plot3d : Tools for plotting 3-D and 2-D data.

Karline Soetaert
NIOZ-Yerseke
The Netherlands

Abstract

R package **plot3d** (Soetaert) contains functions for plotting 3-dimensional data. Many functions are derived from the `persp` function, other functions start from the `image` function.

This vignette is mainly a copy-paste from part of the help files. Many more examples can be found in the help of each function.

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

1. Introduction

R package **plot3d** provides functions for plotting 2- and 3-D data. It contains functions that are either extensions of R's `persp` function or or R's `image` function.

The main extensions to these functions are:

- In addition to the x, y (and z) values, there is a color variable (`colvar`) which defines the colors. [1].
- A color key (`colkey`) can be written next to the figure. It is possible to log-transform the color key.
- The resolution of a figure can be increased (`resfac`).
- Either the `facets` can be colored, just the border, or both.

A different color scheme than the one used by `image` is used by default. They are the 'matlab' colors, called `jet.col` here.

Package **plot3d** contains:

- Functions that are based on the `persp` function:
  - `Persp`, for an extended version of the `persp` function.
  - `ribbon3d`, for a perspective plot as ribbons.
  - `scatter3d`, scatter plots in 3-D shapes (points).

[1] For `Image` this is called 'z', for consistency with R's `image` function.

  - `hist3d`, for plotting 3-D histograms.
  - `surf3d`, for plotting 3-D shapes (or surfaces).
  - `arrow3d`, for plotting arrows in 3D.
  - `slice3d`, for plotting slices from a full 3-D data set.

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

- Colors and colorkey:
  - `colkey`, for adding a color legend.
  - `jet.col`, `ramp.col`, for suitable colors.

- Utility functions:
  - `mesh` for generating rectangular (2D) or (3D) meshes.

- Data sets:
  - `Oxsat` is a (rather large) 3-D data set with the ocean's oxygen saturation values.
  - `Hypsometry` is a 2-D data set with the worlds elevation and the ocean's depth.

This vignette contains some examples from the help-files. Many more examples can be found in the help files.

2. Functions derived from persp

2.1. Persp

`Persp` is an extension of R's `persp` function, while `perspbox` simply draws an empty perspective box. Their arguments are (see the help file for what they mean):

```
 args(Persp)
```

```
function (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", border = NA, facets = TRUE,
    colkey = list(side = 4), resfac = 1, trans = NULL, image = FALSE,
    contour = FALSE, panel.first = NULL, clim = NULL, clab = NULL,
    bty = "b")
NULL
```

```
 args(Perspbox)
```

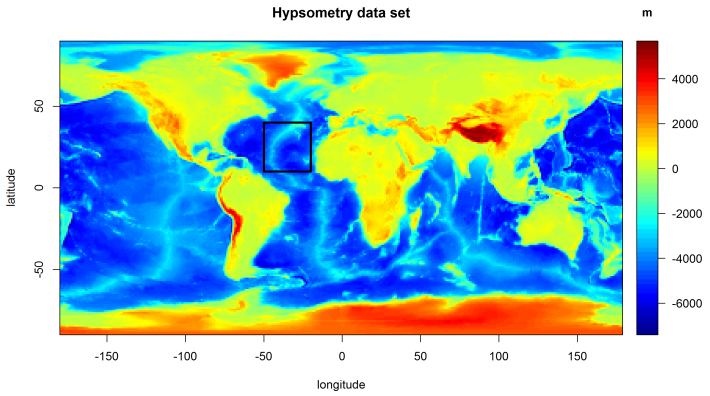

Figure 1: Hypsometry data set

```
function (x = seq(0, 1, length.out = nrow(z)), y = seq(0, 1,
    length.out = ncol(z)), z, bty = c("f", "b", "b2", "g", "bl",
    "xy", "u"), ..., col.axis = "black", col.panel = NULL, lwd.panel = 1,
    col.grid = NULL, lwd.grid = 1, phi = 40, theta = 40, colkey = list(side = 4))
NULL
```

The example from data set `Hypsometry` is used to demonstrate the potential of `Persp` and `Perspbox`.

A part of the `Hypsometry` data set is depicted.

We first plot the data, with the zoomed part.

```
Image(Hypsometry, xlab = "longitude", ylab = "latitude",
       main = "Hypsometry data set", clab = "m")
rect(-50, 10, -20, 40, lwd = 3)
```

```
  ii <- which(Hypsometry$x > -50 & Hypsometry$x < -20)
  jj <- which(Hypsometry$y >  10 & Hypsometry$y <  40)
  Ocean <- Hypsometry$z[ii, jj]
  zlim <- c(-10000, 0)
```

The figure is made with black side-panels.

```
  par(mfrow = c(1, 1), mar = c(1, 1, 1, 1))
 # Actual bathymetry, 4 times increased resolution, with contours
  Persp(z = Hypsometry$z[ii,jj], xlab = "longitude", bty = "bl",
        ylab = "latitude", zlab = "depth", clab = "depth, m",
        expand = 0.5, d = 2, phi = 20, theta = 30, resfac = 4,
        contour = list(col = "grey"), zlim = zlim,

        colkey = list(side = 1, length = 0.5, dist = -0.1))
```

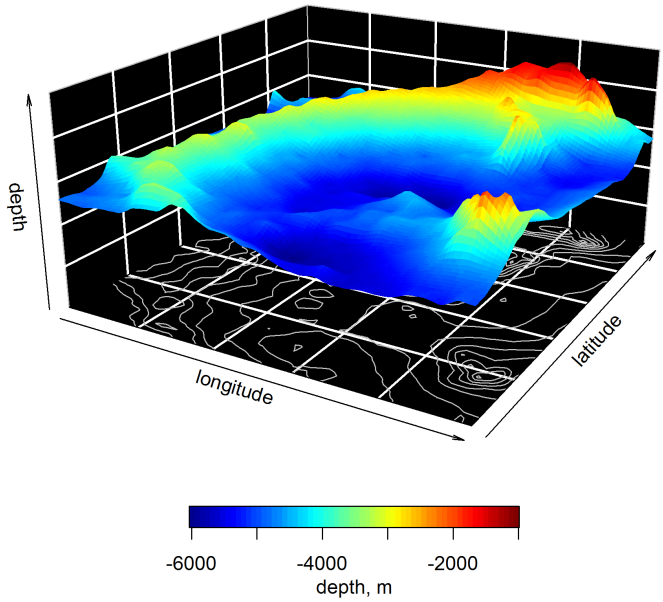

Figure 2: Bathymetry - complex

2.2. slice3d

`slice3d` draws slices from volumetric (3D) data.

```
 args(slice3d)
```

```
function (x, y, z, colvar, ..., phi = 40, theta = 40, xs = min(x),
    ys = max(y), zs = min(z), col = NULL, NAcol = "white", border = NA,
    facets = TRUE, colkey = list(side = 4), trans = NULL, panel.first = NULL,
    clim = NULL, clab = NULL, bty = "b")
NULL
```

Function `mesh3d` is used to generate a full rectangular 3-D mesh. This is used to generate the volumetric data that defines the color. This is visualised by one slice in x and 3 slices in y direction

```
 par(mfrow = c(1, 1))
 x <- y <- z <- seq(-4, 4, by = 0.1)
 M <- mesh(x, y, z)
 R <- with (M, sqrt(x^2 + y^2 +z^2))
 p <- sin(2*R)/(R+1e-3)
 slice3d(x, y, z, colvar = p, edge = FALSE,
          xs = 0, ys = c(-4, 0, 4), zs = NULL, d = 2)
```

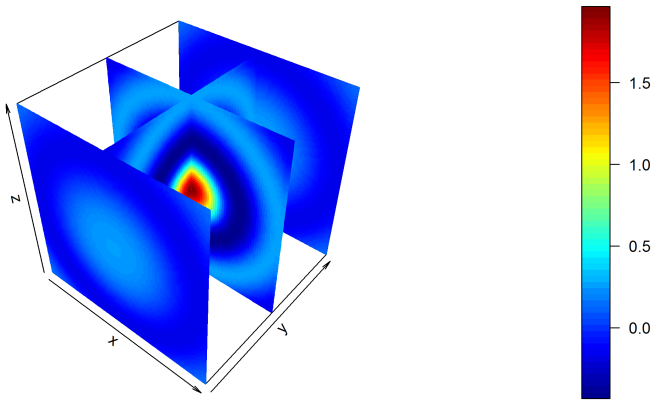

Figure 3: Slices from volumetric data

2.3. surf3d

`surf3d` creates 3-D surface plots.

```
 args(surf3d)
```

```
function (x, y, z, colvar = z, ..., phi = 40, theta = 40, col = NULL,
    NAcol = "white", border = NA, facets = TRUE, colkey = list(side = 4),
    trans = NULL, panel.first = NULL, clim = NULL, clab = NULL,
    bty = "b")
NULL
```

Here are 4 applications, showing the different options.

```
  par(mfrow = c(2, 2), mar = c(0, 0, 0, 0))
 # Shape 1

  M  <- mesh(seq(0,  6*pi, length.out = 100),
             seq(pi/3, pi, length.out = 100))
  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 = 100),
             seq(0, 2*pi, length.out = 100))
  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)
```

2.4. scatter and scatter3d

Functions `scatter` and `scatter3d` draw scatterplots.

```
 args(scatter)
```

```
function (x, y, colvar = NULL, ..., col = NULL, NAcol = "white",
    colkey = list(side = 4), add = FALSE, clim = NULL, clab = NULL)
NULL
```

```
 args(scatter3d)
```

```
function (x, y, z, colvar = z, ..., phi = 40, theta = 40, col = NULL,
    NAcol = "white", colkey = list(side = 4), trans = NULL, panel.first = NULL,
    clim = NULL, clab = NULL)
NULL
```

We first plot the dataset `quakes`, and then create a mathematical data set.

```
 par(mfrow = c(1, 2))
 # the quakes data set
  # before the scatters are drawn,
  # add small dots on basal plane and on the depth plane
```

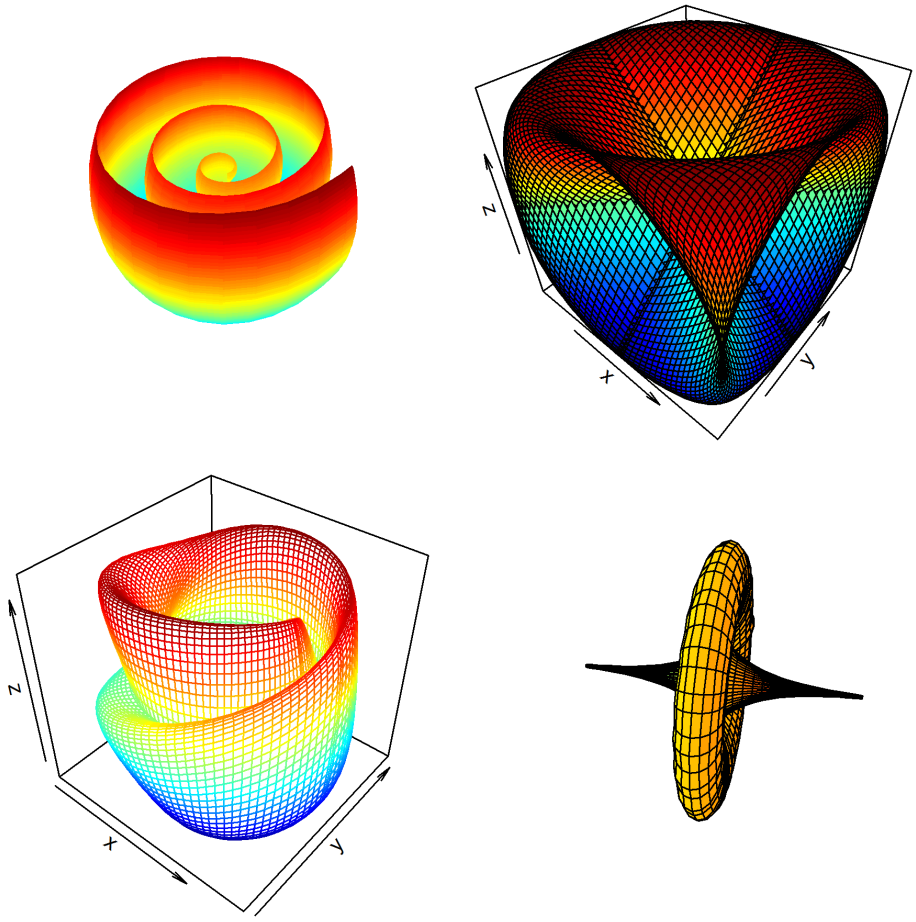

Figure 4: Surface plots

```
  panelfirst <- function(trans) {
     zmin <- min(-quakes$depth)
     XY <- trans3d(quakes$long, quakes$lat,
                   z = rep(zmin, nrow(quakes)), pmat = trans)
     scatter(XY$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$depth, pmat = trans)
     scatter(XY$x, XY$y, colvar = quakes$mag, pch = ".",
             cex = 2, add = TRUE, colkey = FALSE)
   }
  trans <-
    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))
        )
# a full grid 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)
  scatter3d(x, y, z, colvar = z, pch = ".", cex = 2,
            theta = 10, d = 2, colkey = FALSE, main = "A sphere")
```

2.5. arrow3d

```
arrow3d(u, v, w, x = NULL, y = NULL, z = NULL, colvar = NULL,
         ..., scale = 1, arr.max = 0.2, arr.min = 0,
         by = NULL, phi = 30, theta = 30,
         col = NULL, NAcol = "white",
         colkey = list(side = 4), trans = NULL,
         clim = NULL, clab = NULL)
```

The `arrow3d` function is also based on the `persp` function:

```
# 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)
```

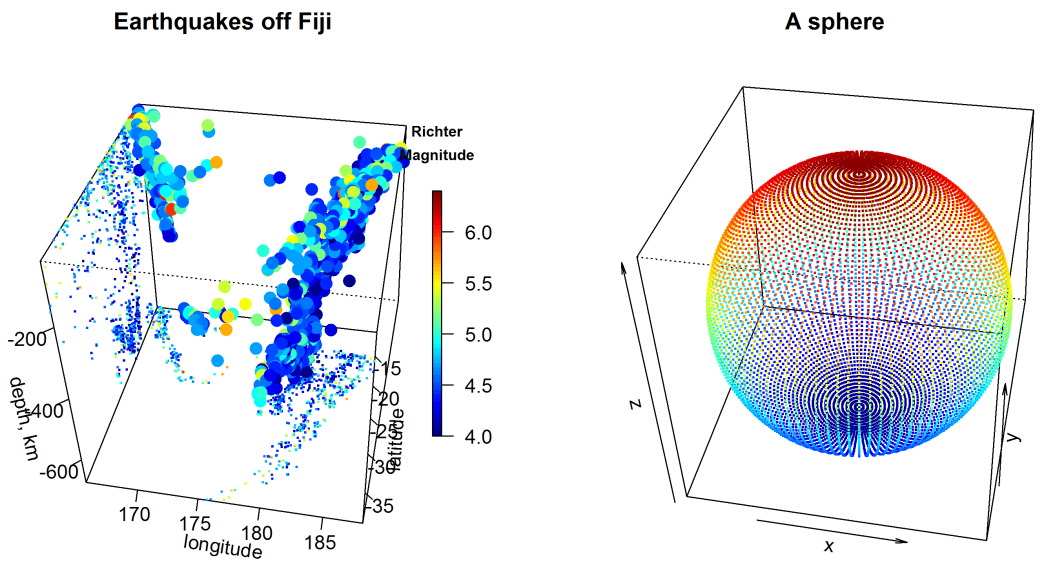

Figure 5: Scatterplot

```
M <- mesh(xx, yy, zz)
x <- M$x; y <- M$y; z <- M$z
# Calculate velocity values at each (x, y, z)
u <- sin(pi*x) * cos(pi*y) * cos(pi*z)
v <- -cos(pi*x) * sin(pi*y) * cos(pi*z)
w <- sqrt(2/3) * cos(pi*x) * cos(pi*y) * sin(pi*z)
length(xx)
```

```
[1] 9
```

```
trans <- arrow3d(u, v, w, x, y, z, colvar = z, clab = "z-value",
                 d = 2, col = c("red", "blue","green"))
# add starting points of the arrows - use R-function trans3d
 points(trans3d(x, y, z, pmat = trans),
        col = c(rep("red", 81), rep("blue", 81), rep("green", 81)))
```

3. More complex coloring

```
Theta <- seq(0, 2*pi, length.out = 359)
Phi   <- seq(0,   pi, length.out = 180)
M <- mesh(Theta, Phi)
```

```
[1] 9
```

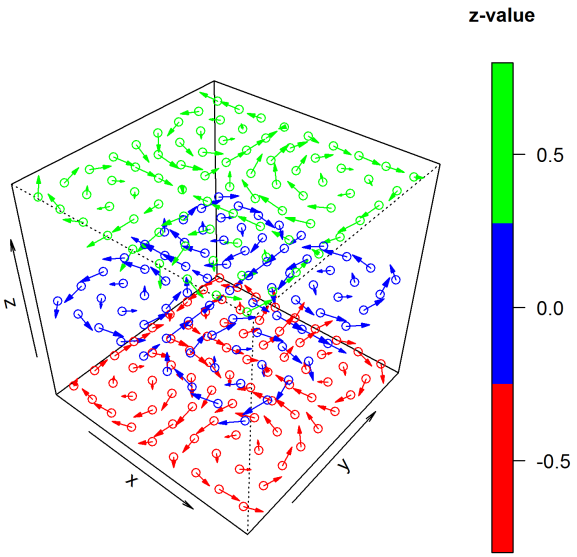

Figure 6: arrow3d function

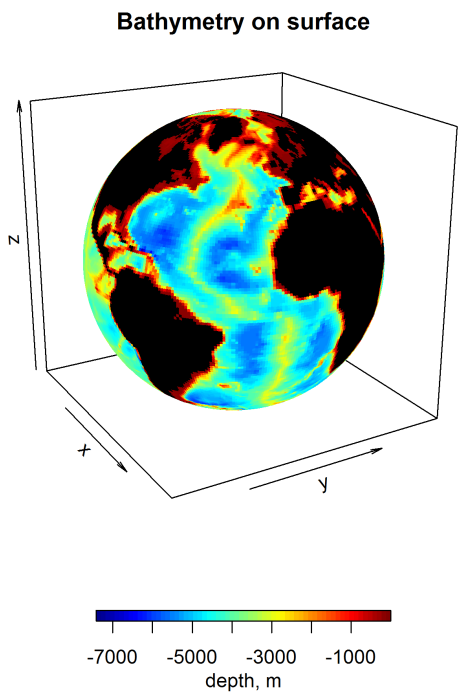

Figure 7: The ocean baythmetry, draped on a spherical surface

```
theta <- M$x; phi <- M$y
x <- cos(theta)*sin(phi)
y <- sin(theta)*sin(phi)
z <- cos(phi)
# mask the land
BB <- Hypsometry$z; BB[BB>=0] <- NA
par(mfrow = c(1, 1))
par(mar = c(3, 2, 3, 3))
surf3d(-x, -y, -z, colvar = BB, phi = 20, theta = 60, d = 3,
         main = "Bathymetry on surface",
         colkey = list(side = 1, length = 0.5, width = 0.5),
         box = TRUE, NAcol = "black", clab = "depth, m")
```

4. Functions based on image

The `Image` function is an extended version of `image`. It has two S3 methods:

```
Image(z =, ...)

Image.matrix(z, x = NULL, y = NULL, ...,
             col = jet.col(100), NAcol = "white", facets = TRUE,
             contour = FALSE, colkey = list(side = 4), resfac = 1,
```

```
            clab = NULL, theta = 0, border = NA)
Image.array(z, margin = c(1, 2), subset, ask = NULL, ...)
```

The data set `Oxsat` has oxygen saturation values in the ocean, at 2dg horizontal resolution, and for 33 depth intervals.

```
names(Oxsat)
```

```
[1] "lon"   "lat"   "depth" "val"   "name"  "units"
```

```
dim(Oxsat$val)
```

```
[1] 180  90  33
```

We use `Image.array` to plot several depth intervals at once, looping over the first and second margin:

```
  Image(z = Oxsat$val, subset = 1:8,
        x = Oxsat$lon, y = Oxsat$lat,
        margin = c(1, 2), NAcol = "black", colkey = FALSE,
        xlab = "longitude", ylab = "latitude",
        main = paste("depth ", Oxsat$depth[1:9], " m"),
        zlim = c(0, 115), mfrow = c(3, 3))
colkey(clim = c(0, 115), clab = c("O2 saturation", "percent"))
```

5. Finally

This vignette was made with Sweave (Leisch 2002).

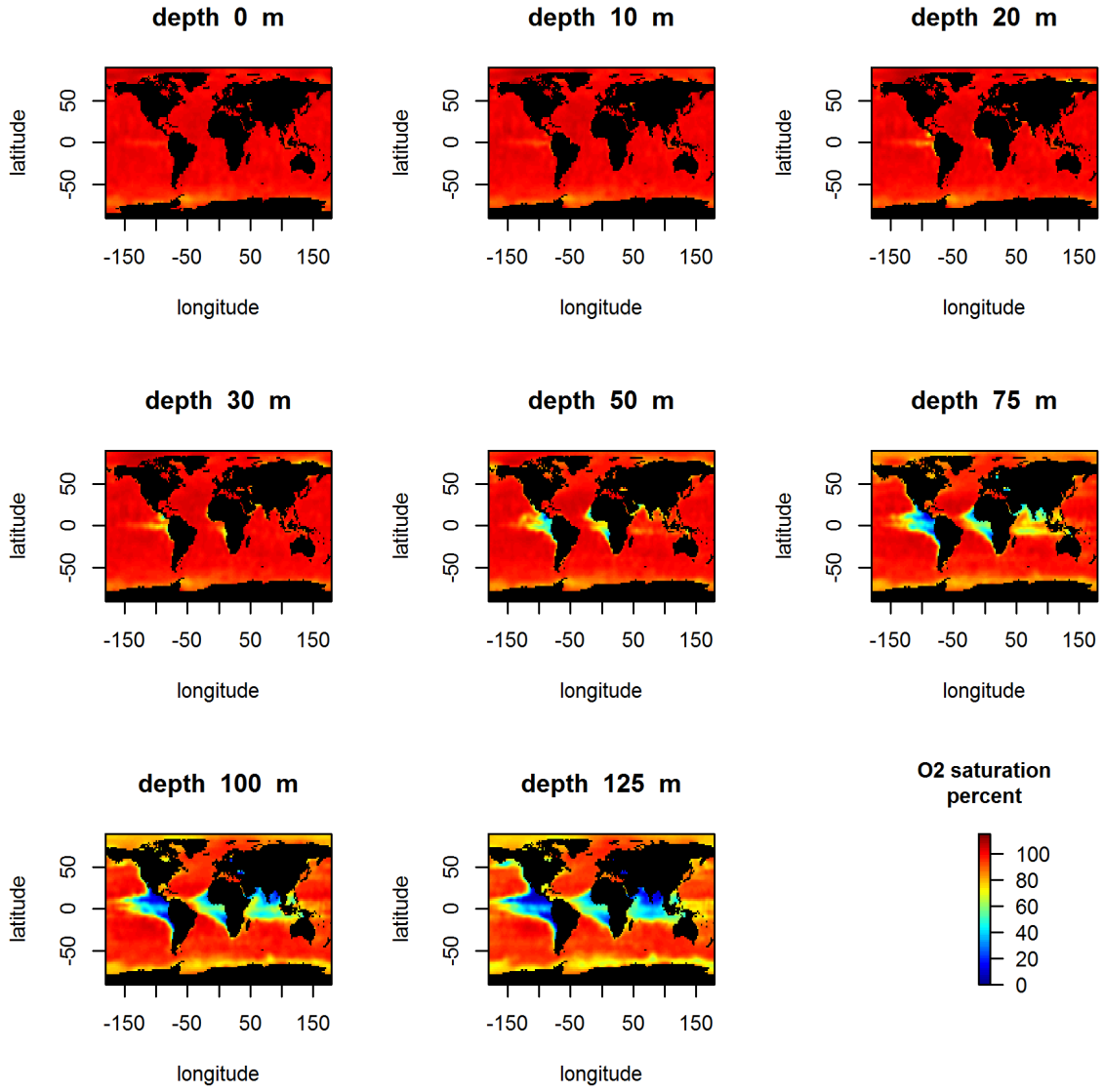

Figure 8: Image function

References

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 (????). *plot3d: Plotting multi-dimensional data.* 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/