Package 'legendplot'

September 22, 2026

Title Standard and 'rgl' Plots with Legends

Version 0.4-1

Date 2026-09-11

Maintainer Ruben Fernandez-Casal <rubenfcasal@gmail.com>

Depends R (>= 2.14.0), graphics

Imports rgl, grDevices

Suggests fields, knitr

Description Provides tools to combine standard R plots or 'rgl' 3D plots with a legend. Facilitates the creation of composite figures that mix 2D or 3D visualizations with a categorical or continuous legend.

License GPL (>= 2)

URL https://rubenfcasal.github.io/legendplot/, https://github.com/rubenfcasal/legendplot/

BugReports https://github.com/rubenfcasal/legendplot/issues/

LazyData yes

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Ruben Fernandez-Casal [aut, cre] (ORCID: <https://orcid.org/0000-0002-5785-3739>)

Repository CRAN

Date/Publication 2026-09-21 22:20:02 UTC

Contents

`legendplot-package` *legendplot: Standard and 'rgl' Plots with Legends*

Description

This package provides tools to combine standard R plots or 'rgl' 3D plots with a legend, facilitating the creation of composite figures that mix 2D or 3D visualizations with a categorical or continuous legend. For more information visit https://rubenfcasal.github.io/legendplot/articles/legendplot.html.

Author(s)

Maintainer: Ruben Fernandez-Casal <rubenfcasal@gmail.com> (ORCID)

Authors:

- Ruben Fernandez-Casal <rubenfcasal@gmail.com> (ORCID)

See Also

Useful links:

- https://rubenfcasal.github.io/legendplot/
- https://github.com/rubenfcasal/legendplot/
- Report bugs at https://github.com/rubenfcasal/legendplot/issues/

`axis3` *3D axis with control over tick length, tick angle and label position*

Description

Modification of `rgl::axis3d()` that adds three extra parameters, `ticksize`, `labeldist` and `tickangle`, to independently control the length of the axis tick marks, the distance between those tick marks and their labels, and the direction (within the plane orthogonal to the axis) in which both the tick marks and the labels are offset (see *Details*). Based on the solution proposed on Stack Overflow (see *References*).

Usage

```
axis3(
  edge,
  at = NULL,
  labels = TRUE,
  tick = TRUE,
  line = TRUE,
  pos = NULL,
  nticks = 5,
  ticksize = 0.05,
  tickangle = 45,
  labeldist = 3,
  ...
)
```

Arguments

`edge`	character string indicating the edge of the bounding box on which the axis is drawn (same format as in `rgl::axis3d()`, e.g. `"z+-"`).
`at`	positions at which the tick marks are drawn; if `NULL` they are computed automatically with `pretty()`.
`labels`	tick labels: logical, indicating whether to automatically label the axes or be omitted, or a vector of labels.
`tick`	logical; if `TRUE` the axis tick marks are drawn.
`line`	logical; if `TRUE` the axis line is drawn.
`pos`	optional position of the axis (see `rgl::axis3d()`).
`nticks`	approximate number of tick marks when `at = NULL`.
`ticksize`	length of the tick marks, as a fraction of the distance to the center of the scene's range.
`tickangle`	direction, in degrees, of the tick marks and labels within the plane orthogonal to the axis (e.g. `0` = along the first perpendicular coordinate, `90` = along the second one, `45` = towards the corner of the bounding box).
`labeldist`	distance between the tick marks and the labels, as a multiple of `ticksize`.
`...`	additional arguments passed to `rgl::segments3d()` and `rgl::text3d()`.

Details

By default (`tickangle = 45`), tick marks and labels are offset simultaneously along both coordinates perpendicular to the axis, towards the corresponding corner of the bounding box. The original behavior of `axis3d()` can be reproduced by setting `tickangle = 45` and `ticksize = 0.05 * sqrt(2)` (the actual relative length of the tick marks).

Value

An `rglId` object (see `rgl::lowlevel()`) with the identifiers of the **rgl** elements added (line, tick marks and labels).

References

Dominic Woolf (2013, Mar 23). R rgl distance between axis ticks and tick labels, Stack Overflow, https://stackoverflow.com/a/39299740/4303451.

See Also

`splot3d()`, `fplot3d()`, `rgl::axis3d()`

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
set.seed(1)
points3d(rnorm(100), rnorm(100), rnorm(100))
box3d()
# Draw several z axis at z+- and z--, with longer tick marks and
# labels closer to them than usual
axis3('z+-', ticksize = 0.5, labeldist = 1)
axis3('z+-', ticksize = 0.5, labeldist = 1, tickangle = 0)
axis3('z--', ticksize = 0.5, labeldist = 1, tickangle = 90)
```

---

categorical-color *Utilities for plotting with a categorical legend*

Description

`hcld.colors()` and `hot.colors()` create a color table useful for coding qualitative information and `fcolor()` assigns colors to categorical (factor) values.

Usage

```
fcolor(f, col = hcld.colors(length(labels)), labels = levels(as.factor(f)))

hcld.colors(n, palette = "Dark 3", ...)

cat.colors(n)
```

Arguments

f	(factor, or vector coercible to factor) values to be converted to colors.
col	colors for each level. Defaults to hcld.colors().
labels	levels to use. Defaults to levels(as.factor(f)).
n	number of colors (>= 1) to be in the palette, or vector coercible to factor, in which case n = nlevels(as.factor(n))
palette	a valid palette name for hcl.colors() (one of hcl.pals()).
...	additional arguments passed to legend().

Value

fcolor() returns vector of colors, one per element of f. hcld.colors() and cat.colors() return a character vector of colors (similar to hcl.colors() or rainbow(); see rgb()).

---

continuous-color *Utilities for plotting with a continuous color scale*

Description

jet.colors and hot.colors create a color table useful for contiguous color scales and scolor assigns colors to a numerical vector.

Usage

```
scolor(s, col = jet.colors(128), slim = range(s, finite = TRUE))

jet.colors(n)

hot.colors(n, rev = TRUE)
```

Arguments

s	values to be converted to the color scale.
col	color table used to set up the color scale (see image for details).
slim	limits used to set up the color scale.
n	number of colors (>= 1) to be in the palette.
rev	logical; if TRUE, the palette is reversed (decreasing overall luminosity).

Details

scolor converts a real valued vector to a color scale. The range slim is divided into length(col) + 1 pieces of equal length. Values which fall outside the range of the scale are coded as NA.

jet.colors generates a rainbow style color table similar to the MATLAB (TM) jet color scheme. It may be appropriate to distinguish between values above and below a central value (e.g. between positive and negative values).

hot.colors generates a color table similar to the MATLAB (TM) hot color scheme (reversed by default). It may be appropriate to represent values ranging from 0 to some maximum level (e.g. density estimation). The default value rev = TRUE may be adequate to grayscale conversion.

Value

scolor, jet.colors and hot.colors return a character vector of colors.

See Also

heat.colors, terrain.colors, rgb.

---

fplot *Add a categorical legend to an standard R plot*

Description

fplot() is designed to combine a standard R plot with a categorical (factor) legend. Analogous to splot(), splits the plotting region into a main panel and a legend panel, and uses legend() to draw the labels with their colors and symbols.

Usage

```
fplot(
  labels,
  col = hcld.colors(length(labels)),
  type = c("box", "point", "line"),
  pch = 16,
  lty = 1,
  lwd = 2,
  cex = 1,
  border = col,
  pt.cex = cex * 1.5,
  seg.len = 1.5,
  horizontal = FALSE,
  legend.shrink = 1,
  legend.width = NULL,
  legend.mar = NULL,
  legend.lab = NULL,
  legend.x = "center",
```

```
  bigplot = NULL,
  smallplot = NULL,
  add = FALSE,
  ...
)
```

Arguments

`labels`	vector with the category labels.
`col`	colors associated with each level (same order as `labels`). Defaults to `hcld.colors()`.
`type`	type of symbols shown in the legend: `"box"` for filled color boxes (as in a classic factor-level legend), `"point"` for points, or `"line"` for line segments.
`pch`	plotting 'character' (symbol) used when `type = "point"`.
`lty, lwd`	line types and widths for lines appearing in the legend, when `type = "line"`.
`cex`	text/symbol size in the legend.
`border, pt.cex, seg.len`	additional `legend()` parameters, also used to estimate the required legend width/height. The default values are: `border = col`, `pt.cex = cex * 1.5` and `seg.len = 1.5`.
`horizontal`	logical; if `FALSE` (default) legend will appear on the right side. If `TRUE` the legend will be along the bottom.
`legend.shrink`	amount to shrink the size of legend relative to the full height or width of the plot.
`legend.width, legend.mar`	control the size and margin of the legend panel, as in `splot()`. If left as `NULL` (default), they are computed automatically following the same character-size logic that `legend()` itself uses internally.
`legend.lab`	legend title.
`legend.x`	legend location relative to the legend panel (argument `x` of `legend()`). Possible values are: `"center"` (default), `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"` or `"right"`.
`bigplot, smallplot`	plot coordinates for main and legend panels. If not passed these will be determined within the function.
`add`	logical; if `TRUE` the legend is just added to the existing plot (the graphical parameters are not changed).
`...`	additional arguments passed to `legend()`.

Value

`fplot()` invisibly returns a list with components: `bigplot`, `smallplot`, `old.par`, `col` and `labels` (`par(old.par)` will reset plot parameters to the values before entering the function).

Side Effects

The plotting region (`par("plt")`) may be changed after exiting, to make it possible to add more features to the plot. They can be restored using the `old.par` returned values or by calling function `par.reset()`.

See Also

legend(), fcolor(), hcld.colors(), cat.colors(), fpoints().

Examples

```
# Plot equivalent to fpoints():
f <- as.factor(mtcars$cyl)
res <- fplot(levels(f), col = cat.colors(f), type = "point",
             legend.lab = "cyl")
with(mtcars,
    plot(hp, qsec, col = fcolor(f, col = res$col),
         pch = 16, cex = 1.5, main = "Motor Trend Car Road Tests")
)
par(res$old.par) # restore graphical parameters
```

---

fplot3d *Add a categorical legend to an 'rgl' plot*

Description

Splits the active rgl device into two subscenes (the main plot and a legend) and draws, in the legend subscene, a colored categorical legend together with its corresponding labels.

Usage

```
fplot3d(
  labels,
  col = hcld.colors(length(labels)),
  type = c("s", "c", "p", "l"),
  legend.zoom = 0.6,
  legend.width = 0.2,
  legend.size = 4,
  legend.dim = 0.2,
  legend.lab = NULL,
  lab.dist = 2.5,
  lab.rev = FALSE,
  ...
)
```

Arguments

labels	vector with the legend labels (levels/categories). Defaults to seq_along(col).
col	vector of colors associated with each label (defaults to hcld.colors(length(labels))).
type	character; symbol/object used to represent each label in the legend: "s" for spheres (default), "c" for cubes, "p" for points, or "l" for (horizontal) line segments.

legend.zoom	zoom factor applied to the legend subscene/panel.
legend.width	relative size of the legend symbols (spheres, cubes or line segments), as a fraction of the corresponding dimension of the legend subscene.
legend.size	point size, if type = "p", or line width, if type = "l" (in pixels; see size and lwd graphical parameters in rgl::par3d(); ignored in all other cases).
legend.dim	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
legend.lab	legend title.
lab.dist	distance between the legend symbols and their text labels, as a multiple of the symbol size (legend.width).
lab.rev	logical; if TRUE the order of the levels in the legend is reversed (by default they are shown from top to bottom).
...	material properties (see material3d()) used to draw the legend symbols.

Details

Mouse rotation is disabled in the legend subscene. Furthermore, since rgl::layout3d(..., mouseMode = "replace") does not copy the user's mouse handlers (which are therefore disabled), the first one is set up for trackball with double click to reset view (via dbltrack3d()), and the second one (if any) for panning (via pan3d()).

Value

Invisibly returns the identifiers of the created subscenes (see rgl::layout3d()).

See Also

fpoints3d(), fshade3d(), splot3d().

Examples

```
library(rgl)
open3d()
# Plot equivalent to fpoints():
f <- as.factor(mtcars$cyl)
fplot3d(levels(f), legend.lab = "cyl")
with( mtcars, plot3d(hp, qsec, mpg, type = "s", col = fcolor(f)))

# Using cubes instead of spheres in the legend
fplot3d(levels(f), legend.lab = "cyl", type = "c")
with( mtcars, plot3d(hp, qsec, mpg, type = "p", col = fcolor(f)))
```

fpoints *Scatter plot with a categorical legend*

Description

A generic function that, by default, draws a scatter plot with points colored according to a factor f and (optionally) adds a categorical legend (fpoints.default() calls fplot() and plot.default(), or plot.xy() if add = TRUE).

Usage

```
fpoints(x, ...)

## Default S3 method:
fpoints(
  x,
  y = NULL,
  f,
  col = hcld.colors(f),
  type = "p",
  cex = 1.5,
  pch = 16,
  legend = TRUE,
  legend.type = c("point", "box", "line"),
  legend.pch = pch,
  legend.cex = 1,
  legend.lty = 1,
  legend.lwd = 2,
  border = col,
  pt.cex = cex,
  seg.len = 1.5,
  horizontal = FALSE,
  legend.shrink = 1,
  legend.width = NULL,
  legend.mar = NULL,
  legend.lab = NULL,
  legend.x = "center",
  bigplot = NULL,
  smallplot = NULL,
  add = FALSE,
  reset = TRUE,
  xlab = NULL,
  ylab = NULL,
  asp = NA,
  ...
)
```

Arguments

x	object used to select a method. In the default method, it provides the x coordinates for the plot (and optionally the y coordinates; any reasonable way of defining the coordinates is acceptable, see the function xy.coords() for details).
...	additional graphical parameters (to be passed to the main plot function or to fpoints.default(); e.g. xlim, ylim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par() function beforehand.
y	y coordinates. Alternatively, a single argument x can be provided.
f	(factor, or vector coercible to factor), with length equal to the number of points, giving the group of each point.
col	colors associated with each level of f Defaults to hcld.colors().
type	character indicating the type of plotting; actually any of the types as in plot.default().
cex	numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex").
pch	vector of plotting characters or symbols: see points().
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the scatter plot in one and the categorical legend in the other. If FALSE only the (coloured) scatter plot is drawn and the legend-related arguments are ignored (fplot() is not called).
legend.type	type of symbols shown in the legend: "box" for filled color boxes (as in a classic factor-level legend), "point" for points, or "line" for line segments (see fplot()).
legend.pch, legend.cex	plotting character and size used in the legend when legend.type = "point".
legend.lty, legend.lwd	line type and width used in the legend when legend.type = "line".
border, pt.cex, seg.len	additional legend() parameters, also used to estimate the required legend width/height. The default values are: border = col, pt.cex = cex * 1.5 and seg.len = 1.5.
horizontal	logical; if FALSE (default) legend will appear on the right side. If TRUE the legend will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend, defaults to a description of f.
legend.x	legend location relative to the legend panel (argument x of legend()). Possible values are: "center" (default), "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright" or "right".

bigplot	plot coordinates for main plot. If not passed, and legend is TRUE, these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed, and legend is TRUE, these will be determined within the function.
add	logical; if TRUE the scatter plot is just added to the existing plot (including the legend if legend = TRUE, although the graphical parameters are not modified).
reset	logical; if FALSE the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE (default) the plot parameters will be reset to the values before entering the function.
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.
asp	the y/x aspect ratio, see plot.window().

Value

fplot() invisibly returns a list with components: bigplot, smallplot, old.par, col and labels (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

If reset = FALSE, the plotting region (par("plt")) may be changed after exiting, to make it possible to add more features to the plot. They can be restored using the old.par returned values or by calling function par.reset().

See Also

fplot(), fcolor(), hcld.colors(), plot.default(), spoints().

Examples

```
with(mtcars,
    fpoints(hp, qsec, f = cyl, main = "Motor Trend Car Road Tests")
)
```

---

fpoints3d *3D scatter plot with a categorical legend*

Description

A generic function that, by default, draws a 3D scatter plot (rgl::plot3d()) with points colored according to the levels of a factor f, and (optionally) adds a categorical legend (via fplot3d()).

Usage

```
fpoints3d(x, ...)

## Default S3 method:
fpoints3d(
  x,
  y = NULL,
  z = NULL,
  f,
  col = hcld.colors(nlevels(f)),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  type = "p",
  legend = TRUE,
  legend.type = c("s", "c", "p", "l"),
  legend.zoom = 0.6,
  legend.width = 0.2,
  legend.size = 4,
  legend.dim = 0.2,
  legend.lab = NULL,
  lab.dist = 2.5,
  lab.rev = FALSE,
  add = FALSE,
  ...
)
```

Arguments

x	object used to select a method. In the default method, it provides the x coordinates for the plot (and optionally the y and z coordinates; any reasonable way of defining the coordinates is acceptable, see the function xyz.coords() for details).
...	additional arguments passed to rgl::plot3d() for the main scatter plot (e.g. size).
y, z	y and z point coordinates. Alternatively, a single argument x can be provided.
f	(factor, or vector coercible to factor), used to color the points.
col	vector of colors associated with each level of f (defaults to hcld.colors(nlevels(f))).
xlab, ylab, zlab	labels for the coordinates.
type	character indicating the type of item to plot (see rgl::plot3d(): "p" for points, "s" for spheres, "l" for lines, "h" for line segments from z = 0, and "n" for nothing).
legend	logical; if TRUE (default), the active rgl device is splitted into two subscenes, drawing the main plot in one and the categorical legend in the other (see fplot3d()). if FALSE only the (coloured) main plot is drawn and the arguments related to the legend are ignored (fplot3d() is not called).

legend.type	character; symbol/object used to represent each level in the legend (see type in fplot3d()): "s" for spheres (default), "c" for cubes, "p" for points, or "l" for (horizontal) line segments.
legend.zoom	zoom factor applied to the legend subscene/panel.
legend.width	relative size of the legend symbols (spheres, cubes or line segments), as a fraction of the corresponding dimension of the legend subscene.
legend.size	point size, if type = "p", or line width, if type = "l" (in pixels; see size and lwd graphical parameters in rgl::par3d(); ignored in all other cases).
legend.dim	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
legend.lab	label for the axis of the color legend, defaults to a description of f.
lab.dist	distance between the legend symbols and their text labels, as a multiple of the symbol size (legend.width).
lab.rev	logical; if TRUE the order of the levels in the legend is reversed (by default they are shown from top to bottom).
add	logical; if TRUE the scatter plot is just added to the existing plot (including the legend if legend = TRUE).

Value

Called for its side effect (draws the 3D scatter plot and, unless legend = FALSE, the legend on the active rgl device).

See Also

spoints3d(), fplot3d(), fcolor(), fshade3d(), rgl::plot3d().

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
with(mtcars, fpoints3d(hp, qsec, mpg, f = cyl, type = "s"))
```

---

fshade3d *Draw 3D mesh objects adding a categorical legend*

Description

Draws a triangular mesh (mesh3d) colored according to the levels of a factor f, and (optionally) adds a categorical legend (via fplot3d()).

Usage

```
fshade3d(
  x,
  f,
  meshColor = c("faces", "vertices"),
  col = hcld.colors(nlevels(f)),
  legend = TRUE,
  legend.type = c("s", "c", "p", "l"),
  legend.zoom = 0.6,
  legend.width = 0.2,
  legend.size = 4,
  legend.dim = 0.2,
  legend.lab = NULL,
  lab.dist = 2.5,
  lab.rev = FALSE,
  ...
)
```

Arguments

x	triangular mesh (mesh3d object, see rgl::mesh3d()).
f	factor used to color the mesh. How its values are interpreted depends on meshColor.
meshColor	determines how color is applied to the mesh: "faces" or "vertices" (see rgl::shade3d()).
col	vector of colors associated with each level of f (defaults to hcld.colors(nlevels(f))).
legend	logical; if TRUE (default), the active rgl device is splitted into two subscenes, drawing the 3D mesh object in one and the categorical legend in the other (see fplot3d()). if FALSE only the (coloured) mesh is drawn and the arguments related to the legend are ignored (fplot3d() is not called).
legend.type	character; symbol/object used to represent each level in the legend (see type in fplot3d()): "s" for spheres (default), "c" for cubes, "p" for points, or "l" for (horizontal) line segments.
legend.zoom	zoom factor applied to the legend subscene/panel.
legend.width	relative size of the legend symbols (spheres, cubes or line segments), as a fraction of the corresponding dimension of the legend subscene.
legend.size	point size, if type = "p", or line width, if type = "l" (in pixels; see size and lwd graphical parameters in rgl::par3d(); ignored in all other cases).
legend.dim	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
legend.lab	label for the axis of the color legend, defaults to a description of f.
lab.dist	distance between the legend symbols and their text labels, as a multiple of the symbol size (legend.width).
lab.rev	logical; if TRUE the order of the levels in the legend is reversed (by default they are shown from top to bottom).
...	additional arguments passed on to rgl::shade3d().

Value

Called for its side effect (draws the mesh and, unless `legend = FALSE`, the legend on the active rgl device); invisibly returns the object identifiers.

See Also

`fplot3d()`, `sshade3d()`, `hcld.colors()`.

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
z_tri <- vb2tri3d(volcanom, volcanom$vb[3, ])
fz_tri <- cut(z_tri, 5)
fshade3d(volcanom, fz_tri, legend.dim = 0.3, lab.rev = TRUE)
```

`mouse-actions` *Utilities for setting the 'rgl' mouse actions*

Description

Functions to set the mouse button actions for an `rgl` subscene (the current one on the active device by default).

`setmouse3d()` sets the 'rgl' built-in mouse button modes (via `rgl::par3d()`).

`pan3d()` configures the callback functions of a mouse button so that dragging it pans the subscene of an (opened) rgl device.

`dbltrack3d()` installs, on the given mouse button of an `rgl` (sub)scene, the standard "trackball" rotation behavior (as that of parameter `mouseMode` of `rgl::par3d()`), extended so that double-clicking resets the viewpoint to "default" values.

Usage

```
setmouse3d(
  none = "none",
  left = "trackball",
  right = "zoom",
  middle = "fov",
  wheel = "pull",
  dev = cur3d(),
  subscene = currentSubscene3d(dev)
)

pan3d(
  button,
  dev = cur3d(),
```

```
  subscene = currentSubscene3d(dev),
  message = FALSE
)

dbltrack3d(
  button = 1,
  max.interval = 0.4,
  capture.now = FALSE,
  message = FALSE,
  dev = cur3d(),
  subscene = currentSubscene3d(dev)
)
```

Arguments

none, left, right, middle, wheel	actions for the corresponding mouse button; see parameter mouseMode of rgl::par3d().
dev	rgl device on which the mouse actions are configured. Defaults to the active device (see rgl::cur3d()).
subscene	subscene on which the mouse actions are configured. Defaults to the device's active subscene (see rgl::currentSubscene3d()).
button	mouse button on which panning is configured. Use 1 for left, 2 for right, 3 for middle, 4 for wheel, and 0 to set this action when no button is pressed.
message	whether to print a message confirming the configured button, device and subscene.
max.interval	maximum time, in seconds, between two clicks for them to be treated as a double-click.
capture.now	logical; if FALSE (default) the "default" viewpoint is restored on double-click (see setviewpoint3d()). If TRUE the "reset" viewpoint is set to the values in effect when dbltrack3d() is called.

Details

pan3d() is based on the function of the same name included as an example in the documentation of rgl::rgl.setMouseCallbacks().

dbltrack3d() reimplements the standard trackball rotation (following the mouseCallbacks demo shipped with **rgl**, demo(mouseCallbacks, package = "rgl")) so that it can measure the time elapsed since the previous click; if this is below max.interval, the click is treated as a double-click and the view is reset instead of starting a new rotation.

Value

setmouse3d() invisibly returns the resulting mouseMode vector.

pan3d() is called for its side effect (registers the mouse callbacks); it optionally prints a message confirming the configured button, device and subscene.

dbltrack3d() is called for its side effect (registers the mouse callbacks); it optionally prints a message confirming the configured button, device and subscene.

See Also

new3d(), rgl::par3d(), rgl::rgl.setMouseCallbacks().

Examples

```
library(rgl)
open3d()
setmouse3d(middle = "zoom") # zoom with middle button or wheel
pan3d(2) # pan the subscene with right button
dbltrack3d(1) # rotate as usual with left button; double-click resets the view
shade3d(volcanom, col = "lightgreen")
```

---

new3d *New 'rgl' plot*

Description

Clear the current rgl device or open a new one (if there are no devices or if open = TRUE).

Usage

```
new3d(open = FALSE, clear = "all", ...)
```

Arguments

open	if TRUE a new rgl device is opened.
clear	rgl stack(s) to remove, see argument type in rgl::clear3d() (used only if there is an active device and open = FALSE). Defaults to "all", with a new light source added to the scene.
...	additional arguments, passed to rgl::open3d() or rgl::clear3d().

Details

In addition, new3d() changes several of the default mouse actions in **rgl**, assigning the middle button to zoom (via setmouse3d()), the right button to pan (via pan3d()), and enabling a double-click with the left button to restore the scene's default viewpoint (via dbltrack3d(); keeping the mouse acting as a virtual trackball, rotating the scene, when this button is held down).

Value

Called for its side effect (opens or clears the rgl device, and configures the mouse mode); invisibly returns the current device.

See Also

rgl::open3d(), rgl::clear3d(), setmouse3d(), pan3d(), dbltrack3d().

Examples

```
library(rgl)
# New rgl device
new3d()
shade3d(volcanom, col = "lightgreen")
# Use the right mouse button to panning,
# the middle button (or wheel) to zoom,
# rotate as usual with left button; double-click resets the view
# NOTE: these mouse actions currently **do not work with RMarkdown** documents.
# ...
# Clear the current device
new3d()
sshade3d(volcanom, s = volcanom$vb[3, ], meshColor = "facesvertices",
         col = terrain.colors(128))
```

`par.reset` *Restore graphical parameters*

Description

Restores the graphical parameters changed by `splot()`/`fplot()` (or by `sxxx()`/`fxxx()` functions with `reset = FALSE`) or those previously set as the default ones.

Usage

```
par.reset(default = FALSE, set = FALSE)
```

Arguments

`default`	logical; if `FALSE` (default) restores the parameters saved automatically by the last call to `splot()` or `fplot()`. If `TRUE`, restores the *default* parameters (those in effect before the first call to `splot()`/`fplot()` or set with `par.reset(set = TRUE)`).
`set`	logical; if `TRUE`, saves the *current* graphical parameters as the new defaults (the graphical parameters are not changed).

Value

Invisibly returns the graphical parameters as they were before restoring/saving (see `par()`).

See Also

`splot()`, `fplot()`, `par()`

Examples

```
scale.range <- range(mtcars$mpg)
splot(slim = scale.range, legend.lab = "mpg")
with(mtcars,
  plot(hp, qsec, col = scolor(mpg, slim = scale.range),
       pch = 16, cex = 1.5)
)
par.reset() # restores parameters from the last splot()/fplot() call

# Set current parameters as the new default
par.reset(set = TRUE)
# ... later, after several splot()/fplot() calls ...
par.reset(default = TRUE) # restores those that were set as defaults
```

---

simage *Image plot with a color scale*

Description

simage (generic function) draws an image (a grid of colored rectangles) and (optionally) adds a legend strip with the color scale (calls splot and image).

Usage

```
simage(x, ...)

## Default S3 method:
simage(
  x = seq(0, 1, len = nrow(s)),
  y = seq(0, 1, len = ncol(s)),
  s,
  slim = range(s, finite = TRUE),
  col = jet.colors(128),
  breaks = NULL,
  legend = TRUE,
  horizontal = FALSE,
  legend.shrink = 0.8,
  legend.width = 1.2,
  legend.mar = ifelse(horizontal, 3.1, 5.1),
  legend.lab = NULL,
  bigplot = NULL,
  smallplot = NULL,
  lab.breaks = NULL,
  axis.args = NULL,
  legend.args = NULL,
  reset = TRUE,
```

```
  xlab = NULL,
  ylab = NULL,
  asp = NA,
  ...
)
```

Arguments

x	grid values for x coordinate. If x is a list, its components x$x and x$y are used for x and y, respectively. For compatibility with image, if the list has component z this is used for s.
...	additional graphical parameters (to be passed to image or simage.default; e.g. xlim, ylim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par function beforehand (e.g. par(cex.lab = 2) to increase colorbar labels).
y	grid values for y coordinate.
s	matrix containing the values to be used for coloring the rectangles (NAs are allowed). Note that x can be used instead of s for convenience.
slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the image plot in one and the legend with the color scale in the other. If FALSE only the image plot is drawn and the arguments related to the legend are ignored (splot is not called).
horizontal	logical; if FALSE (default) legend will appear on the right side. If TRUE the legend will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
bigplot	plot coordinates for main plot. If not passed these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed these will be determined within the function.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).

legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
reset	logical; if FALSE the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE (default) the plot parameters will be reset to the values before entering the function.
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.
asp	the y/x aspect ratio, see plot.window().

Value

Invisibly returns a list with the following 3 components:

bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

If reset = FALSE, the plotting region (par("plt")) may be changed after exiting, to make it possible to add more features to the plot. They can be restored using the old.par returned values or by calling function par.reset().

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal rubenfcasal@gmail.com.

See Also

splot, spoints, spersp, image, image.plot.

Examples

```
# Regularly spaced 2D data
nx <- c(40, 40) # ndata =  prod(nx)
x1 <- seq(-1, 1, length.out = nx[1])
x2 <- seq(-1, 1, length.out = nx[2])
trend <- outer(x1, x2, function(x,y) x^2 - y^2)
simage( x1, x2, trend, main = 'Trend')
```

spersp *Perspective plot with a color scale*

Description

spersp (generic function) draws a perspective plot of a surface over the x-y plane with the facets being filled with different colors and (optionally) adds a legend strip with the color scale (calls splot and persp).

Usage

```
spersp(x, ...)

## Default S3 method:
spersp(
  x = seq(0, 1, len = nrow(z)),
  y = seq(0, 1, len = ncol(z)),
  z,
  s = z,
  slim = range(s, finite = TRUE),
  col = jet.colors(128),
  breaks = NULL,
  legend = TRUE,
  horizontal = FALSE,
  legend.shrink = 0.8,
  legend.width = 1.2,
  legend.mar = ifelse(horizontal, 3.1, 5.1),
  legend.lab = NULL,
  bigplot = NULL,
  smallplot = NULL,
  lab.breaks = NULL,
  axis.args = NULL,
  legend.args = NULL,
  reset = TRUE,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  theta = 40,
  phi = 20,
  ticktype = "detailed",
  cex.axis = 0.75,
  ...
)
```

Arguments

x	grid values for x coordinate. If x is a list, its components x$x and x$y are used for x and y, respectively. If the list has component z this is used for z.

...	additional graphical parameters (to be passed to persp or spersp.default; e.g. xlim, ylim, zlim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par function beforehand (e.g. par(cex.lab = 2) to increase colorbar labels).
y	grid values for y coordinate.
z	matrix containing the values to be plotted (NAs are allowed). Note that x can be used instead of z for convenience.
s	matrix containing the values used for coloring the facets.
slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the perspective plot in one and the legend with the color scale in the other. If FALSE only the (coloured) perspective plot is drawn and the arguments related to the legend are ignored (splot is not called).
horizontal	logical; if FALSE (default) legend will appear on the right side. If TRUE the legend will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
bigplot	plot coordinates for main plot. If not passed these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed these will be determined within the function.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
reset	logical; if FALSE the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE (default) the plot parameters will be reset to the values before entering the function.
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.

zlab	label for the z axis, defaults to a description of z.
theta	x-y rotation angle for perspective (azimuthal direction).
phi	z-angle for perspective (colatitude).
ticktype	character; "simple" draws just an arrow parallel to the axis to indicate direction of increase; "detailed" draws normal ticks as per 2D plots.
cex.axis	magnification to be used for axis annotation (relative to the current setting of par("cex")).

Value

Invisibly returns a list with the following 4 components:

pm	the viewing transformation matrix (see persp for details), a 4 x 4 matrix that can be used to superimpose additional graphical elements using the function trans3d.
bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

If reset = FALSE, the plotting region (par("plt")) may be changed after exiting, to make it possible to add more features to the plot. They can be restored using the old.par returned values or by calling function par.reset().

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal rubenfcasal@gmail.com.

See Also

splot, spoints, simage, image, image.plot, persp.

Examples

```
# Regularly spaced 2D data
nx <- c(40, 40) # ndata =  prod(nx)
x1 <- seq(-1, 1, length.out = nx[1])
x2 <- seq(-1, 1, length.out = nx[2])
trend <- outer(x1, x2, function(x,y) x^2 - y^2)
spersp( x1, x2, trend, main = 'Trend', zlab = 'y')
# Multiple plots
set.seed(1)
```

```
y <- trend + rnorm(prod(nx), 0, 0.1)
# 1x2 plot
old.par <- par(mfrow = c(1,2))
spersp( x1, x2, trend, main = 'Trend', zlab = 'y', reset = FALSE)
spersp( x1, x2, y, main = 'Data', zlab = 'y', reset = FALSE)
par(old.par)
```

---

spersp3d *3D surface plot with a continuous color scale*

Description

A generic function that draws a 3D surface (rgl::persp3d()) colored according to a continuous scale associated with a vector of values s, and (optionally) adds a color-bar legend (via splot3d()).

Usage

```
spersp3d(x, ...)

## Default S3 method:
spersp3d(
  x,
  y = NULL,
  z = NULL,
  s = z,
  slim = range(s, finite = TRUE),
  col = jet.colors(128),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  aspect = !add,
  legend = TRUE,
  legend.zoom = 0.4,
  legend.width = 0.1,
  legend.dim = 0.2,
  legend.lab = NULL,
  box = TRUE,
  lab.breaks = NULL,
  lab.ticksize = 0.5,
  lab.dist = 3,
  add = FALSE,
  ...
)
```

Arguments

x	object used to select a method. In the default method, it typically provides the grid values for the x axis. The grid may be specified in several ways, see rgl::surface3d().
...	additional arguments passed to rgl::persp3d() for the main plot.
y	grid values for the y axis.
z	matrix with the surface height at each grid point (of dimension length(x) by length(y) if x and y are vectors).
s	matrix with the values used for coloring the surface (one color per vertex). Defaults to z.
slim	limits (vector of length 2 with the minimum and maximum) used to set up the color scale.
col	color table used to set up the color scale. Defaults to jet.colors(128).
xlab, ylab, zlab	titles for the axes (character strings; expressions are not accepted)
xlim, ylim, zlim	x-, y- and z-limits. If present, the plot is clipped to this region.
aspect	either a logical indicating whether to adjust the aspect ratio, or a new ratio.
legend	logical; if TRUE (default), the active rgl device is splitted into two subscenes, drawing the main plot in one and the categorical legend in the other (see fplot3d()). if FALSE only the (coloured) main plot is drawn and the arguments related to the legend are ignored (fplot3d() is not called).
legend.zoom	zoom factor applied to the legend subscene (see rgl::view3d()).
legend.width	relative width of the color bar, as a fraction of the corresponding dimension of the legend subscene/panel.
legend.dim	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
legend.lab	legend title.
box	logical; if TRUE (the default) a rectangle is drawn around the color bar.
lab.breaks	labels for the legend axis breaks; if NULL they are generated automatically (see axis3()).
lab.ticksize	length of the legend axis tick marks, as a fraction of the distance to the center of the legend panel.
lab.dist	distance between the tick marks and the labels of the legend axis, as a multiple of lab.ticksize.
add	logical; if TRUE the scatter plot is just added to the existing plot (including the legend if legend = TRUE).

Value

Called for its side effect (draws the 3D surface plot and, unless legend = FALSE, the legend on the active rgl device).

See Also

splot3d(), scolor(), sshade3d(), rgl::persp3d().

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
x <- seq(0, 1, length.out = 30)
y <- seq(0, 1, length.out = 30)
z <- outer(x, y, function(x, y) sin(2*pi*x) + 4*(y-0.5)^2 - 0.5)
spersp3d(x, y, z)
```

---

splot *Add a continuous color scale legend to an standard R plot*

Description

splot() is designed to combine a standard R plot with a legend representing a (continuous) color scale. This is done by splitting the plotting region into two parts. Keeping one for the main chart and putting the legend in the other.

Usage

```
splot(
  slim = c(0, 1),
  col = jet.colors(128),
  breaks = NULL,
  horizontal = FALSE,
  legend.shrink = 0.9,
  legend.width = 1.2,
  legend.mar = ifelse(horizontal, 3.1, 5.1),
  legend.lab = NULL,
  bigplot = NULL,
  smallplot = NULL,
  lab.breaks = NULL,
  axis.args = NULL,
  legend.args = NULL,
  add = FALSE
)
```

Arguments

slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
horizontal	logical; if FALSE (default) legend will appear on the right side. If TRUE the legend will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.

legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
bigplot	plot coordinates for main plot. If not passed these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed these will be determined within the function.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
add	logical; if TRUE the legend is just added to the existing plot (the graphical parameters are not changed).

Details

For instance, sxxxx() functions (spoints(), simage() and spersp()) draw the corresponding high-level plot (xxxx()), after calling splot(), to include a legend strip for the color scale.

These functions are based on function image.plot of package **fields**, see its documentation for additional information.

Value

Invisibly returns a list with the following 3 components:

bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

The plotting region (par("plt")) may be changed after exiting, to make it possible to add more features to the plot. They can be restored using the old.par returned values or by calling function par.reset().

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal rubenfcasal@gmail.com.

See Also

jet.colors, hot.colors, scolor, spoints, simage, spersp, image, image.plot.

Examples

```
# Plot equivalent to spoints():
scale.range <- range(mtcars$mpg)
res <- splot(slim = scale.range, legend.lab = "mpg")
with(mtcars,
   plot(hp, qsec, col = scolor(mpg, slim = scale.range),
        pch = 16, cex = 1.5, main = "Motor Trend Car Road Tests")
)
par(res$old.par) # restore graphical parameters

# Multiple plots with a common legend:
# regularly spaced 2D data...
set.seed(1)
nx <- c(40, 40) # ndata =  prod(nx)
x1 <- seq(-1, 1, length.out = nx[1])
x2 <- seq(-1, 1, length.out = nx[2])
trend <- outer(x1, x2, function(x,y) x^2 - y^2)
y <- trend + rnorm(prod(nx), 0, 0.1)
scale.range <- c(-1.2, 1.2)
scale.color <- jet.colors(256)
# 1x2 plot with some room for the legend...
old.par <- par(mfrow = c(1,2), omd = c(0.05, 0.85, 0.05, 0.95))
image( x1, x2, trend, zlim = scale.range, main = 'Trend', col = scale.color)
image( x1, x2, y, zlim = scale.range, main = 'Data', col = scale.color)
par(old.par)
# the legend can be added to any plot...
splot(slim = scale.range, col = scale.color, legend.shrink = 0.7, add = TRUE)
## note that argument 'zlim' in 'image' corresponds with 'slim' in 'sxxxx' functions.
```

---

splot3d *Add a continuous color scale legend to an 'rgl' plot*

Description

Splits the active rgl device into two subscenes (the main plot and a color-bar legend) and draws, in the legend subscene, a continuous color scale together with its corresponding values.

Usage

```
splot3d(
  slim = c(0, 1),
  col = jet.colors(128),
  legend.zoom = 0.4,
  legend.width = 0.1,
  legend.dim = 0.2,
  legend.lab = NULL,
  box = TRUE,
  lab.breaks = NULL,
  lab.ticksize = 0.5,
  lab.dist = 3,
  ...
)
```

Arguments

`slim`	limits (vector of length 2 with the minimum and maximum) used to set up the color scale.
`col`	color table used to set up the color scale. Defaults to `jet.colors(128)`.
`legend.zoom`	zoom factor applied to the legend subscene (see `rgl::view3d()`).
`legend.width`	relative width of the color bar, as a fraction of the corresponding dimension of the legend subscene/panel.
`legend.dim`	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
`legend.lab`	legend title.
`box`	logical; if `TRUE` (the default) a rectangle is drawn around the color bar.
`lab.breaks`	labels for the legend axis breaks; if `NULL` they are generated automatically (see `axis3()`).
`lab.ticksize`	length of the legend axis tick marks, as a fraction of the distance to the center of the legend panel.
`lab.dist`	distance between the tick marks and the labels of the legend axis, as a multiple of `lab.ticksize`.
`...`	additional arguments passed to `axis3()`.

Details

Mouse rotation is disabled in the legend subscene. Furthermore, since `rgl::layout3d(..., mouseMode = "replace")` does not copy the user's mouse handlers (which are therefore disabled), the first one is set up for trackball with double click to reset view (via `dbltrack3d()`), and the second one (if any) for panning (via `pan3d()`).

Value

Invisibly returns the identifiers of the created subscenes (see `rgl::layout3d()`).

See Also

spoints3d(), spersp3d(), sshade3d(), axis3(), fplot3d()

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
scale.range <- range(mtcars$mpg)
splot3d(slim = scale.range, legend.lab = "mpg")
with( mtcars, plot3d(hp, qsec, wt, type = "s",
                     col = scolor(mpg, slim = scale.range)))
```

---

spoints *Scatter plot with a color scale*

Description

A generic function that, by default, draws a scatter plot with points filled with different colors and (optionally) adds a legend strip with the color scale (spoints.default() calls splot() and plot.default(), or plot.xy() if add = TRUE).

Usage

```
spoints(x, ...)

## Default S3 method:
spoints(
  x,
  y = NULL,
  s,
  slim = range(s, finite = TRUE),
  col = jet.colors(128),
  breaks = NULL,
  type = "p",
  legend = TRUE,
  horizontal = FALSE,
  legend.shrink = 1,
  legend.width = 1.2,
  legend.mar = ifelse(horizontal, 3.1, 5.1),
  legend.lab = NULL,
  bigplot = NULL,
  smallplot = NULL,
  lab.breaks = NULL,
  axis.args = NULL,
  legend.args = NULL,
  add = FALSE,
```

```
  reset = TRUE,
  pch = 16,
  cex = 1.5,
  xlab = NULL,
  ylab = NULL,
  asp = NA,
  ...
)
```

Arguments

x	object used to select a method. In the default method, it provides the x coordinates for the plot (and optionally the y coordinates; any reasonable way of defining the coordinates is acceptable, see the function xy.coords() for details).
...	additional graphical parameters (to be passed to the main plot function or to spoints.default(); e.g. xlim, ylim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par() function beforehand (e.g. par(cex.lab = 2) to increase colorbar labels).
y	y coordinates. Alternatively, a single argument x can be provided.
s	numerical vector containing the values used for coloring the points.
slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
type	character indicating the type of plotting; actually any of the types as in plot.default().
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the main plot in one and the legend with the color scale in the other. If FALSE only the (coloured) main plot is drawn and the arguments related to the legend are ignored (splot() is not called).
horizontal	logical; if FALSE (default) legend will appear on the right side. If TRUE the legend will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend, defaults to a description of s.
bigplot	plot coordinates for main plot. If not passed, and legend is TRUE, these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed, and legend is TRUE, these will be determined within the function.

lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
add	logical; if TRUE the scatter plot is just added to the existing plot (including the legend if legend = TRUE, although the graphical parameters are not modified).
reset	logical; if FALSE the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE (default) the plot parameters will be reset to the values before entering the function.
pch	vector of plotting characters or symbols: see points().
cex	numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex").
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.
asp	the y/x aspect ratio, see plot.window().

Value

Invisibly returns a list with the following 3 components:

bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

If reset = FALSE, the plotting region (par("plt")) may be changed after exiting, to make it possible to add more features to the plot. The graphical parameters can be restored using the old.par returned values or by calling function par.reset().

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal rubenfcasal@gmail.com.

See Also

splot(), simage(), spersp(), image(), fields::image.plot(), plot.default(), fpoints().

Examples

```
with(mtcars,
    spoints(hp, qsec, mpg, main = "Motor Trend Car Road Tests")
)
```

---

spoints3d *3D scatter plot with a continuous color scale*

Description

A generic function that, by default, draws a 3D scatter plot (`rgl::plot3d()`) with points colored according to a continuous scale associated with a vector of values `s`, and (optionally) adds a colorbar legend (via `splot3d()`).

Usage

```
spoints3d(x, ...)

## Default S3 method:
spoints3d(
  x,
  y = NULL,
  z = NULL,
  s,
  slim = range(s, finite = TRUE),
  col = jet.colors(128),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  type = "p",
  legend = TRUE,
  legend.zoom = 0.4,
  legend.width = 0.1,
  legend.dim = 0.2,
  legend.lab = NULL,
  box = TRUE,
  lab.breaks = NULL,
  lab.ticksize = 0.5,
  lab.dist = 3,
  add = FALSE,
  ...
)
```

Arguments

`x`	object used to select a method. In the default method, it provides the `x` coordinates for the plot (and optionally the `y` and `z` coordinates; any reasonable way

	of defining the coordinates is acceptable, see the function xyz.coords() for details).
...	additional arguments passed to rgl::plot3d() for the main scatter plot (e.g. size).
y, z	y and z point coordinates. Alternatively, a single argument x can be provided.
s	vector used to color the points.
slim	limits (vector of length 2 with the minimum and maximum) used to set up the color scale.
col	color table used to set up the color scale. Defaults to jet.colors(128).
xlab, ylab, zlab	labels for the coordinates.
type	character indicating the type of item to plot (see rgl::plot3d(): "p" for points, "s" for spheres, "l" for lines, "h" for line segments from z = 0, and "n" for nothing).
legend	logical; if TRUE (default), the active rgl device is splitted into two subscenes, drawing the main plot in one and the categorical legend in the other (see fplot3d()). if FALSE only the (coloured) main plot is drawn and the arguments related to the legend are ignored (fplot3d() is not called).
legend.zoom	zoom factor applied to the legend subscene (see rgl::view3d()).
legend.width	relative width of the color bar, as a fraction of the corresponding dimension of the legend subscene/panel.
legend.dim	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
legend.lab	legend title.
box	logical; if TRUE (the default) a rectangle is drawn around the color bar.
lab.breaks	labels for the legend axis breaks; if NULL they are generated automatically (see axis3()).
lab.ticksize	length of the legend axis tick marks, as a fraction of the distance to the center of the legend panel.
lab.dist	distance between the tick marks and the labels of the legend axis, as a multiple of lab.ticksize.
add	logical; if TRUE the scatter plot is just added to the existing plot (including the legend if legend = TRUE).

Value

Called for its side effect (draws the 3D scatter plot and, unless legend = FALSE, the legend on the active rgl device).

See Also

fpoints3d(), splot3d(), scolor(), sshade3d(), rgl::plot3d().

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
with(mtcars, spoints3d(hp, qsec, wt, s = mpg, type = "s"))
```

---

sshade3d *Draw 3D mesh objects adding a continuous color scale*

Description

Draws a triangular mesh (mesh3d) colored according to a continuous scale associated with a vector of values s (via rgl::shade3d()), and (optionally) adds a color-bar legend (via splot3d()).

Usage

```
sshade3d(
  x,
  s,
  meshColor = c("faces", "facesvertices", "vertices"),
  slim = range(s, finite = TRUE),
  col = jet.colors(128),
  legend = TRUE,
  legend.zoom = 0.4,
  legend.width = 0.1,
  legend.dim = 0.2,
  legend.lab = NULL,
  box = TRUE,
  lab.breaks = NULL,
  lab.ticksize = 0.5,
  lab.dist = 3,
  ...
)
```

Arguments

x	triangular mesh (mesh3d object, see rgl::mesh3d()).
s	values used to color the mesh. How they are interpreted depends on meshColor.
meshColor	determines how material colours (and textures) are interpreted: "faces" applies the color per face; "facesvertices" assumes that s contains one value per vertex, calculates the average per face (via vb2tri3d()), and applies the resulting color to each face; "vertices" applies the color per vertex. See rgl::shade3d().
slim	limits (vector of length 2 with the minimum and maximum) used to set up the color scale.
col	color table used to set up the color scale. Defaults to jet.colors(128).

legend	logical; if TRUE (default), the active rgl device is splitted into two subscenes, drawing the 3D mesh object in one and the legend with the color scale in the other (see splot3d()). if FALSE only the (coloured) mesh is drawn and the arguments related to the legend are ignored (splot3d() is not called).
legend.zoom	zoom factor applied to the legend subscene (see rgl::view3d()).
legend.width	relative width of the color bar, as a fraction of the corresponding dimension of the legend subscene/panel.
legend.dim	relative dimension of the legend panel, as a fraction of the full width (or height) of the device.
legend.lab	legend title.
box	logical; if TRUE (the default) a rectangle is drawn around the color bar.
lab.breaks	labels for the legend axis breaks; if NULL they are generated automatically (see axis3()).
lab.ticksize	length of the legend axis tick marks, as a fraction of the distance to the center of the legend panel.
lab.dist	distance between the tick marks and the labels of the legend axis, as a multiple of lab.ticksize.
...	additional arguments passed to rgl::shade3d().

Value

Called for its side effect (draws the mesh and, unless add = TRUE, the legend on the active rgl device); invisibly returns the object identifiers.

See Also

splot3d(), scolor(), vb2tri3d(), fshade3d(), rgl::shade3d().

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
sshade3d(volcanom, s = volcanom$vb[3, ], meshColor = "facesvertices")
```

---

vb2tri3d *Value per triangle from values at vertices*

Description

For each triangle of a mesh, computes the average of the values associated with its three vertices. This is useful for going from a per-vertex value to a per-face value (for example, to use meshColor = "faces" in rgl::shade3d() starting from a quantity defined at the vertices).

Usage

```
vb2tri3d(x, s)
```

Arguments

x	triangular mesh (mesh3d object with an it component, see rgl::mesh3d()).
s	vector of values associated with the vertices of x (same length as the number of columns of x$vb).

Value

Numeric vector with one value per triangle (column of x$it), equal to the average of the values of s at its vertices.

See Also

sshade3d()

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
z_tri <- vb2tri3d(volcanom, volcanom$vb[3, ])
shade3d(volcanom, col = scolor(z_tri, col = terrain.colors(128)))
```

---

viewpoints *Work with 'rgl' viewpoints*

Description

Functions to retrieve, save, restore, list and remove rgl viewpoints; see *Details* for additional information.

Usage

```
addviewpoint3d(name, view = getview3d())

setviewpoint3d(name = "default")

lsviewpoints3d(...)

rmviewpoints3d(...)

getviewpoints3d()

getview3d()

setview3d(view)
```

Arguments

name	character string giving the name under which a viewpoint is stored (addviewpoint3d()) or looked up (setviewpoint3d()).
view	a list with the viewpoint parameters, typically the value returned by getview3d().
...	additional arguments to be passed to ls() or rm().

Details

addviewpoint3d() stores view under name, silently overwriting any viewpoint previously stored under the same name. By default view is the current viewpoint, as returned by getview3d().

setviewpoint3d() looks up the viewpoint stored under name and applies it to the current rgl subscene via setview3d().

lsviewpoints3d() returns the names of all the viewpoints currently stored.

rmviewpoints3d() Deletes one or more saved viewpoints by name. If called without arguments, it deletes all of them, always retaining the "default" viewpoint.

getviewpoints3d() returns a named list with all the viewpoints currently stored, one entry per name.

getview3d() retrieves the parameters that define the current viewpoint of the current rgl subscene (zoom, user matrix and user projection), so that they can be saved and restored later with setview3d().

setview3d() applies to the current rgl subscene a viewpoint previously obtained with getview3d() (this is also what setviewpoint3d() uses internally to restore a stored viewpoint).

Value

addviewpoint3d() is called for its side effect of storing view under name; it invisibly returns view.

setviewpoint3d() is called for its side effect of restoring a stored viewpoint to the current rgl subscene; it invisibly returns the corresponding view.

lsviewpoints3d() returns a character vector with the names of the stored viewpoints.

rmviewpoints3d() is called for its side effect of removing one or more stored viewpoints.

getviewpoints3d() returns a named list with all the stored viewpoints.

getview3d() returns a list with the components zoom, userMatrix and userProjection (see rgl::par3d()).

setview3d() is called mainly for its side effect of setting the viewpoint of the current rgl subscene (see rgl::par3d()); it returns the value returned by par3d().

See Also

rgl::par3d()

Examples

```
library(rgl)
open3d()
# Alternatively, use `new3d()` to clear the current device or open a new one.
# It also saves the default viewpoint under the name "default"
shade3d(volcanom, col = "lightgreen")

# ... rotate, zoom or pan the scene interactively ...
# Save the current viewpoint under the name "myview"
addviewpoint3d("myview")

# ... rotate, zoom or pan the scene interactively ...
# Restore the default viewpoint
setviewpoint3d()
# Restore the saved viewpoint
setviewpoint3d("myview")

# Names of the stored viewpoints
lsviewpoints3d()
# All stored viewpoints, as a named list
views <- getviewpoints3d()
# Remove all stored viewpoints
rmviewpoints3d()
lsviewpoints3d()
```

---

volcanom *Surface mesh of Auckland's Maunga Whau volcano*

Description

Surface mesh corresponding to the altitude on a 10m by 10m grid of Auckland's Maunga Whau volcano.

Usage

```
data(volcanom)
```

Format

`volcanom`**:**

A rgl::mesh3d-class object with 5307 vertices and 10320 triangles.

Source

datasets::volcano and example taken from rgl::surface3d.

Examples

```
library(rgl)
open3d() # Alternatively, use `new3d()` to clear the current device or open a new one
aspect3d(1, 1, 3)  # Exaggerate the relief
shade3d(volcanom, col = "lightgreen")
```

Index