Package ‘plot3Drgl’

May 26, 2021

Version 1.0.2
Title Plotting Multi-Dimensional Data - Using ‘rgl’
Author Karline Soetaert <karline.soetaert@nioz.nl>
Maintainer Karline Soetaert <karline.soetaert@nioz.nl>
Depends rgl, plot3D, R (>= 3.2.3)
Imports grDevices
Description The ‘rgl’ implementation of plot3D functions.
License GPL (>= 3.0)
NeedsCompilation no
Repository CRAN
Date/Publication 2021-05-26 11:50:03 UTC

R topics documented:

plot3Drgl-package ...................................................... 1
cutrgl ................................................................. 2
image2Drgl ............................................................ 3
persp3Drgl ............................................................. 7
plotrgl ................................................................. 10

Index 14

plot3Drgl-package Functions to plot multi-dimensional data using rgl.

Description

Package plot3Drgl provides an interface from package plot3D to package rgl.
It will plot most (but not all) features from plots generated with plot3D, except for the color keys
and polygons.
It also also includes rgl implementations of 2-D functions (arrows, points, contours, images), which
can be zoomed, moved, and sections selected.
Author(s)
Karline Soetaert

See Also
Functions from R-package plot3D.
Functions from R-package rgl.

• For 3-D graphs:
  – plotrgl the main function that translates plot3D graphs to rgl.

• For 2-D graphs:
  – arrows2Drgl creates 2-D arrow plots.
  – scatter2Drgl for point and line plots.
  – image2Drgl, contour2Drgl an rgl implementation of the image and contour functions.

Apart from the usual zooming, it is also possible to move the figure in the rgl window (based on an example in the rgl package).
Rectangular areas can be selected from rgl plots, using cutrgl while uncutrgl will restore the original plot.

cutrgl

Cutting a rectangular region from an rgl plot.

Description
cutrgl zooms in on a selected region of the plot. It overwrites the current plot. Selection is done by dragging over the plot, with the left mousekey clicked.
croprgl zooms in on a region of the plot defined by the axes limits. It overwrites the current plot.
uncutrgl and uncroprgl restore the original plot, but keep the current orientation.

Usage
cutrgl (...)  
croprgl (xlim = NULL, ylim = NULL, zlim = NULL, ...)  
uncutrgl (...)  
uncroprgl(...)  

Arguments
xlim, ylim, zlim  
The limits of the plot.

...  
Any argument that can be passed to the function plotrgl, e.g argument lighting, or to the rgl functions par3d, open3d or material3d. Exceptions are new and add (which are always FALSE).
### Value

Returns the updated plotting list. See `plotdev`.

### Note

Both functions will not work when another active `rgl` window has been opened. In that case, `cutrgl` will freeze R, and the escape key should be used.

### Author(s)

Karline Soetaert &lt;karline.soetaert@nioz.nl&gt;

### See Also

`plotrgl`,

`material3d`, `par3d` for `rgl` arguments that can be passed to the function.

### Examples

```r
## Not run:
ribbon3D(z = volcano, zlim = c(-100, 200), image = TRUE, plot = FALSE)
plotrgl(new = TRUE) # new window
cutrgl() # select region with left mouse
cutrgl() # second selection
uncutrgl() # original restored

## End(Not run)
```

---

**image2Drgl**

2-D image, contour, scatterplots, segments and arrows in `rgl`.

### Description

`image2Drgl` plots an image in `rgl`.

`contour2Drgl` creates a contourplot in `rgl`.

`scatter2Drgl` creates a scatterplot (`lineplot`, `points`, ...) in `rgl`.

`points2Drgl` is shorthand for `scatter2Drgl(..., type = "p")`

`lines2Drgl` is shorthand for `scatter2Drgl(..., type = "l")`

`arrows2Drgl` and `segments3D` plot arrows and segments in `rgl`.

`text2Drgl` plots labels in `rgl`.

These functions were implemented for their side effect that `rgl` plots can be zoomed, translocated, rectangular selections taken.
Usage

image2Drgl (z, x = seq(0, 1, length.out = nrow(z)),
y = seq(0, 1, length.out = ncol(z)), ..., 
col = NULL, NAcol = "white", breaks = NULL, border = NA,
facets = TRUE, colkey = NULL, contour = FALSE,
smooth = FALSE, clim = NULL, clab = NULL, shade = NA,
inttype = 1, dz = 0, add = FALSE)

contour2Drgl (z, x = seq(0, 1, length.out = nrow(z)),
y = seq(0, 1, length.out = ncol(z)), ..., 
col = NULL, colkey = NULL, clim = NULL, clab = NULL, dz = 0.1,
add = FALSE)

scatter2Drgl (x, y, colvar = NULL, ..., 
col = NULL, NAcol = "white", breaks = NULL, colkey = NULL,
clim = NULL, clab = NULL, CI = NULL, dz = 0.1, add = FALSE)

text2Drgl (x, y, labels, colvar = NULL, ..., 
col = NULL, NAcol = "white", breaks = NULL, colkey = NULL,
clim = NULL, clab = NULL, dz = 0.1, add = FALSE)

arrows2Drgl (x0, y0, x1, y1, colvar = NULL, ..., 
col = NULL, NAcol = "white", breaks = NULL, colkey = NULL,
clim = NULL, clab = NULL, type = "simple", dz = 0.1, add = FALSE)

segments2Drgl (x0, y0, x1, y1, colvar = NULL, ..., 
col = NULL, NAcol = "white", breaks = NULL, colkey = NULL,
clim = NULL, clab = NULL, dz = 0.1, add = FALSE)

rect2Drgl (x0, y0, x1, y1, colvar = NULL, ..., 
col = NULL, NAcol = "white", breaks = NULL, colkey = NULL,
clim = NULL, clab = NULL, dz = 0.1, add = FALSE)

lines2Drgl(x, y, ...)

points2Drgl(x, y, ...)

Arguments

x, y
Vectors with the x- and y-values.

z
The variable used for coloring the image plot, or containing the values to be plotted for the contour plot.

x0, y0
coordinates of points from which to draw the arrows.

x1, y1
coordinates of points to which to draw the arrows. At least one must be supplied.

colvar
The variable used for coloring the scatter plot or the arrows. If NULL, then col will be used as such.
labels
The text to be written. A vector of length equal to length of x, y.
col
Color palette to be used for the z or colvar variable. If colvar is NULL, then
the colors are used as such.
NAcol
Color to be used for NA values; 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.
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.
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. Creates the illusion of perspective. See persp.
contour
If TRUE, then a contour plot will be added to the image plot. Also allowed is to
pass a list with arguments for the contour2D function.
smooth
Logical, specifying whether Gouraud shading (smooth) or flat shading should
be used. (if TRUE then function cutrgl will not work).
colkey
A logical, NULL (default), or a list with parameters for the color key (legend).
Not all arguments from the original colkey function from plot3D are supported.
For instance, color keys will always be put on the 4th margin.
clim
Only if colvar is specified, the range of the color variable used. Values of
colvar that extend the range will be put to NA.
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.
inttype
The interpolation type to create the polygons, either interpolating the z (inttype
= 1 or the x,y values (inttype = 2) - see persp3D.
CI
A list with parameters and values for the confidence intervals or NULL. If a
list it should contain at least the item x or y. Other parameters should be one
of (with defaults): alen = 0.01,lty = par("lty"),lwd = 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 of the scatter points are used.
type
The type of the arrow head, one of "simple" (the default, which uses R-function
arrows) or "triangle".
dz
The 'layer depth'. The z-position is defined as 1 + dz.
add
Logical. If TRUE, then the image, contour or points will be added to the current
plot. If FALSE a new plot is started.
... additional arguments passed to the plotting methods. The following persp argu-
ments can be specified: xlim,ylim,zlim,xlab,ylab,zlab,main,sub,r,d,scale,expand,box,axes,
In addition, the perspbox arguments col.axis,col.panel,lwd.panel,col.grid,lwd.grid
can also be given a value.
Also the arguments lty,lwd can be specified.
The arguments after ... must be matched exactly.
Details

The first step in 2D rgl plotting consists in calling a 3-D function from package plot3D with argument plot set to FALSE.

image2Drgl and contour2Drgl call the image3D and contour3D functions of R-package plot3D, with colvar equal to z. Functions scatter2Drgl and arrows2Drgl call scatter3D and arrows3D. The z value argument to the 3-D functions is set equal to 1 + dz; For contour3D, scatter3D and arrows3D, it is by default equal to 1.1, while for image3D it is 1. This way, contours, points, segments and arrows will be drawn on top of the image.

The next step is to create a 3-D rgl plot, by calling plotrgl. After that, the viewing arguments are set equal to view3d(phi = 0, fov = 0), i.e. the plot is viewed at from the top.

The actions of the mouse on the plots is to zoom (left, middle), and to move it (right key).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

image3D, contour3D, scatter3D, segments3D, text3D and arrows3D on which the functions are based.

image2D, contour2D, scatter2D, segments2D, text2D, arrows2D for plot3D’s functions, to plot in ordinary R graphics.

cutrgl for cutting a rectangular region from the 2D plot.

Examples

```r
## =======================================================================
## image and points
## =======================================================================
image2Drgl(z = volcano, contour = TRUE, main = "volcano")
scatter2Drgl(x = seq(0, 1, by = 0.2), y = seq(0, 1, by = 0.2), cex = 3, add = TRUE)

## Not run:
cutrgl() # select a rectangle
uncutrgl()
## End(Not run)

## =======================================================================
## scatter points, and lines
## =======================================================================
scatter2Drgl(cars[,1], cars[,2], xlab = "speed", ylab = "dist")

## Not run:
cutrgl()
## End(Not run)
```
lc <- lowess(cars)
scatter2Drgl(lc$x, lc$y, type = "l", add = TRUE, lwd = 4)
## Not run:
cutrgl()
uncutrgl()
## End(Not run)

## confidence intervals

x <- sort(rnorm(10))
y <- runif(10)
cv <- sqrt(x^2 + y^2)
CI <- list(lwd = 2)
CI$x <- matrix (nrow = length(x), ncol = 2, data = rep(0.125, 2*length(x)))
scatter2D(x, y, colvar = cv, pch = 16, cex = 2, CI = CI)
scatter2Drgl(x, y, colvar = cv, cex = 2, CI = CI)

## arrows

arrows2Drgl(x0 = 100*runif(30), y0 = runif(30), x1 = 100*runif(30),
y1 = runif(30), length = 0.1*runif(30), col = 1:30, angle = 15:45,
type = c("simple", "triangle"), lwd = 2)
x0 <- 1:30
x1 <- 2:31
arrows2Drgl(x0 = x0, y0 = sin(x0), x1 = x1, y1 = sin(x1),
colvar = x1, lwd = 2)

persp3Drgl

3-D plotting functions using rgl.

Description

Functions persp3Drgl, ribbon3Drgl, hist3Drgl produce perspective plots using rgl; they are similar to functions persp3D, ribbon3D, hist3D from package plot3D.

Functions scatter3Drgl, points3Drgl, lines3D, segments3Drgl produce scatter plots and line plots using rgl; they are similar to functions scatter3D, points3D, lines3D, segments3D from package plot3D.

Functions slice3Drgl, points3Drgl, isosurf3Drgl, voxel3Drgl can visualise volumetric (3D) data using rgl; they are similar to functions slice3D, slicecont3D, isosurf3D, voxel3D from package plot3D.

Functions surf3Drgl, spheresurf3Drgl produce surface plots using rgl; they are similar to functions surf3D, spheresurf3D from package plot3D.
Functions `box3Drgl`, `border3Drgl`, `rect3Drgl`, `text3Drgl` produce boxes, rectangles, texts to 3D plots using rgl; they are similar to functions `box3D`, `border3D`, `rect3D`, `text3Drgl` from package `plot3D`.

Usage

```r
persp3Drgl(...) ribbon3Drgl(...) hist3Drgl(...) scatter3Drgl(...) points3Drgl(...) lines3Drgl(...) slice3Drgl(...) slicecont3Drgl(...) isosurf3Drgl(...) voxel3Drgl(...) triangle3Drgl(...) surf3Drgl(...) spheresurf3Drgl(...) segments3Drgl(...) image3Drgl(...) contour3Drgl(...) box3Drgl(...) border3Drgl(...) rect3Drgl(...) text3Drgl(...)```

Arguments

... arguments passed to the plotting methods of package `plot3D`, or to the `plotrgl` method. The following `persp` arguments can be specified: `xlim`, `ylim`, `zlim`, `xlab`, `ylab`, `zlab`, `main`, `sub`, `r`, `d`, `s`, `scale`, `expand`, `box`, `axes`, `nticks`, `ticktype`, `shade`, `ltheta`, `lphi`.

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

Details

The first step in 3D rgl plotting consists in calling the corresponding 3-D function from package `plot3D` with argument `plot` set to `FALSE`.

The next step is to create a 3-D rgl plot, by calling `plotrgl`.

The actions of the mouse on the plots is to zoom (left, middle), and to move it (right key).

Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

`plotdev` to plot first in ordinary R graphics and then in rgl

for `plotrgl` to plot first in ordinary R graphics and then in rgl
cutrgl for cutting a rectangular region from the rgl plot.

Examples

```r
## perspective plots
persp3Drgl(z = volcano, contour = list(side = "zmax"))

# ribbon, in x--direction
V <- volcano[seq(1, nrow(volcano), by = 5),
             seq(1, ncol(volcano), by = 5)] # lower resolution
ribbon3Drgl(z = V, ticktype = "detailed")
hist3Drgl(z = V, col = "grey", border = "black", lighting = TRUE)

## Not run:
cutrgl() # select a rectangle
uncutrgl()

## End(Not run)

## scatter points
with(quakes, scatter3Drgl(x = long, y = lat, z = -depth,
colvar = mag, cex = 3))
plotdev() # plots same on ordinary device...

## 3D surface
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)

# alpha makes colors transparent
surf3Drgl(x, y, z, colvar = z, border = "black", smooth = TRUE,
           alpha = 0.2)

## volumetric data
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)

slice3Drgl(x, y, z, colvar = p, col = jet.col(alpha = 0.5),
xs = 0, ys = c(-4, 0, 4), zs = NULL, d = 2)

---

**plot3d**  
*Plot 3D graphs in rgl window.*

**Description**

plot3d plots objects created with functions from package plot3d in an rgl window.

**Usage**

```r
plot3d (lighting = FALSE, new = TRUE, add = FALSE, smooth = FALSE, ...)
```

**Arguments**

- `lighting` Logical, when `TRUE` will add light. Default is with lighting toggled off; this is similar to shading
- `new` Logical, when `TRUE` will open a new window. When `FALSE` will start a new plot in the same window. Is overruled (to `FALSE`) if `add` is `TRUE`.
- `add` Logical, when `TRUE` will add to the current plot.
- `smooth` Logical, specifying whether Gouraud shading (smooth) or flat shading should be used. See `material3d` from R-package rgl. (note: if `TRUE` then `cutrgl` will not work). This only affects images.
- `...` Any argument to the rgl functions, e.g arguments from `par3d`, `open3d` or `material3d`.

**Note**

Arrows are best reproduced with argument `type` from the `arrows3D` function (package plot3d) set equal to "cone", although this does not always work well (there is probably a flaw in how it is implemented). Another option is to use `type = "triangle"`, which simply maps the arrows on the xy-plane, ignoring the z-axis. In this case, only a view from above (phi = 0) will produce symmetric arrowheads (when `scale = TRUE` and `expand = 1`).

The translation of `scatter3D` ignores the `pch` argument but displays all symbols as squares (if `pch = .`) or as filled circles.

The color key is not (cannot be) plotted in rgl.

In rgl, both `lty` and `lwd` have to be one number. For `lwd`, this has been overruled, i.e. line widths can be a vector. It is still not possible to use different line types in one type of object.

The actions of the mouse on the plots is to rotate (left), to move (middle), and to zoom it (right).
Author(s)

Karline Soetaert <karline.soetaert@nioz.nl>

See Also

material3d, par3d for rgl arguments that can be passed to the function.
A similar function, plotdev, from package plot3D plots the 3D graphs to the current device (other than a rgl-device).

Any function of package plot3D: see e.g. help files of persp3D, scatter3D, arrows3D, slice3D, surf3D.

Direct rgl functions, see persp3Drgl, scatter3Drgl, etc....

Examples

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

### Composite image and contour in 3D
### plot reduced resolution (for speed) volcano to traditional window:
VV <- volcano[seq(1, nrow(volcano), by = 3), seq(1, ncol(volcano), by = 3)]
persp3D(z = VV, contour = list(side = "zmax"))

plotrgl(new = TRUE) # new window

# add light, smooth surface change x-axis limits
plotrgl(new = FALSE, lighting = TRUE,
xlim = c(0.2, 0.8), smooth = TRUE)

# same:
# persp3Drgl(z = volcano, contour = list(side = "zmax"),
# lighting = TRUE, xlim = c(0.2, 0.8), smooth = TRUE)

### scatters with fitted surface and droplines (see ?scatter3D)
###
par (mfrow = c(1, 1))
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)

plot3D(xy, pch = 19, col = "red")

# fitted surface
z <- predict(fit, newdata = xy)

persp3D(z = z, xlab = "wt", ylab = "disp", zlab = "mpg")

# add droplines
droplines3D(x = wt, y = disp, z = mpg, pch = 21)
})
mpg.pred <- matrix(nrow = 30, ncol = 30,
data = predict(fit, newdata = data.frame(xy),
iterval = "prediction")[,1])

# 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,
facets = NA, fit = fitpoints),
main = "mtcars")

})

plotrgl()

## =======================================================================
## scatter3D with text
## =======================================================================

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",
labels = rownames(USArrests), cex = 0.8,
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))

plotrgl()

## =======================================================================
## spheresurf3D
## =======================================================================

AA <- Hypsometry$z

# log transformation of color variable; full = TRUE to plot both halves
spheresurf3D(AA, theta = 90, phi = 30, box = FALSE,
full = TRUE, plot = FALSE)

# change the way the left mouse reacts
plotrgl(mouseMode = c("zAxis", "zoom", "fov"))

## =======================================================================
## Arrows - has a flaw
```r
## =======================================================================
z <- seq(0, 2*pi, length.out = 100)
x <- cos(z)
y <- sin(z)

z0 <- z[seq(1, by = 10, length.out = 10)]
z1 <- z[seq(9, by = 10, length.out = 10)]

# cone arrow heads
arrows3D(x0 = 10*cos(z0), y0 = sin(z0), z0 = z0,
x1 = 10*cos(z1), y1 = sin(z1), z1 = z1,
type = "cone", length = 0.4, lwd = 4,
angle = 20, col = 1:10, plot = FALSE)

plotrgl(lighting = TRUE)

## 2D plot

## 2D plot

image2D(z = volcano)
plotrgl()

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

* **hplot**
  cutrgl, 2
  image2Drgl, 3
  persp3Drgl, 7
  plotrgl, 10

* **package**
  plot3Drgl-package, 1

arrows, 5
arrows2D, 6
arrows2Drgl, 2
arrows2Drgl(image2Drgl), 3
arrows3D, 6, 10, 11

border3D, 8
border3Drgl(persp3Drgl), 7
box3D, 8
box3Drgl(persp3Drgl), 7

colkey, 5
contour, 5
contour2D, 5, 6
contour2Drgl, 2
contour2Drgl(image2Drgl), 3
contour3D, 6
contour3Drgl(persp3Drgl), 7
croprgl(cutrgl), 2
cutrgl, 2, 2, 5, 6, 9, 10

hist3D, 7
hist3Drgl(persp3Drgl), 7

image2D, 6
image2Drgl, 2, 3
image3D, 6
image3Drgl(persp3Drgl), 7
isosurf3D, 7
isosurf3Drgl(persp3Drgl), 7

lines2Drgl(image2Drgl), 3
lines3D, 7
lines3Drgl(persp3Drgl), 7

material3d, 2, 3, 10, 11
open3d, 2, 10

par3d, 2, 3, 10, 11
persp, 5, 8
persp3D, 5, 7, 11
persp3Drgl, 7, 11
perspbox, 5, 8
plot3Drgl(plot3Drgl-package), 1
plot3Drgl-package, 1
plotdev, 3, 8, 11
plotrgl, 2, 3, 6, 8, 10
points2Drgl(image2Drgl), 3
points3D, 7
points3Drgl(persp3Drgl), 7

rect2Drgl(image2Drgl), 3
rect3D, 8
rect3Drgl(persp3Drgl), 7
ribbon3D, 7
ribbon3Drgl(persp3Drgl), 7

scatter2D, 6
scatter2Drgl, 2
scatter2Drgl(image2Drgl), 3
scatter3D, 6, 7, 10, 11
scatter3Drgl(persp3Drgl), 7
segments2D, 6
segments2Drgl(image2Drgl), 3
segments3D, 6, 7
segments3Drgl(persp3Drgl), 7
slice3D, 7, 11
slice3Drgl(persp3Drgl), 7
slicecont3D, 7
slicecont3Drgl(persp3Drgl), 7
spheresurf3D, 7
spheresurf3Drgl(persp3Drgl), 7
surf3D, 7, 11
surf3Drgl (persp3Drgl), 7

text2D, 6
text2Drgl (image2Drgl), 3
text3D, 6
text3Drgl, 8
text3Drgl (persp3Drgl), 7
triangle3Drgl (persp3Drgl), 7

uncroprgl (cutrgl), 2
uncutrgl, 2
uncutrgl (cutrgl), 2

voxel3D, 7
voxel3Drgl (persp3Drgl), 7