Package ‘scatterplot3d’

May 5, 2023

Version  0.3-44
Date     2023-05-05
Title    3D Scatter Plot
Author   Uwe Ligges <ligges@statistik.tu-dortmund.de>, Martin Maechler, Sarah Schnackenberg
Maintainer Uwe Ligges <ligges@statistik.tu-dortmund.de>
Description Plots a three dimensional (3D) point cloud.
Depends  R (>= 2.7.0)
License  GPL-2
Encoding UTF-8
Imports  grDevices, graphics, stats
NeedsCompilation no
Repository CRAN
Date/Publication 2023-05-05 09:20:09 UTC

R topics documented:

  scatterplot3d .................................................. 1

Index

  scatterplot3d  3D Scatter Plot

Description

Plots a three dimensional (3D) point cloud.
Usage

scatterplot3d(x, y=NULL, z=NULL, color=par("col"), pch=par("pch"),
       main=NULL, sub=NULL, xlim=NULL, ylim=NULL, zlim=NULL,
       xlab=NULL, ylab=NULL, zlab=NULL, scale.y=1, angle=40,
       axis=TRUE, tick.marks=TRUE, label.tick.marks=TRUE,
       x.ticklabs=NULL, y.ticklabs=NULL, z.ticklabs=NULL,
       y.margin.add=0, y.axis.offset=1, grid=TRUE, box=TRUE,
       lab=par("lab"), lab.z=mean(lab[1:2]), type="p",
       highlight.3d=FALSE, mar=c(5,3,4,3)+0.1, bg=par("bg"),
       col.axis=par("col.axis"), col.grid="grey", col.lab=par("col.lab"),
       cex.symbols=par("cex"), cex.axis=0.8 * par("cex.axis"),
       cex.lab=par("cex.lab"), font.axis=par("font.axis"),
       font.lab=par("font.lab"), lty.axis=par("lty"),
       lty.grid=par("lty"), lty.hide=NULL, lty.hplot=par("lty"),
       log="", asp=NA, ...)

Arguments

x
  the coordinates of points in the plot.
y
  the y coordinates of points in the plot, optional if x is an appropriate structure.
z
  the z coordinates of points in the plot, optional if x is an appropriate structure.
color
  colors of points in the plot, optional if x is an appropriate structure. Will be ignored if highlight.3d = TRUE.
pch
  plotting "character", i.e. symbol to use.
main
  an overall title for the plot.
sub
  sub-title.
xlim, ylim, zlim
  the x, y and z limits (min, max) of the plot. Note that setting enlarged limits may not work as exactly as expected (a known but unfixed bug).
xlab, ylab, zlab
  titles for the x, y and z axis.
scale.y
  scale of y axis related to x- and z axis.
angle
  angle between x and y axis (Attention: result depends on scaling).
axis
  a logical value indicating whether axes should be drawn on the plot.
tick.marks
  a logical value indicating whether tick marks should be drawn on the plot (only if axis = TRUE).
label.tick.marks
  a logical value indicating whether tick marks should be labeled on the plot (only if axis = TRUE and tick.marks = TRUE).
x.ticklabs, y.ticklabs, z.ticklabs
  vector of tick mark labels.
y.margin.add
  add additional space between tick mark labels and axis label of the y axis
y.axis.offset
  a numeric (default 1) specifying the offset of y axis tick mark labels from the axis, see offset argument of text.
grid  a logical value indicating whether a grid should be drawn on the plot.
box   a logical value indicating whether a box should be drawn around the plot.
lab   a numerical vector of the form c(x, y, len). The values of x and y give the
(approximate) number of tickmarks on the x and y axes.
lab.z the same as lab, but for z axis.
type  character indicating the type of plot: "p" for points, "l" for lines, "h" for vertical
lines to x-y-plane, etc.
highlight.3d points will be drawn in different colors related to y coordinates (only if type = "p"
or type = "h", else color will be used). On some devices not all colors can be displayed. In this case try the postscript
device or use highlight.3d = FALSE.
mar   A numerical vector of the form c(bottom, left, top, right) which gives the lines
of margin to be specified on the four sides of the plot. See section Values on
how to change the setting back to the default / previous setting.
bg    background (fill) color for the open plot symbols given by pch = 21:25.
col.axis, col.grid, col.lab
the color to be used for axis / grid / axis labels.
cex.symbols the magnification to be used for point symbols.
cex.axis, cex.lab
the magnification to be used for axis annotation and labels relative to the current.
font.axis, font.lab
the font to be used for axis annotation / labels.
lty.axis, lty.grid
the line type to be used for axis / grid.
lty.hide line style used to plot ‘non-visible’ edges (defaults of the lty.axis style)
lty.hplot the line type to be used for vertical segments with type = "h".
log   Not yet implemented! A character string which contains "x" (if the x axis is to
be logarithmic), "y", "z", "xy", "xz", "yz", "xyz".
asp   numeric, giving the aspect ratio z/x or z/y, see ‘Note’.
... more graphical parameters can be given as arguments, pch = 16 or pch = 20 may
be nice.

Value

xyz.convert function which converts coordinates from 3D (x, y, z) to 2D-projection (x, y) of
scatterplot3d. Useful to plot objects into existing plot.
points3d function which draws points or lines into the existing plot.
plane3d function which draws a plane into the existing plot: plane3d(Intercept, x.coef = NULL, y.coef = NULL, lty = "dashed", lty.box = NULL, draw_lines = TRUE, draw_polygon = FALSE, polygon_args = list(border = NA, col = rgb(0,0,0,0.2)), ...). Instead of Intercept a vector containing 3 elements or an (g)lm object
can be specified. The argument lty.box allows to set a different line style
for the intersecting lines in the box’s walls. The arguments draw_lines and
The `draw_polygon` argument allows for choosing whether to represent the plane via line segments or as a solid surface, respectively. The list in `polygon_args` collects arguments to be passed to the underlying `polygon` call that draws a solid (or transparent) plane if `draw_polygon=TRUE`.

The `box3d` function which “refreshes” the box surrounding the plot.

The `contour3d` function which draws contour lines into the existing plot:

```r
contour3d(f, x.count = 10, y.count = 10, type = "l", lty = "24", x.resolution = 50, y.resolution = 50, ...).
```

The first argument can be an `lm` object of two dimensions or a function of two arguments. In both cases the dimensions have to be given in the order `x, y` of the `scatterplot3d` call. The arguments `x.count` and `y.count` specify how many segments should be drawn for each dimension. The arguments `x.resolution` and `y.resolution` control the number of locations where the segments have to be evaluated.

The `par.mar` argument:

As the function modifies the `par("mar")` settings of the current device and needs to keep these in case you add elements to the plot later on, you can change these back via `par(object$par.mar)` in case you want to add more plots with default margins to the current device.

**Note**

Some graphical parameters should only be set as arguments in `scatterplot3d` but not in a previous `par()` call. One of these is `mar`, which is also non-standard in another way: Users who want to extend an existing `scatterplot3d` graphic with another function than `points3d`, `plane3d` or `box3d`, should consider to set `par(mar = c(b, l, t, r))` to the value of `mar` used in `scatterplot3d`, which defaults to `c(5, 3, 4, 3) + 0.1`.

Other `par` arguments may be split into several arguments in `scatterplot3d`, e.g., for specifying the line type. And finally some of `par` arguments do not apply here, e.g., many of those for axis calculation. So we recommend to try the specification of graphical parameters at first as arguments in `scatterplot3d` and only if needed as arguments in previous `par()` call.

If `asp` is a finite positive value then the window is set up so that one data unit in the x or y direction (the one that is plotted horizontally - depends on `angle`) is equal in length to `asp × one data unit in the z direction`. The variation of `asp` is only reasonable if the default values `x.ticklabs=NULL, y.ticklabs=NULL, z.ticklabs=NULL` are not changed.

**Author(s)**

Uwe Ligges <ligges@statistik.tu-dortmund.de>, Martin Maechler, Sarah Schnackenberg

**References**


**See Also**

`persp`, `plot`, `par`. 
Examples

## On some devices not all colors can be displayed.
## Try the postscript device or use highlight.3d = FALSE.

### example 1
z <- seq(-10, 10, 0.01)
x <- cos(z)
y <- sin(z)
scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue",
              col.grid="lightblue", main="scatterplot3d - 1", pch=20, mar=c(0,0,0,0))

### example 2
temp <- seq(-pi, 0, length = 50)
x <- c(rep(1, 50) %*% t(cos(temp)))
y <- c(cos(temp) %*% t(sin(temp)))
z <- c(sin(temp) %*% t(sin(temp)))
scatterplot3d(x, y, z, highlight.3d=TRUE,
              col.axis="blue", col.grid="lightblue",
              main="scatterplot3d - 2", pch=20)

### example 3
temp <- seq(-pi, 0, length = 50)
x <- c(rep(1, 50) %*% t(cos(temp)))
y <- c(cos(temp) %*% t(sin(temp)))
z <- 10 * c(sin(temp) %*% t(sin(temp)))
color <- rep("green", length(x))
temp <- seq(-10, 10, 0.01)
x <- c(x, cos(temp))
y <- c(y, sin(temp))
z <- c(z, temp)
color <- c(color, rep("red", length(temp)))
scatterplot3d(x, y, z, color, pch=20, xlim=c(-2, 10),
              main="scatterplot3d - 3")

### example 4
my.mat <- matrix(runif(25), nrow=5)
dimnames(my.mat) <- list(LETTERS[1:5], letters[11:15])
my.mat # the matrix we want to plot ...
s3d.dat <- data.frame(cols=as.vector(col(my.mat)),
                      rows=as.vector(row(my.mat)),
                      value=as.vector(my.mat))
scatterplot3d(s3d.dat, type="h", lwd=5, pch=" ",
             x.ticklabs=colnames(my.mat), y.ticklabs=rownames(my.mat),
             color=grey(25:1/40), main="scatterplot3d - 4")

### example 5
data(trees)
s3d <- scatterplot3d(trees, type="h", highlight.3d=TRUE,
                     angle=55, scale.y=0.7, pch=16, main="scatterplot3d - 5")
# Now adding some points to the "scatterplot3d"
s3d$points3d(seq(10,20,2), seq(85,60,-5), seq(60,10,-10),
             pch=20, col="blue", lwd=3, col.grid="lightblue")
col="blue", type="h", pch=16)
# Now adding a regression plane to the "scatterplot3d"
attach(trees)
my.lm <- lm(Volume ~ Girth + Height)
s3d$plane3d(my.lm, lty.box = "solid")

## example 6; by Martin Maechler
cubedraw <- function(res3d, min = 0, max = 255, cex = 2, text. = FALSE)
{
  ## Purpose: Draw nice cube with corners
cube01 <- rbind(c(0,0,1), 0, c(1,0,0), c(1,1,0), 1, c(0,1,1), # < 6 outer
c(1,0,1), c(0,1,0)) # <- "inner": fore- & back-ground

cub <- min + (max-min)* cube01
## visible corners + lines:
res3d$points3d(cub[c(1:6,1,7,3,7,5) ,], cex = cex, type = 'b', lty = 1)
## hidden corner + lines
res3d$points3d(cub[c(2,8,4,8,6), ], cex = cex, type = 'b', lty = 3)
if(text.## debug
  text(res3d$xyz.convert(cub), labels=1:nrow(cub), col='tomato', cex=2)
}
## 6 a) The named colors in R, i.e. colors()
cc <- colors()
crgb <- t(col2rgb(cc))
par(xpd = TRUE)
rr <- scatterplot3d(crgb, color = cc, box = FALSE, angle = 24,
xlim = c(-50, 300), ylim = c(-50, 300), zlim = c(-50, 300))
cubedraw(rr)
## 6 b) The rainbow colors from rainbow(201)
rbc <- rainbow(201)
Rrb <- t(col2rgb(rbc))
rR <- scatterplot3d(Rrb, color = rbc, box = FALSE, angle = 24,
xlim = c(-50, 300), ylim = c(-50, 300), zlim = c(-50, 300))
cubedraw(rR)
rR$points3d(Rrb, col = rbc, pch = 16)
Index

* hplot
  scatterplot3d, 1

par, 4
persp, 4
plot, 4
polygon, 4
scatterplot3d, 1
text, 2