plot3Drgl : Tools for plotting 3-D and 2-D data in openGL.

Karline Soetaert
NIOZ-Yerseke
The Netherlands

Abstract

R package **plot3Drgl** (Soetaert 2014b) contains functions for plotting multi-dimensional data in openGl, based on functions as in **plot3D** (Soetaert 2013).

A related package that depends on **plot3Drgl**is **OceanView** (Soetaert 2014a) which contains functions for visualing oceanographic data.

Keywords: plot, persp, image, 2-D, 3-D, scatter plots, surface plots, slice plots, openGL, R .

1. Introduction

The R package **plot3D** (Soetaert 2013) provides functions for plotting 2- and 3-D data. Package **plot3Drgl** allows to plot these functions also in openGL, as made available by package **rgl** (Adler and Murdoch 2013).

One possibility is to first create a plot in base R-graphics, and then use function `plotrgl` to depict the same figure in rgl.

The main advantage of rgl over base graphics is that it allows to interactively rotate, zoom, and shift the graphics, and even select regions. However, in contrast to the base R plot functions in **plot3D**, it is not possible to plot a colorkey.

2. Function plotrgl

Typically I start by making a 3D plot using functions from package **plot3D**. Although not necessary, plotting can be postponed by setting argument `plot = FALSE`

```
persp3D(z = volcano, plot = FALSE)
```

The figure is then plotting in openGL by function `plotrgl`, whose arguments are:

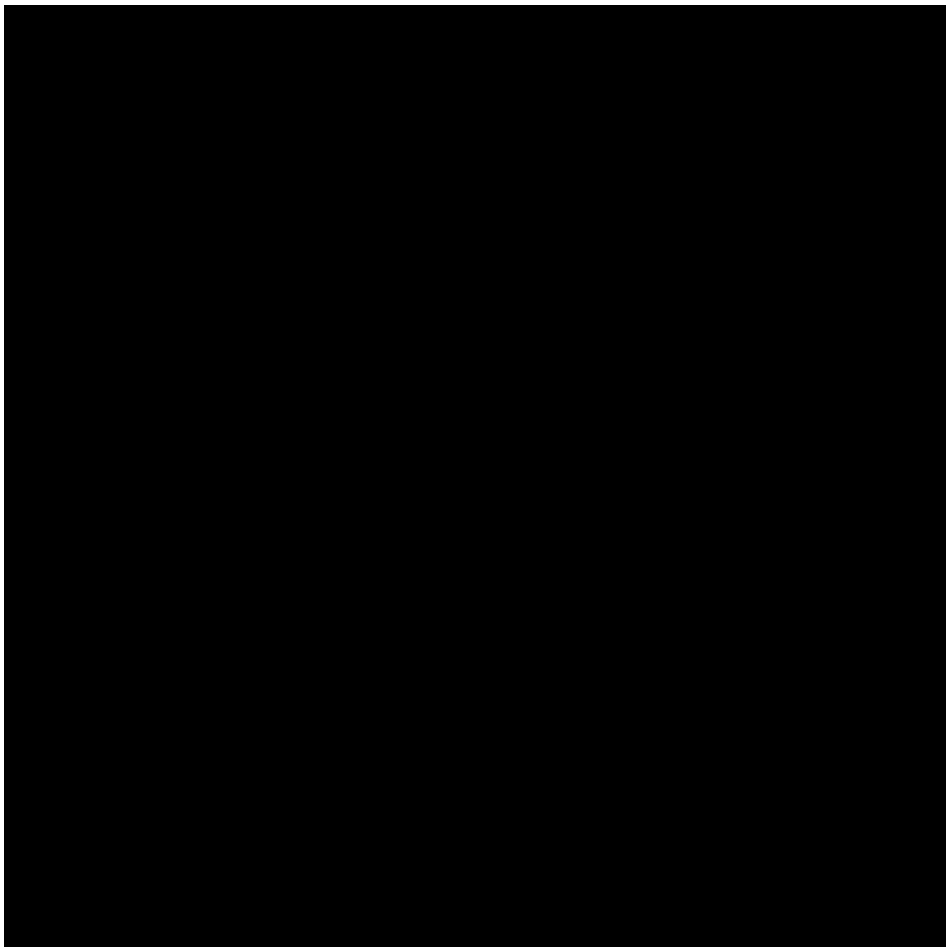
```
args(plotrgl)
```

```
function (lighting = FALSE, new = TRUE, add = FALSE, smooth = FALSE,
    ...)
NULL
```

Here the `...` are any parameter that would be passed to **rgl** functions `par3d`, `open3d` or `material3d`.

Argument `smooth` adds Gouraud shading, while `lighting` adds a light source.

```
plotrgl(smooth = TRUE, lighting = TRUE)
```



Now you can use the left mouse key to rotate the plot, the middle mouse key to move it, and the right key to zoom. You may also want to try function `cutrgl`, which allows to cut parts of the plot.

An alternative, shorter version to do the same is:

```
persp3Drgl(z = volcano, smooth = TRUE, lighting = TRUE)
```

Function `croprgl` can be used to adapt the ranges (not shown)

```
cutrgl()                      # requires selection using left mouse
croprgl(xlim = c(0.2, 0.8))
uncutrgl()                    # restores original plot
```

The same figure in base R-graphics looks less nice but has a colorkey:

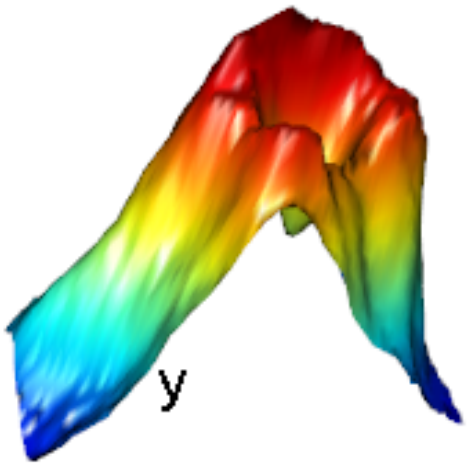


Figure 1: The volcano, after a region has been cutted

```
plotdev(shade = 0.1)
```

3. scatter plot example

A linear regression of the mtcars data can be easily plotted both in base graphics and using rgl:

```
attach(mtcars)
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")[,1])
# fitted points for droplines to surface
fitpoints <- predict(fit)

scatter3D(z = mpg, x = wt, y = disp, colvar = abs(mpg - fitpoints),
       pch = 18, cex = 2, theta = 20, phi = 20, ticktype = "detailed",
```

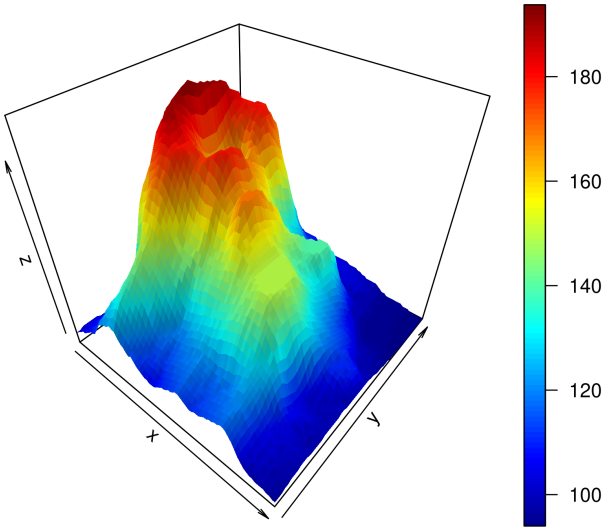
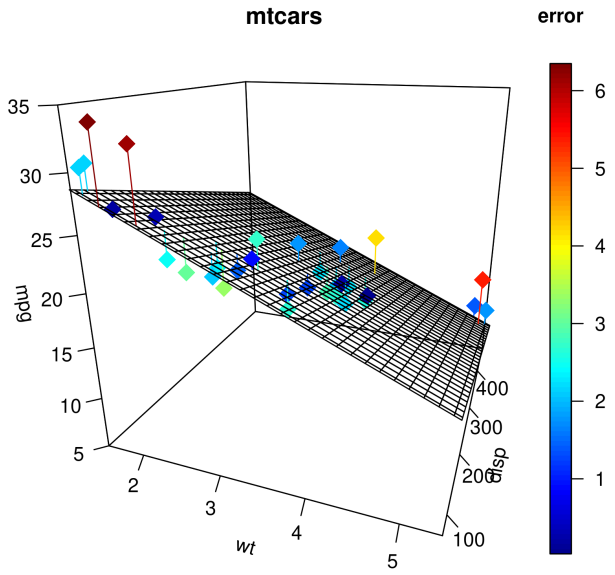


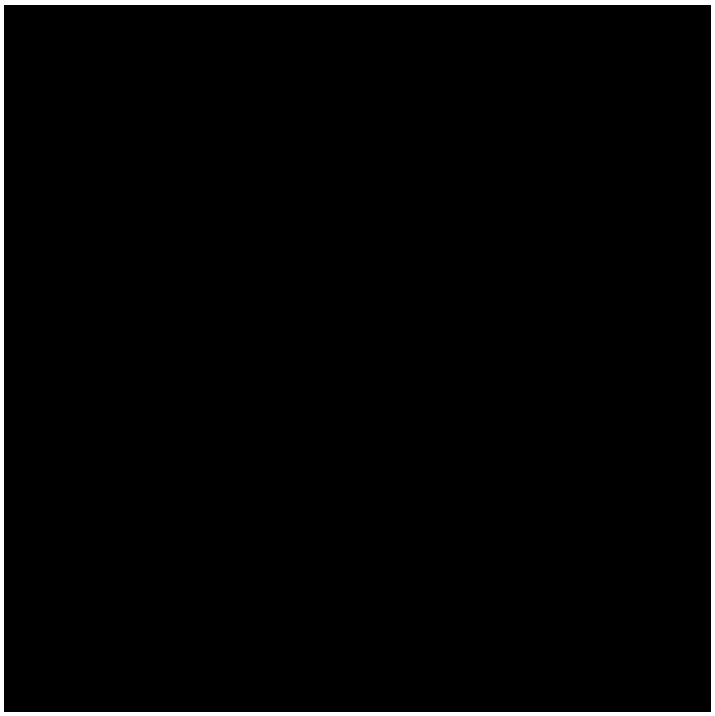
Figure 2: The volcano, using base R graphics

```
xlab = "wt", ylab = "disp", zlab = "mpg", main = "mtcars",
clab = "error", zlim = c(5, 35),
surf = list(x = wt.pred, y = disp.pred, z = mpg.pred,
            facets = NA, border = "black", fit = fitpoints)
)
```

```
detach(mtcars)
```

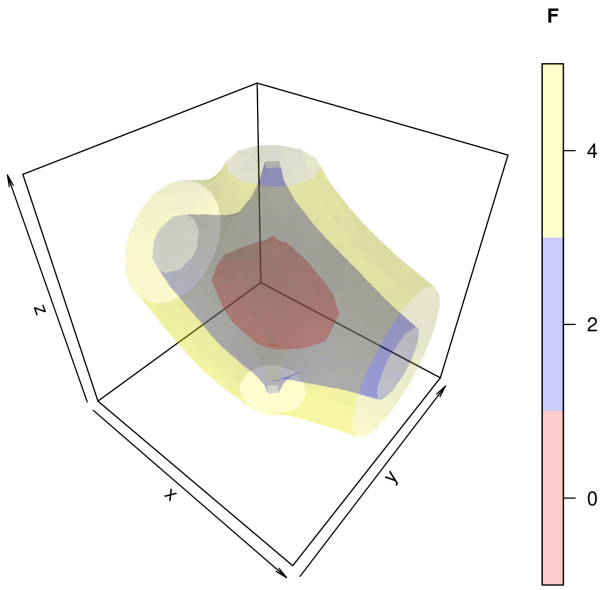
```
plotrgl(new = FALSE)
```





4. isosurfaces

Function `isosurf3D` from **plot3D** creates surfaces of equal scalar value from a volumetric data



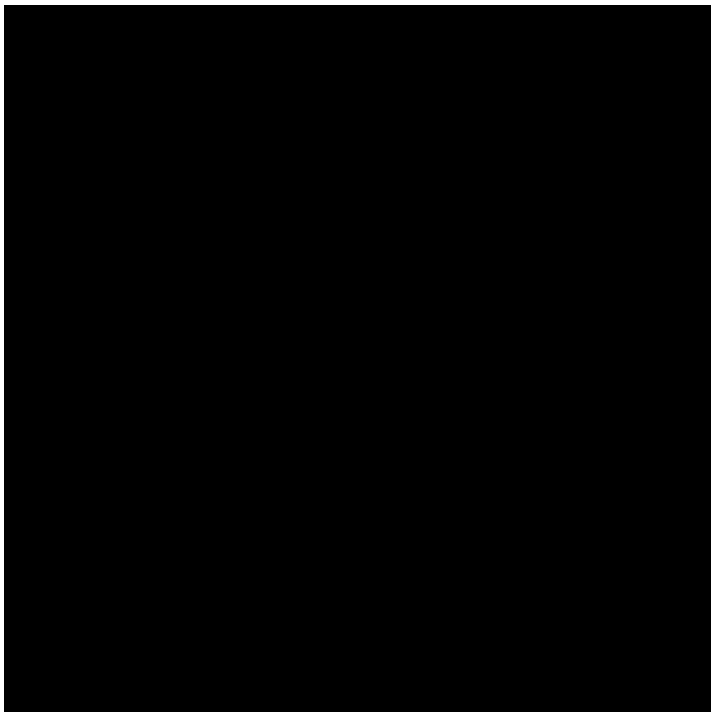
set. It makes use of a function from package **misc3d** (Feng and Tierney 2008).

If we depict several isosurfaces, it is best to use transparent colors by setting argument `alpha` smaller than 1. Plotting transparent surfaces is very slow in base graphics, but not so in openGL.

```
 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))
# 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, plot = FALSE)
```

```
plotdev()
```

```
plotrgl(new = FALSE, lighting = TRUE)
```



5. Issues

- Sometimes the axes are not drawn in `rgl` plots. If you want axes, just type

  ```
  decorate3d()
  ```

- The package contains a function to visualise arrows in rgl as cones. But it has a flaw, as the arrows are distorted, if not perpendicular to the z-axis. Use with care

6. Finally

This vignette was made with Sweave (Leisch 2002).

References

Adler D, Murdoch D (2013). *rgl: 3D visualization device system (OpenGL)*. R package version 0.93.945, URL http://CRAN.R-project.org/package=rgl.

Feng D, Tierney L (2008). "Computing and Displaying Isosurfaces in R." *Journal of Statistical Software*, **28**(1). URL http://www.jstatsoft.org/v28/i01/.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), *Compstat 2002 - Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL http://www.stat.uni-muenchen.de/~leisch/Sweave.

Soetaert K (2013). *plot3D: Plotting multi-dimensional data.* R package version 1.0.

Soetaert K (2014a). *OceanView: Visualisation of Oceanographic Data and Model Output.* R package version 1.0.

Soetaert K (2014b). *plot3Drgl: Plotting multi-dimensional data - using rgl.* R package version 1.0.

Affiliation:

Karline Soetaert
Royal Netherlands Institute of Sea Research (NIOZ)
4401 NT Yerseke, Netherlands
E-mail: karline.soetaert@nioz.nl
URL: http://http://www.nioz.nl/