gridlines, "bl" has a black background, "bl2" has a black background with grey lines. "u" means that the user will specify the arguments `col.axis`, `col.panel`, `lwd.panel`, `col.grid`, `lwd.grid` manually. "n" means that no box will be drawn. This is the same as setting `box = FALSE`. 

---

**Perspective box**

*Creates an empty perspective box, ready for adding objects*
col.axis, col.panel, col.grid

The color of the axis line, of the axis panel or of the grid lines. Only used if bty = "u".

lwd.panel, lwd.grid

The width of the panel border or of the grid lines. Only used if bty = "u".

theta, phi

The angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude. See persp.

col

Colors to be used for coloring the colvar variable. Here only used for assessing if a color key should be drawn.

colkey

A logical, NULL (default), or a list with parameters for the color key (legend). List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab, and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col.ticks, and col.grid. The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = 4, at = NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL. See colkey.

The default is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. If colkey = FALSE, no color key legend will be added.

plot

Logical. If TRUE (default), a plot is created, otherwise the viewing transformation matrix is returned (as invisible).

... additional arguments passed to persp.

The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, d, s, and scales. Arguments scale and expand affect the size of the axes. The arguments after ... must be matched exactly.

Details

The arguments xlim, ylim, zlim only affect the axes. All objects will be plotted, including those that fall out of these ranges. To select objects only within the axis limits, use plotdev.

The predefined box types bty are defined as follows:

"f": all panels are shown and transparent, also the persp default.

"b": only backward panels shown.

"b2": as "b" with col.grid = "grey".

"g": only backward panels shown; col.panel = grey(0.95), col.axis = "grey", lwd.grid = 2 and col.grid = "white".

"bl": only backward panels shown; col.panel = "black", col.axis = "grey", lwd.grid = 2 and col.grid = "white".

"n": no box is drawn.

Value

Function perspbox returns the viewing transformation matrix. See trans3D.
Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

persp3D, scatter2D, surf3D for examples where box types different than the default are used.
Hypsometry for an example where colored axis-panels are added to a figure started with perspbox.

Examples

# save plotting parameters
pm <- par("mfrow")
pmar <- par("mar")

# The 4 predefined box types

par(mfrow = c(2, 2), mar = c(1, 1, 1, 1))
# box type with only backward panels
perspbox(z = volcano, bty = "b", ticktype = "detailed", d = 2,
        main = "bty = 'b'")
# box as in 'persp'
perspbox(z = volcano, bty = "f", ticktype = "detailed",
        d = 2, main = "bty = 'f'")
# back panels with gridlines, detailed axes
perspbox(z = volcano, bty = "b2", ticktype = "detailed",
        d = 2, main = "bty = 'b2'")
# ggplot-type, simple axes
perspbox(z = volcano, bty = "g",
        d = 2, main = "bty = 'g'")

# A user-defined box

par(mfrow = c(1, 1))
perspbox(z = diag(2), bty = "u", ticktype = "detailed",
        col.panel = "gold", col.axis = "white",
        scale = FALSE, expand = 0.4,
        col.grid = "grey", main = "user-defined")

# restore plotting parameters
par(mfrow = pm)
par(mar = pmar)
Scatter plots

**Description**

scatter2D and scatter3D plot a (2- or 3-dimensional) dataset with a color variable as points or lines.

text3D plot a 3-D dataset with a color variable as text labels.

points3D is shorthand for `scatter3D(..., type = "p")`

lines3D is shorthand for `scatter3D(..., type = "l")`

points2D is shorthand for `scatter2D(..., type = "p")`

lines2D is shorthand for `scatter2D(..., type = "l")`

The 2D functions are included for their side effect of having a color key.

**Usage**

`scatter3D (x, y, z, ..., colvar = z, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", CI = NULL, surf = NULL, add = FALSE, plot = TRUE)`

`text3D (x, y, z, labels, ..., colvar = NULL, phi = 40, theta = 40, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, panel.first = NULL, clim = NULL, clab = NULL, bty = "b", CI = NULL, surf = NULL, add = FALSE, plot = TRUE)`

`points3D (x, y, z, ...)`

`lines3D (x, y, z, ...)`

`scatter2D (x, y, ..., colvar = NULL, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, clim = NULL, clab = NULL, CI = NULL, add = FALSE, plot = TRUE)`

`lines2D(x, y, ...)`

`points2D(x, y, ...)`

`text2D (x, y, labels, ..., colvar = NULL, col = NULL, NAcol = "white", breaks = NULL, colkey = NULL, clim = NULL, clab = NULL, add = FALSE, plot = TRUE)`
Arguments

x, y, z
Vectors with x, y and z-values of the points to be plotted. They should be of equal length, and the same length as colvar (if present).

colvar
The variable used for coloring. For scatter3D, it need not be present, but if specified, it should be a vector of equal length as (x, y, z).

theta, phi
the angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude. see persp.

col
Color palette to be used for coloring the colvar variable. If col is NULL and colvar is specified, then a red-yellow-blue colorscheme (jet.col) will be used. If col is NULL and colvar is not specified, then col will be "black".

NAcol
Colors to be used for colvar values that are NA.

breaks
a set of finite numeric breakpoints for the colors; must have one more breakpoint than color and be in increasing order. Unsorted vectors will be sorted, with a warning.

colkey
A logical, NULL (default), or a list with parameters for the color key (legend). List parameters should be one of side, plot, length, width, dist, shift, addlines, col.clab, cex.clab and the axis parameters at, labels, tick, line, pos, outer, font, lty, lwd, lwd.ticks, col.box, col.axis, col.ticks.

The defaults for the parameters are side = 4, plot = TRUE, length = 1, width = 1, dist = 0, shift = 0, addlines = FALSE, col.clab = NULL, cex.clab = par("cex.lab"), side.clab = NULL, line.clab = NULL, adj.clab = NULL, font.clab = NULL) See colkey.

The default is to draw the color key on side = 4, i.e. in the right margin. If colkey = NULL then a color key will be added only if col is a vector. Setting colkey = list(plot = FALSE) will create room for the color key without drawing it. if colkey = FALSE, no color key legend will be added.

CI
A list with parameters and values for the confidence intervals or NULL. If a list it should contain at least the item x, y or z (latter for scatter3D). These should be 2-columned matrices, defining the left/right intervals. Other parameters should be one of (with defaults): alen = 0.01, lty = par("lty"), lw = par("lwd"), col = NULL, to set the length of the arrow head, the line type and width, and the color. If col is NULL, then the colors as specified by colvar are used. See examples.

panel.first
A function to be evaluated after the plot axes are set up but before any plotting takes place. This can be useful for drawing background grids or scatterplot smooths. The function should have as argument the transformation matrix, e.g. it should be defined as function(pmat). See example of persp3D and last example of voxel3D.

clab
Only if colkey is not NULL or FALSE, the label to be written on top of the color key. The label will be written at the same level as the main title. To lower it, clab can be made a vector, with the first values empty strings.

clim
Only if colvar is specified, the range of the color variable, used for the color key. Values of colvar that extend the range will be put to NA.

bty
The type of the box, the default draws only the back panels. Only effective if the persp argument (box) equals TRUE (this is the default). See perspbox. Note: the bty = "g", "b2", "b1" can also be specified for scatter2D (if add = FALSE).
labels

The text to be written. A vector of length equal to length of x, y, z.

surf

If not NULL, a list specifying a (fitted) surface to be added on the scatterplot. The list should include at least x, y, z, defining the surface, and optional: colvar, col, NAcol, border, facets. Note that the default is that colvar is not specified which will set colvar = z. The argument fit should give the fitted z-values, in the same order as the z-values of the scatter points, for instance produced by predict. When present, this will produce droplines from points to the fitted surface.

add

Logical. If TRUE, then the points will be added to the current plot. If FALSE a new plot is started.

plot

Logical. If TRUE (default), a plot is created, otherwise (for 3D plots) the viewing transformation matrix is returned (as invisible).

... additional arguments passed to the plotting methods.

The following persp arguments can be specified: xlim, ylim, zlim, xlab, ylab, zlab, main, sub, r, d, sc, xlimf, ylimf, zlimf, resf, box, boxframe, cex.axis, cex.lab, cex.main, font, ticktype, tickcol, ticklab, xrot, yrot, zrot, perspbox, shade, lighting, alpha, ltheta, lphi, shade, shading, default = FALSE. For plot3D, the arguments lty, lwd can be specified; type can be specified for all except text3D.

In case type = "b" or "o", then pch, cex, bg can also be specified.

The arguments after ... must be matched exactly.

Value

Function scatter3D returns the viewing transformation matrix. See trans3D.

Note

For scatter2D and scatter3D the plottypes that are supported are: type = "p", type = "l", type = "h", type = "o". For type = "b", type = "o" is used instead.

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

persp for the function on which this implementation is based.

mesh, trans3D, slice3D, for other examples of scatter2D or scatter3D.

plotdev for zooming, rescaling, rotating a plot.

package scatterplot3D for an implementation of scatterplots that is not based on persp.
Examples

# save plotting parameters
pm <- par("mfrow")

## A sphere

par(mfrow = c(1, 1))
M <- mesh(seq(0, 2*pi, length.out = 100),
    seq(0, pi, length.out = 100))
u <- M$x ; v <- M$y
x <- cos(u)*sin(v)
y <- sin(u)*sin(v)
z <- cos(v)

# full panels of box are drawn (bty = "f")
scatter3D(x, y, z, pch = ".", cex = 2, col = "red",
bty = "f", colkey = FALSE)

## Different types

par (mfrow = c(2, 2))
z <- seq(0, 10, 0.2)
x <- cos(z)
y <- sin(z)*z

# greyish background for the boxtype (bty = "g")
scatter3D(x, y, z, phi = 0, bty = "g",
    pch = 20, cex = 2, ticktype = "detailed")

# add another point
scatter3D(x = 0, y = 0, z = 0, add = TRUE, colkey = FALSE,
    pch = 18, cex = 3, col = "black")

# add text
text3D(x = cos(1:10), y = (sin(1:10)*(1:10) - 1),
    z = 1:10, colkey = FALSE, add = TRUE,
    labels = LETTERS[1:10], col = c("black", "red"))

# line plot
scatter3D(x, y, z, phi = 0, bty = "g", type = "l",
    ticktype = "detailed", lwd = 4)

# points and lines
scatter3D(x, y, z, phi = 0, bty = "g", type = "b",
    ticktype = "detailed", pch = 20,
    cex = c(0.5, 1, 1.5))
# vertical lines
scatter3D(x, y, z, phi = 0, bty = "g", type = "h",
    ticktype = "detailed")

## With confidence interval

x <- runif(20)
y <- runif(20)
z <- runif(20)

par(mfrow = c(1, 1))
CI <- list(z = matrix(nrow = length(x),
    data = rep(0.05, 2*length(x))))

# greyish background for the boxtype (bty = "g")
scatter3D(x, y, z, phi = 0, bty = "g", CI = CI,
    col = gg.col(100), pch = 18, cex = 2, ticktype = "detailed",
    xlim = c(0, 1), ylim = c(0, 1), zlim = c(0, 1))

# add new set of points
x <- runif(20)
y <- runif(20)
z <- runif(20)

CI2 <- list(x = matrix(nrow = length(x),
    data = rep(0.05, 2*length(x))),
    z = matrix(nrow = length(x),
    data = rep(0.05, 2*length(x))))

scatter3D(x, y, z, CI = CI2, add = TRUE, col = "red", pch = 16)

## With a surface

par(mfrow = c(1, 1))

# surface = volcano
M <- mesh(1:nrow(volcano), 1:ncol(volcano))
N <- 100
xs <- runif(N) * 87
ys <- runif(N) * 61
zs <- runif(N)*50 + 154

# scatter + surface
scatter3D(xs, ys, zs, ticktype = "detailed", pch = 16,
    bty = "f", xlim = c(1, 87), ylim = c(1,61), zlim = c(94, 215),
    surf = list(x = M$x, y = M$y, z = volcano,
        NAcol = "grey", shade = 0.1))
**Scatter plots**

```r
## A surface and CI
## =======================================================================
par(mfrow = c(1, 1))
M <- mesh(seq(0, 2*pi, length = 30), (1:30)/100)
z <- with (M, sin(x) + y)

# points 'sampled'
N <- 30
xs <- runif(N) * 2*pi
ys <- runif(N) * 0.3
zs <- sin(xs) + ys + rnorm(N)*0.3

CI <- list(z = matrix(nrow = length(xs),
data = rep(0.3, 2*length(xs))),
lwd = 3)

# facets = NA makes a transparent surface; borders are black
scatter3D(xs, ys, zs, ticktype = "detailed", pch = 16,
xlim = c(0, 2*pi), ylim = c(0, 0.3), zlim = c(-1.5, 1.5),
CI = CI, theta = 20, phi = 30, cex = 2,
surf = list(x = M$x, y = M$y, z = z, border = "black", facets = NA))

## droplines till the fitted surface
## =======================================================================
with (mtcars, {

# linear regression
fit <- lm(mpg ~ wt + disp)

# predict values on regular xy grid
wt.pred <- seq(1.5, 5.5, length.out = 30)
disp.pred <- seq(71, 472, length.out = 30)
xy <- expand.grid(wt = wt.pred,
disp = disp.pred)

mpg.pred <- matrix (nrow = 30, ncol = 30,
data = predict(fit, newdata = data.frame(xy),
interval = "prediction"))

# fitted points for droplines to surface
fitpoints <- predict(fit)
scatter3D(z = mpg, x = wt, y = disp, pch = 18, cex = 2,
theta = 20, phi = 20, ticktype = "detailed",
xlab = "wt", ylab = "disp", zlab = "mpg",
surf = list(x = wt.pred, y = disp.pred, z = mpg.pred,
theta = 20, phi = 30, facets = NA, border = "black"))
})
```
Scatter plots

```r
facets = NA, fit = fitpoints),
main = "mtcars")
```

## Two ways to make a scatter 3D of quakes data set

```r
par(mfrow = c(1, 1))
# first way, use vertical spikes (type = "h")
with(quakes, scatter3D(x = long, y = lat, z = -depth, colvar = mag,
pch = 16, cex = 1.5, xlab = "longitude", ylab = "latitude",
zlab = "depth, km", clab = c("Richter","Magnitude"),
main = "Earthquakes off Fiji", ticktype = "detailed",
type = "h", theta = 10, d = 2,
colkey = list(length = 0.5, width = 0.5, cex.clab = 0.75))

# second way: add dots on bottom and left panel
# before the scatters are drawn,
# add small dots on basal plane and on the depth plane
panelfirst <- function(pmat) {
  zmin <- min(-quakes$depth)
  XY <- trans3D(quakes$long, quakes$lat,
z = rep(zmin, nrow(quakes)), pmat = pmat)
  scatter2D(XY$x, XY$y, colvar = quakes$mag, pch = ".",
cex = 2, add = TRUE, colkey = FALSE)
  xmin <- min(quakes$long)
  XY <- trans3D(x = rep(xmin, nrow(quakes)), y = quakes$lat,
z = -quakes$depth, pmat = pmat)
  scatter2D(XY$x, XY$y, colvar = quakes$mag, pch = ".",
cex = 2, add = TRUE, colkey = FALSE)
}

with(quakes, scatter3D(x = long, y = lat, z = -depth, colvar = mag,
pch = 16, cex = 1.5, xlab = "longitude", ylab = "latitude",
zlab = "depth, km", clab = c("Richter","Magnitude"),
main = "Earthquakes off Fiji", ticktype = "detailed",
panel.first = panelfirst, theta = 10, d = 2,
colkey = list(length = 0.5, width = 0.5, cex.clab = 0.75))
```

## text3D and scatter3D

```r
with(USArrests, text3D(Murder, Assault, Rape,
colvar = UrbanPop, col = gg.col(100), theta = 60, phi = 20,
xlab = "Murder", ylab = "Assault", zlab = "Rape",
main = "USA arrests",
label = rownames(USArrests), cex = 0.6,
```
Scatter plots

bty = "g", ticktype = "detailed", d = 2,
clab = c("Urban","Pop"), adj = 0.5, font = 2)

with(USArrests, scatter3D(Murder, Assault, Rape - 1,
colvar = UrbanPop, col = gg.col(100),
type = "h", pch = ".", add = TRUE))

## zoom near origin

# display axis ranges
getplist()[c("xlim","ylim","zlim")]

# choose suitable ranges
plotdev(xlim = c(0, 10), ylim = c(40, 150),
zlim = c(7, 25))

## text3D to label x- and y axis

par(mfrow = c(1, 1))
hist3D (x = 1:5, y = 1:4, z = VADeaths,
 bty = "g", phi = 20, theta = -60,
 xlab = "", ylab = "", zlab = "", main = "VADeaths",
col = "#0072B2", border = "black", shade = 0.8,
ticktype = "detailed", space = 0.15, d = 2, cex.axis = 1e-9)

text3D(x = 1:5, y = rep(0.5, 5), z = rep(3, 5),
labels = rownames(VADeaths),
add = TRUE, adj = 0)
text3D(x = rep(1, 4), y = 1:4, z = rep(0, 4),
labels = colnames(VADeaths),
add = TRUE, adj = 1)

## Scatter2D; bty can also be set = to one of the perspbox alternatives

par(mfrow = c(2, 2))
x <- seq(0, 2*pi, length.out = 30)
scatter2D(x, sin(x), colvar = cos(x), pch = 16,
ylab = "sin", clab = "cos", cex = 1.5)

# other box types:
scatter2D(x, sin(x), colvar = cos(x), type = "l", lwd = 4, bty = "g")
scatter2D(x, sin(x), colvar = cos(x), type = "b", lwd = 2, bty = "b2")
# transparent colors and spikes
scatter2D(x, sin(x), colvar = cos(x), type = "h", lwd = 4, alpha = 0.5)

## mesh examples and scatter2D
par(mfrow = c(1, 2))
x <- seq(-1, 1, by = 0.1)
y <- seq(-2, 2, by = 0.2)

grid <- mesh(x, y)
z <- with(grid, cos(x) * sin(y))
image2D(z, x = x, y = y)
points(grid)
scatter2D(grid$x, grid$y, colvar = z, pch = 20, cex = 2)

## scatter plot with confidence intervals

par(mfrow = c(2, 2))
x <- sort(rnorm(10))
y <- runif(10)
cv <- sqrt(x^2 + y^2)
CI <- list(lwd = 2)
CI$x <- matrix(nrow = length(x), data = c(rep(0.25, 2*length(x))))
scatter2D(x, y, colvar = cv, pch = 16, cex = 2, CI = CI)
scatter2D(x, y, colvar = cv, pch = 16, cex = 2, CI = CI, type = "b")

CI$y <- matrix(nrow = length(x), data = c(rep(0.05, 2*length(x))))
CI$col <- "black"
scatter2D(x, y, colvar = cv, pch = 16, cex = 2, CI = CI)

CI$y[c(2,4,8,10), ] <- NA # Some points have no CI
CI$x[c(2,4,8,10), ] <- NA # Some points have no CI
CI$alen <- 0.02 # increase arrow head
scatter2D(x, y, colvar = cv, pch = 16, cex = 2, CI = CI)

## Scatter on an image

par(mfrow = c(1, 1))
# image of oxygen saturation
oxlim <- range(Oxsat$val[,1], na.rm = TRUE)
image2D(z = Oxsat$val[,1], x = Oxsat$lon, y = Oxsat$lat,
        contour = TRUE,
        xlab = "longitude", ylab = "latitude",
        main = "Oxygen saturation", clim = oxlim, clab = ")

# (imaginary) measurements at 5 sites
lon <- c(11.2, 6.0, 0.9, -4, -8.8)
lat <- c(-19.7,-14.45,-9.1,-3.8, -1.5)
O2sat <- c(90, 95, 92, 85, 100)

# add to image; use same zrange; avoid adding a color key
scatter2D(colvar = O2sat, x = lon, y = lat, clim = oxlim, pch = 16,
    add = TRUE, cex = 2, colkey = FALSE)

## Scatter on a contourplot

par(mfrow = c(1, 1))

# room for colorkey by setting colkey = list(plot = FALSE)

# contour plot of the ocean's bathymetry
Depth <- Hypsometry$z
Depth[Depth > 0] <- NA
contour2D(z = Depth, x = Hypsometry$x, y = Hypsometry$y,
    xlab = "longitude", ylab = "latitude",
    col = "black", NACol = "grey", levels = seq(-6000, 0, by = 2000),
    main = "Oxygen saturation along ship track",
    colkey = list(plot = FALSE))

# add data to image; with a color key
scatter2D(colvar = O2sat, x = lon, y = lat, pch = 16,
    add = TRUE, cex = 2, clab = "%")

## scatter2D for time-series plots

# Plotting sunspot 'anomalies'
sunspot <- data.frame(year = time(sunspot.month),
    anom = sunspot.month - mean(sunspot.month))

# long-term moving average of anomaly
ff <- 100
sunspot$ma <- filter(sunspot$anom, rep(1/ff, ff), sides = 2)

with (sunspot, lines2D(year, anom,
    colvar = anom > 0,
    col = c("pink", "lightblue"),
    main = "sunspot anomaly", type = "h",
    colkey = FALSE, las = 1, xlab = "year", ylab = "")

lines2D(sunspot$year, sunspot$ma, add = TRUE)

# The same
# with (sunspot, plot(year, anom,
#    col = c("pink", "lightblue")[(anom > 0) + 1],
#    main = "sunspot", type = "h", las = 1))

# but this does not work due to NAs...
# lines(sunspot$year, sunspot$ma)

## text2D
trans3D

### Transformation of 3D elements

**Description**

trans3D is the `plot3D` equivalent of `trans3d`, that projects 3-D elements to 2 dimensions.

**Usage**

```r
trans3D (x, y, z, pmat)
```

**Arguments**

- `x, y, z`: Vectors, matrices, arrays, with x, y and z-values.
- `pmat`: A 4 x 4 viewing transformation matrix, suitable for projecting the 3D coordinates (x,y,z) into the 2D plane using homogeneous 4D coordinates (x,y,z,t); such matrices are returned by any of the 3-D plotting functions from package `plot3D` and by `persp()`.

**Value**

A list with two components:

- `x, y` the projected 2-D coordinates of the 3-D input `x, y, z`.

In contrast to `trans3d`, `trans3D` the returned values `x` and `y` will be of the same class and dimensions as the input `x` and `y`. If inputted `x, y, z` are matrices or arrays, so will the projected coordinates be.

**Author(s)**

Karline Soetaert <karline.soetaert@nioz.nl>

**See Also**

- `scatter3D`, `slice3D`, `surf3D`
trans3D

Examples

```r
## ========================================================================
## 3-D mesh
## ========================================================================

x <- y <- z <- c(-1, 0, 1)

# plot a 3-D mesh
(M <- mesh(x, y, z))

# plot result
pmat <- scatter3D(M$x, M$y, M$z, pch = "+", cex = 3, colkey = FALSE)

# add line
XY <- trans3D(x = c(-1, 1), y = c(-1, 1), z = c(-1, 1), pmat = pmat)
lines(XY, lwd = 2, col = "blue")

## ========================================================================
## Example 2
## ========================================================================

pmat <- perspbox(z = diag(2))
XY <- trans3D(x = runif(30), y = runif(30), z = runif(30), pmat = pmat)
polygon(XY, col = "darkblue")
```
Index

*Topic datasets
  3-D data set, 23
*Topic hplot
  2-D data set, 4
  2D image and contour plots, 6
  3-D arrows, segments, polygons, boxes, rectangles, 13
  3-D contours, 20
  3-D perspectives, 25
  3-D surfaces, 34
  3-D volume visualisation, 40
  Color key legend, 47
  Colors, 51
  Composite plots, 55
  images in 3D frame, 60
  Mesh generation, 64
  Perspective box, 65
  Scatter plots, 68
  trans3D, 78
*Topic package
  plot3D-package, 2
  2-D data set, 4
  2D image and contour plots, 6
  3-D arrows, segments, polygons, boxes, rectangles, 13
  3-D contours, 20
  3-D data set, 23
  3-D perspectives, 25
  3-D surfaces, 34
  3-D volume visualisation, 40
  alpha.col (Colors), 51
  arrows, 16, 17
  arrows2D, 3
  arrows2D (3-D arrows, segments, polygons, boxes, rectangles), 13
  arrows3D, 3, 64
  arrows3D (3-D arrows, segments, polygons, boxes, rectangles), 13
  axis, 49
  border3D, 3
  border3D (3-D arrows, segments, polygons, boxes, rectangles), 13
  box3D, 3
  box3D (3-D arrows, segments, polygons, boxes, rectangles), 13
  colkey, 3, 7, 15, 21, 27, 35, 42, 61, 66, 69
  colkey (Color key legend), 47
  Color key legend, 47
  colorRamp, 53
  colorRampPalette, 53
  Colors, 51
  Composite plots, 55
  computeContour3d, 41
  contour, 6, 7, 9, 20, 22, 27, 35
  contour2D, 3
  contour2D (2D image and contour plots), 6
  contour3D, 43
  contour3D (3-D contours), 20
  createisosurf (3-D volume visualisation), 40
  createvoxel (3-D volume visualisation), 40
  dev.interactive, 8
  getplist (Composite plots), 55
  gg.col, 3
  gg.col (Colors), 51
  gg2.col (Colors), 51
  hist3D, 2
  hist3D (3-D perspectives), 25
  Hypsometry, 3, 30, 67
  Hypsometry (2-D data set), 4

80
INDEX

image, 2, 3, 6–9
image2D, 3, 4, 24, 27, 63
image2D (2D image and contour plots), 6
image3D (images in 3D frame), 60
ImageOcean, 3, 9
ImageOcean (2-D data set), 4
images in 3D frame, 60
isosurf3D, 2
isosurf3D (3-D volume visualisation), 40
jet.col, 3, 8, 9, 15, 21, 26, 28, 35–37, 41, 42, 55, 69
jet.col (Colors), 51
jet2.col (Colors), 51
lines2D (Scatter plots), 68
lines3D, 2
lines3D (Scatter plots), 68
mesh, 3, 70
mesh (Mesh generation), 64
Mesh generation, 64

Oxsat, 3, 9, 44
Oxsat (3-D data set), 23
par, 8
persp, 2, 8, 15, 16, 21, 22, 25–30, 35–37, 41, 43, 55, 56, 61, 62, 65, 66, 69, 70, 78
persp3D, 2, 9, 16, 21, 36, 42, 62, 64, 67, 69
persp3D (3-D perspectives), 25
perspbox, 16, 21, 22, 28, 36, 43, 55, 62, 69, 70
perspbox (Perspective box), 65
Perspective box, 65
plot.plist (Composite plots), 55
plot3D (plot3D-package), 2
plot3D-package, 2
plotdev, 16, 22, 28, 30, 36, 37, 43, 44, 62, 66, 70
plotdev (Composite plots), 55
points2D (Scatter plots), 68
points3D, 2
points3D (Scatter plots), 68
polygon, 8, 9, 16, 56
polygon2D, 3
polygon2D (3-D arrows, segments, polygons, boxes, rectangles), 13
polygon3D, 3
polygon3D (3-D arrows, segments, polygons, boxes, rectangles), 13
ramp.col, 3
ramp.col (Colors), 51
rasterImage, 8, 9
rect2D, 3
rect2D (3-D arrows, segments, polygons, boxes, rectangles), 13
rect3D, 3
rect3D (3-D arrows, segments, polygons, boxes, rectangles), 13
ribbon3D, 2
ribbon3D (3-D perspectives), 25
Scatter plots, 68
collect2D (Scatter plots), 68
collect3D (Scatter plots), 68
segments, 17
segments2D, 3
segments2D (3-D arrows, segments, polygons, boxes, rectangles), 13
segments3D, 3
segments3D (3-D arrows, segments, polygons, boxes, rectangles), 13
selectpplist (Composite plots), 55
setpplist (Composite plots), 55
slice3D, 2, 64, 78
slice3D (3-D volume visualisation), 40
slicecont3D, 2
slicecont3D (3-D volume visualisation), 40
spheresurf3D, 3
spheresurf3D (3-D surfaces), 34
surf3D, 3, 62–64, 67, 78
surf3D (3-D surfaces), 34
text2D, 3
text2D (Scatter plots), 68
text3D, 3, 30
text3D (Scatter plots), 68
trans3D, 16, 22, 29, 37, 43, 56, 62, 66, 70, 78
trans3d, 78
triangle3D (3-D volume visualisation), 40

voxel3D, 3, 16, 21, 36, 62, 69
voxel3D (3-D volume visualisation), 40