Package ‘vegan3d’

February 20, 2025

Title Static and Dynamic 3D and Editable Interactive Plots for the 'vegan' Package

Version 1.4-0

Depends R (>= 3.2.0), vegan (>= 2.3-0)

Suggests tcltk

Imports cluster, lattice, rgl, scatterplot3d (>= 0.3-40)

Description Static and dynamic 3D plots to be used with ordination results and in diversity analysis, especially with the vegan package.

License GPL-2

BugReports `https://github.com/vegandevs/vegan3d/issues`

URL `https://cran.r-project.org/`, `https://github.com/vegandevs/vegan3d`

NeedsCompilation no

Author Jari Oksanen [aut, cre],
Roeland Kindt [aut],
Gavin L. Simpson [aut],
Duncan Murdoch [ctb]

Maintainer Jari Oksanen <`jhoksane@gmail.com`>

Repository CRAN

Date/Publication 2025-02-20 10:10:02 UTC

Contents

`vegan3d-package` *Dynamic and Static 3D and Interactive 2D Plots for Ordination*

Description

The **vegan3d** package provides 3D plotting for all **vegan** ordination methods or any other ordination method that **vegan** `scores` function can handle. It can also display `hclust` results in 3D over a 2D plane. Dynamic 3D plots are based on the **rgl** package and static plots are drawn with **scatterplot3d** and **lattice** packages. The package also provides 2D editable interactive plots for ordination. The points are fixed at their ordination scores, but labels can be moved to better position, and the result can be saved in various graphics formats or saved in R session and re-created with R plot commands.

Index of help topics:

ordilattice3d	3D Trellis (Lattice) Plots for Ordination
ordiplot3d	Three-Dimensional Ordination Graphics
ordirgl	Three-Dimensional Dynamic Ordination Graphics
orditkplot	Ordination Plot with Movable Labels
orditree3d	Draw Cluster Tree over a Plane
rgl.isomap	Dynamic 3D plot of isomap ordination.
rgl.renyiaccum	Dynamic Perspective Plot of Renyi Diversity Accumulation
vegan3d-package	Dynamic and Static 3D and Interactive 2D Plots for Ordination

Drawing with rgl Functions

The **rgl** graphics are dynamic 3D plots that can be spinned and zoomed with mouse. The **vegan3d** package provides interface to ordination and clustering objects. The functions use **rgl** setting and conventions and do not change the user settings. For general configuration of the plots, users should check **rgl** documentation. For instance, general look and feel of drawn items can be configured with `material3d`.

The **rgl** package may not be available on all platforms, and the package is not automatically loaded. If you want to use **rgl** functions, you must either prefix commands with `rgl::` or call `library(rgl)` in your session. If you cannot launch `rgl` window (or widget) on your platform, check the instructions on the **rgl**. In most cases, it should be possible to use **rgl** at least on a compatible browser.

Function `ordirgl` is similar to `ordiplot` in **vegan**, and any ordination result can be drawn with similar conventions. Functions with `orgl` prefix add items to existing plots, for instance, `orglellipse` is analogous to `ordiellipse`.

Function `ordirgltree` draws an `hclust` dendrogram over a plane. It was originally developed for 2D ordination planes, but any other plane can be used, for instance a projected map.

Functions `rgl.isomap` and `rgl.renyiaccum` provide alternative dynamic 3D plots for **vegan** `isomap` and `renyiaccum` functions.

Drawing with scatterplot3d and lattice Functions

The **scatterplot3d** package draws static 3D graphics, and **vegan3d** provides an interface for ordination and clustering objects. You must consult the `scatterplot3d` documentation for configuring your plots. **vegan3d** function `ordilattice3d` is similar, but is based on the **lattice** package, and can be modified using **lattice** commands and arguments.

Function `ordiplot3d` is similar to `ordirgl` or `ordiplot` and draws a static 3D plot in the standard graphical device. It returns invisibly a plotting object which contains the projected points, and **vegan** `ordi*` prefix functions can use this object. For instance, `ordiellipse` will add ellipses on the projected points.

Function `ordilattice3d` is a similar to `ordiplot3d` in **lattice** graphics, but currently more limited than `ordiplot3d`. The **lattice** package is not loaded and you need to use `library(lattice)` or prefix your **lattice** commands with `lattice::`. There are obvious ways of improving the function. Contributions are welcome.

Function `orditree3d` will draw an `hclust` dendrogram over a plane similarly as `ordirgltree`.

Editable Graphics with Tcl/Tk based orditkplot

Function draws ordination scores in a new Tcl/Tk window with fixed points and their editable labels. The labels can be moved to better positions which helps with crowded plots. It is also possible to zoom into graph to display only a part of the complete graph. The edited result can be saved in various graphical formats or saved as an R object in the session for further manipulation or re-creating the graph with standard R plotting tools.

Tcl/Tk is not available on all platforms or R installations. Use command `capabilities("tcltk")` to see if you have it in your system.

There are obvious ways of improving the function. Contributions are welcome.

`ordilattice3d` *3D Trellis (Lattice) Plots for Ordination*

Description

Function `ordicloud` provides an interface for 3D ordination graphics using Trellis function `cloud` in package **lattice**.

Usage

```
ordilattice3d(x, data = NULL, formula, display = "sites", choices = 1:3,
    panel = "panel.ordi3d", prepanel = "prepanel.ordi3d",  ...)
```

Arguments

`x`	An ordination result that **vegan** function `scores` knows: any ordination result in **vegan** and many others.
`data`	Optional data to amend ordination results. The ordination results are found from `x`, but you may give here data for other variables needed in plots. Typically these are environmental data.

`formula`	Formula to define the plots. A default formula will be used if this is omitted. The ordination axes must be called by the same names as in the ordination results (and these names vary among methods).
`display`	The kind of scores: an argument passed to `scores`.
`choices`	The axes selected: an argument passed to `scores`.
`panel, prepanel`	The names of the panel and prepanel functions.
`...`	Arguments passed to `scores` methods or **lattice** functions.

Details

The functions provide an interface to the **lattice** function `cloud`. All graphical parameters are passed to the **lattice** functions so that the graphs are configurable.

The argument `x` must always be an ordination result. The scores are extracted with **vegan** function `scores` so that these functions work with all **vegan** ordinations and many others.

The `formula` is used to define the models. Function has a default formula which is used if `formula` is missing. The formula must use the names of ordination scores and names of `data`.

The ordination scores are found from `x`, and `data` is optional. The `data` should contain other variables than ordination scores to be used in plots. Typically, they are environmental variables (typically factors) to define panels or plot symbols (**lattice** argument `groups`).

The proper work is done by the panel function. The layout can be changed by defining own panel functions. See `panel.cloud` for details and survey of possibilities.

Value

The function return `Lattice` objects of class `"trellis"`.

Note

Function is still rudimentary. For instance, it cannot display biplot arrows or factor centroids from constrained ordination or `envfit`.

Author(s)

Jari Oksanen

See Also

For functions used in `ordilattice3d` see `Lattice`, `cloud`, `panel.cloud`. Function `ordiplot3d` provides a more mature and flexible alternative which can also handle biplot arrows and factor centroids and adding new graphical elements to the plots.

Examples

```
data(dune, dune.env)
ord <- cca(dune)

ordilattice3d(ord, form = CA2 ~ CA3*CA1, groups = Manure, data = dune.env,
   scaling="sites")
```

```
ordilattice3d(ord, form = CA2 ~ CA3*CA1 | Management, groups = Manure,
   data = dune.env, auto.key = TRUE, type = c("p","h"), scaling = "sites")
```

`ordiplot3d` *Three-Dimensional Ordination Graphics*

Description

Function `ordiplot3d` displays three-dimensional ordination graphics using `scatterplot3d`. Function works with all ordination results form **vegan** and all ordination results known by `scores` function.

Usage

```
ordiplot3d(object, display = "sites", choices = c(1,3,2), col = "black",
    ax.col = "red", arr.len = 0.1, arr.col = "blue", envfit,
    xlab, ylab, zlab, ...)
```

Arguments

`object`	An ordination result or any object known by `scores`.
`display`	Display `"sites"` or `"species"` or other ordination object recognized by `scores`.
`choices`	Selected three axes. First axis is horizontal, second vertical and third is “depth”. The default will use axes 1 and 2 as front panel.
`col`	Colours of points. Can be a vector, and factors are interpreted as their internal numerical codes.
`ax.col`	Axis colour (concerns only the crossed axes through the origin).
`arr.len`	’Length’ (width) of arrow head passed to `arrows` function.
`arr.col`	Colour of biplot `arrows` and centroids of environmental variables.
`envfit`	Fitted environmental variables from `envfit` displayed in the graph.
`xlab, ylab, zlab`	Axis labels passed to `scatterplot3d`. If missing, labels are taken from the ordination result. Set to `NA` to suppress labels.
`...`	Other parameters passed to graphical functions.

Details

Function `ordiplot3d` plots static three-dimensional scatter diagrams using `scatterplot3d`. Function uses most default settings of underlying graphical functions, and you must consult their help pages to change graphics to suit your taste (see `scatterplot3d`).

Function returns invisibly an object of class `ordiplot3d` which inherits from `ordiplot`. The result object contains the projected coordinates of plotted items and functions to convert 3D data to 2D (see `scatterplot3d`). Function will display only one selected set of `scores`, typically either `"sites"` or `"species"`. Examples show how to use the invisible return object to add another set of points to the projected plot.

In constrained ordination (`cca`, `rda`, `capscale`), biplot arrows and centroids are always displayed similarly as in two-dimensional plotting function `plot.cca`. Alternatively, it is possible to display fitted environmental vectors or class centroids from `envfit`. These are displayed similarly as the results of constrained ordination, and they can be shown only for non-constrained ordination. The user must remember to specify at least three axes in `envfit` if the results are used with these functions.

The function has a `scores` method to extract the projected coordinates from the invisible return object. Standard **vegan** functions can be used with the returned object. You can use any function from the `ordihull` and `ordiarrows` families (see Examples).

Value

Function `ordiplot3d` returns invisibly an object of class `"ordiplot3d"` inheriting from `ordiplot`. The return object will contain the coordinates projected onto two dimensions for `points`, and the projected coordinates of `origin`, and possibly the projected coordinates of the heads of `arrows` and `centroids` of environmental variables. The result will also contain the object returned by `scatterplot3d`, including function `xyz.convert` which projects three-dimensional coordinates onto the plane used in the current plot (see Examples). In addition, there is a function `envfit.convert` that projects a three-dimensional `envfit` object to the current plot.

Warning

Please note that `scatterplot3d` sets internally some graphical parameters (such as `mar` for margins) and does not honour default settings. It is advisable to study carefully the documentation and examples of `scatterplot3d`.

Author(s)

Jari Oksanen

See Also

`scatterplot3d`, `ordiplot`, `ordiarrows`, `ordihull`. Function `ordilattice3d` provides an alternative based on **lattice** package. However, it not yet quite matured and cannot display biplot (or fitted) arrows and factor centroids. Currently, its main advantage is that it can split plots in different panels by external variables.

Examples

```
### Default 'ordiplot3d'
data(dune, dune.env)
ord <- cca(dune ~ A1 + Moisture, dune.env)
ordiplot3d(ord, scaling = "sites")
### A boxed 'pin' version
ordiplot3d(ord, scaling = "sites", type = "h")
### More user control
pl <- ordiplot3d(ord, scaling = "symmetric", angle=15, type="n")
points(pl, "points", pch=16, col="red", cex = 0.7)
### identify(pl, "arrows", col="blue") would put labels in better positions
text(pl, "arrows", col="blue", pos=3)
```

```
text(pl, "centroids", col="blue", pos=1, cex = 1)
### Add species using xyz.convert function returned by ordiplot3d
sp <- scores(ord, choices=1:3, display="species", scaling="symmetric")
text(pl$xyz.convert(sp), rownames(sp), cex=0.7, xpd=TRUE)
### Two ways of adding fitted variables to ordination plots
ord <- cca(dune)
ef <- envfit(ord ~ Moisture + A1, dune.env, choices = 1:3)
### 1. use argument 'envfit'
ordiplot3d(ord, envfit = ef)
### 2. use returned envfit.convert function for better user control
pl3 <- ordiplot3d(ord, scaling = "sites")
plot(pl3$envfit.convert(ef), at = pl3$origin)
### envfit.convert() also handles different 'choices' of axes
pl3 <- ordiplot3d(ord, choices = c(1,3,2))
plot(pl3$envfit.convert(ef), at = pl3$origin)
### vegan::ordiXXXX functions can add items to the plot
ord <- cca(dune)
pl4 <- with(dune.env, ordiplot3d(ord, scaling = "sites", col = Management, pch=16))
with(dune.env, ordiellipse(pl4, Management, draw = "poly", col = 1:4,
  alpha = 60))
with(dune.env, ordispider(pl4, Management, col = 1:4, label = TRUE))
```

`ordirgl` *Three-Dimensional Dynamic Ordination Graphics*

Description

Function `ordirgl` displays three-dimensional dynamic ordination graphs which can be rotated and zoomed. This function works with all ordination results from vegan and all ordination results known by the `scores` function. The `orgl`-prefixed functions add elements to the `ordirgl` graph similarly as `ordi`-prefixed functions in **vegan**.

Usage

```
ordirgl(object, display = "sites", choices = 1:3, type = "p", col = "black",
    ax.col = "red", arr.col = "yellow", radius, text, envfit, ...)
orglpoints(object, display = "sites", choices = 1:3, radius, col = "black", ...)
orgltext(object, text, display = "sites", choices = 1:3, adj = 0.5,
    col = "black", ...)
orglsegments(object, groups, order.by, display = "sites", choices = 1:3,
    col = "black", ...)
orglspider(object, groups, display = "sites", w = weights(object, display),
    choices = 1:3, col = "black", ...)
orglellipse(object, groups, display = "sites", w = weights(object, display),
    kind = c("sd", "se", "ehull"), conf, choices = 1:3, alpha = 0.3,
    col = "red", ...)
orglspantree(object, spantree, display = "sites", choices = 1:3,
    col = "black", ...)
orglcluster(object, cluster, prune = 0, display = "sites", choices = 1:3,
    col = "black", ...)
```

Arguments

`object`	An ordination result or any object known by `scores`.
`display`	Display `"sites"` or `"species"` or other ordination object recognized by `scores`.
`choices`	Selected three axes.
`type`	The type of plots: `"p"` for points or `"t"` for text labels.
`ax.col`	Axis colour (concerns only the crossed axes through the origin).
`arr.col`	Colour of biplot `arrows` and centroids of environmental variables.
`radius`	Size of points in the units of ordination scores.
`text`	Text to override the default with `type = "t"`.
`envfit`	Fitted environmental variables from `envfit` displayed in the graph. Use `envfit = NA` to suppress display of environmental variables in constrained ordination.
`adj`	Text justification passed to `text3d`.
`groups`	Factor giving the groups for which the graphical item is drawn.
`order.by`	Order points by this variable within `groups`.
`w`	Weights used to find the average within group. Weights are used automatically for `cca`, and `decorana` results, unless undone by the user. `w=NULL` sets equal weights to all points.
`kind`	Draw ellipse for standard deviations of points (`"sd"`) or standard deviations of their averages (`"se"`) or an ellipsoid hull enclosing all points in the group (`"ehull"`.
`conf`	Confidence limit for ellipses, e.g., 0.95. If not given, sd or se ellipses are drawn.
`col`	Colour of items. This can be a vector and factors are interpreted as their internal numerical values. If the function has a `groups` argument, vector `col` is used for each of these, and for other functions it is matched to points in `ordirgl` (see Details below).
`alpha`	Transparency of colour between 0.0 (fully transparent) and 1.0 (non-transparent).
`spantree`	A minimum spanning tree object from **vegan** `spantree`.
`cluster`	Result of hierarchic cluster analysis, such as `hclust` or `agnes`.
`prune`	Number of upper levels hierarchies removed from the tree. If `prune` > 0, tree will be cut into `prune + 1` disconnected trees.
`...`	Other parameters passed to graphical functions.

Details

Function `ordirgl` plots dynamic graphics using OpenGL with the `rgl` package. It clears the graphics device and starts a new plot. The function was designed for ordination methods in the **vegan** package, but it can handle any method known to **vegan** `scores` function, or to any three column matrix. The `orgl`-prefixed functions add items to the opened **rgl** graphics device.

Function `ordirgl` uses most default settings of underlying graphical functions in `rgl`. It plots only one set of points, but functions `orglpoints` and `orgltext` can add new items to an existing plot. The points are plotted using `spheres3d` and the text using `texts3d` which both have their own configuration switches and their general look and feel can be modified with `material3d`. The point size is directly defined by `radius` argument in the units of ordination scores in `spheres3d`, but `ordirgl` uses a default size of 1% of the length of the longest axis, and this can be further modified by the `cex` multiplier.

In constrained ordination (`cca`, `rda`, `capscale`), biplot arrows and centroids are always displayed similarly as in two-dimensional plotting function `plot.cca`. Alternatively, it is possible to display fitted environmental vectors or class centroids from `envfit` in both graphs. These are displayed similarly as the results of constrained ordination, and they can be shown only for non-constrained ordination. The user must remember to specify at least three axes in `envfit` if the results are used with these functions.

Function `orglsegments` is similar to **vegan** `ordisegments` and connects points by line segments. This can be useful for regular transects. The colour of segments can be a vector which corresponds to the `groups` and will be recycled.

Function `orglspider` is similar as **vegan** `ordispider`: it connects points to their weighted centroid within `"groups"`, and in constrained ordination it can connect `"wa"` or weighted averages scores to corresponding `"lc"` or linear combination scores if `"groups"` is missing. Function `orglellipse` is similar as **vegan** `ordiellipse` and draws ellipsoids of standard deviance, standard error or confidence regions for `groups`. At least four points are needed to define an ellipsoid in 3D, and even these will fail if all points are strictly on 2D. The `col` argument for both of these functions can be a vector corresponding to the `groups`.

Function `orglspantree` adds a minimum spanning tree from **vegan** `spantree`. This a 3D equivalent of `lines.spantree`. Function `orglcluster` adds a hierarchic cluster tree from `hclust` or related functions. This is a 3D equivalent of `ordicluster`. The `col` argument for both of these functions can be a vector corresponding to the connected points. In `orglspantree` the line colour is a mixture of colours of joined points, and in `orglcluster` it is a mixture of all points in the cluster.

Value

Function `ordirgl` returns nothing.

Warning

Function `ordirgl` uses OpenGL package `rgl` which may not be functional in all platforms.

Author(s)

Jari Oksanen

See Also

`rgl`, `spheres3d`, `text3d`, `rgl.viewpoint`, `envfit`. These are 3D dynamic variants of **vegan** functions `ordiplot`, `ordisegments`, `ordispider` and `ordiellipse`, `ordicluster` and `lines.spantree`.

Examples

```
if (interactive() && require(rgl, quietly = TRUE)) {
data(mite, mite.env)
ord <- rda(decostand(mite, "hellinger"))
ordirgl(ord, size=4, col = "yellow")
```

```
orgltext(ord, display = "species")
## show groups of Shrub abundance
## ordirgl: col by points
with(mite.env, ordirgl(ord, col = as.numeric(Shrub), scaling = "sites"))
## orglspider & orglellipse: col by groups
with(mite.env, orglspider(ord, Shrub, col = 1:3, scaling = "sites"))
with(mite.env, orglellipse(ord, Shrub, col = 1:3, kind = "se", conf = 0.95,
    scaling = "sites"))
}
```

`orditkplot` *Ordination Plot with Movable Labels*

Description

Function `orditkplot` produces an editable ordination plot with points and labels. The labels can be moved with mouse, and the edited plot can be saved as an encapsulated postscript file or exported via R `plot` function to other graphical formats, or saved in the R session for further processing.

Usage

```
orditkplot(x, display = "species", choices = 1:2, width, xlim, ylim,
   tcex = 0.8, tcol, pch = 1,  pcol, pbg, pcex = 0.7, labels,  ...)
## S3 method for class 'orditkplot'
plot(x, ...)
## S3 method for class 'orditkplot'
points(x, pch = x$args$pch, cex = x$args$pcex,
       col = x$args$pcol, bg = x$args$pbg, ...)
## S3 method for class 'orditkplot'
text(x, cex = x$args$tcex, col = x$args$tcol,
     font = attr(x$labels, "font"), ...)
## S3 method for class 'orditkplot'
scores(x, display, ...)
```

Arguments

`x`	An ordination result or any other object that `scores` can handle, or for the `plot` function the object dumped from the interactive `orditkplot` session.
`display`	Type of `scores` displayed. For ordination scores this typically is either `"species"` or `"sites"`, and for `orditkplot` result it is either `"points"` or `"labels"`.
`choices`	Axes displayed.
`width`	Width of the plot in inches; defaults to the current width of the graphical device.
`xlim, ylim`	x and y limits for plots: points outside these limits will be completely removed.
`tcex`	Character expansion for text labels.
`tcol`	Colour of text labels.

`pch, pcol, pbg`	Point type and outline and fill colours. Defaults `pcol="black"` and `pbg="transparent"`. Argument `pbg` has an effect only in filled plotting characters `pch = 21` to `25`.
`pcex`	Expansion factor for point size.
`labels`	Labels used instead of row names.
`cex, col, bg, font`	graphical parameters used in the `points` and `text` methods. See `par`.
`...`	Other arguments passed to the function. These can be graphical parameters (see `par`) used in the plot, or extra arguments to `scores`. These arguments are ignored in `plot`, but honoured in `text` and `points`.

Details

Function `orditkplot` uses **tcltk** package to draw Tcl/Tk based ordination graphics with points and labels. The function opens an editable canvas with fixed points, but the labels can be dragged with mouse to better positions or edited. In addition, it is possible to zoom to a part of the graph.

The function knows the following mouse operations:

- **Left mouse button** can be used to move labels to better positions. A line will connect a label to the corresponding point.
- **Double clicking left mouse button** opens a window where the label can be edited. After editing the label, hit the Return key.
- **Right mouse button** (or alternatively, Shift-Mouse button with one-button mouse) can be used for zooming to a part of the graph. Keeping the mouse button down and dragging will draw a box of the zoomed area, and after releasing the button, a new plot window will be created (this is still preliminary: all arguments are not passed to the new plot).

In addition there are buttons for the following tasks: **Copy to EPS** copies the current plot to an encapsulated postscript (eps) file using standard Tcl/Tk utilities. The faithfulness of this copy is system dependent. Button **Export plot** uses `plot.orditkplot` function to redraw the plot into graphical file formats. Depending on the system, the following graphical formats may be available: eps, pdf, svg, png, jpeg, tiff, bmp or xfig. Some of the output formats may be edited with external software: svg files with Illustrator or Inkscape, and xfig with the legacy program XFig. Button **Save to R** writes the edited coordinates of labels and points to the R session for further processing, and the `plot.orditkplot` function can be used to display the results. For faithful replication of the plot, the graph must have similar dimensions as the `orditkplot` canvas had originally. The `plot` function cannot be configured, but it uses the same settings as the original Tcl/Tk plot. However, `points` and `text` functions are fully configurable, but use the stored defaults for consistency with `plot.orditkplot` if none are supplied. Finally, button **Close** closes the window.

The produced plot will have equal aspect ratio. The width of the horizontal axis is fixed, but vertical axes will be scaled to needed height, and you can use scrollbar to move vertically if the whole canvas does not fit the window. If you use dumped labels in ordinary R plots, your plot must have the same dimensions as the `orditkplot` canvas to have identical location of the labels.

The function only displays one set of scores. However, you can use `ordipointlabel` (**vegan**) to produce a result object that has different points and text types for several sets of scores and this can be further edited with `orditkplot`. For a good starting solution you need to scale the `ordipointlabel` result so that the points span over the whole horizontal axis. The function cannot show environmental variables or constraints, but it is limited to unconstrained ordination.

The plot is a Tcl/Tk canvas, but the function tries to replicate standard graphical device of the platform, and it honours several graphical parameters (see `par`). Many of the graphical parameters can be given on the command line, and they will be passed to the function without influencing other graphical devices in R. At the moment, the following graphical parameters are honoured: `pch` `bg`, `cex`, `cex.axis`, `cex.lab`, `col` (for labels), `col.axis`, `col.lab`, `family` (for font faces), `fg`, `font`, `font.axis`, `font.lab`, `lheight`, `lwd` (for the box), `mar`, `mex`, `mgp`, `ps`, `tcl`. These can be set with `par`, and they also will influence other plots similarly.

The `tkcanvas` text cannot be rotated, and therefore vertical axis is not labelled, and `las` `par`ameter will not be honoured in the Tcl/Tk plot, but it will be honoured in the exported R plots and in `plot.orditkplot`.

Value

Function returns nothing useful directly, but you can save the edited graph to a file or save the edited positions to an R session for further processing and plotting.

Note

You need **tcltk** package and R must have been configured with `capabilities` for `tcltk`. Depending on your OS, you may need to start X11 and set the display before loading **tcltk** and starting the function (for instance, with `Sys.setenv("DISPLAY"=":0")`). See `tcltk-package`.

Author(s)

Jari Oksanen

See Also

Function `ordipointlabel` is an automatic procedure with similar goals of avoiding overplotting, and its output can be edited with `orditkplot`. See `ordiplot`, `plot.cca`, and `orditorp` for alternative ordination plots.

Examples

```
if(interactive() && capabilities("tcltk")) {
data(varespec)
ord <- cca(varespec)
## Do something with the graph and end by clicking "Dismiss"
orditkplot(ord, mar = c(4,4,1,1)+.1, font=3)
## Use ordipointlabel to produce a plot that has both species and site
## scores in different colors and plotting symbols
pl <- ordipointlabel(ord)
orditkplot(pl)
}
```

`orditree3d` *Draw Cluster Tree over a Plane*

Description

Function draws a 3D plot where ordination result is at the bottom plane and a `hclust` dendrogram is drawn above the plane.

Usage

```
orditree3d(ord, cluster, prune = 0, display = "sites", choices = c(1, 2),
    col = "blue", text, type = "p", ...)
ordirgltree(ord, cluster, prune = 0, display = "sites", choices = c(1, 2),
    col = "blue", text, type = "p", ...)
```

Arguments

`ord`	An ordination object or an `ordiplot` object or any other structure defining a 2D plane.
`cluster`	Result of hierarchic cluster analysis, such as `hclust` or `agnes` or any other clustering that can be coerced to a compliant format by `as.hclust`.
`prune`	Number of upper levels hierarchies removed from the tree. If `prune` > 0, tree will be cut into `prune + 1` disconnected trees.
`choices`	Choice of ordination axes.
`display`	Ordination scores displayed.
`col`	Colour of tree. The colour can be a vector and it is used for the points, text and terminal branches. The colour of internal branches is a mixture of connected leaves.
`text`	Text to replace the default of item labels when `type = "t"`.
`type`	Display of leaves: `"p"` for points, `"t"` for text, and `"n"` for no display.
`...`	Arguments passed to `scores` and graphical functions.

Details

`orditree3d` uses **scatterplot3d** package to draw a static 3D plot of the dendrogram over the ordination, and `ordirgltree` uses **rgl** to make a dynamic, spinnable plot. The functions were developed to plot a cluster dendrogram over a 2D ordination plane, but any other plane can be used, for instance, a map.

Value

Function `orditree3d` returns invisibly a `scatterplot3d` result object amended with items `points` and `internal` that give the projected coordinates of ordination scores and internal nodes, and `col.points` and `col.internal` that give their colours. All matrix-like objects can be accessed with `scores`.

Function `ordirgltree` returns nothing.

Author(s)

Jari Oksanen.

See Also

`orglcluster` and `ordicluster` (in **vegan**).

Examples

```
data(dune, dune.env)
d <- vegdist(dune)
m <- metaMDS(d, trace = FALSE)
cl <- hclust(d, "aver")
orditree3d(m, cl, pch=16, col=cutree(cl, 3))
## ordirgltree makes ordinary rgl graphics. It accepts
## material3d() settings, and you can add elements to the
## open graph (for instance, bbox3d()).
if (interactive() && require(rgl, quietly = TRUE)) {
with(dune.env, ordirgltree(m, cl, col = as.numeric(Management), size = 6,
   lwd = 2, alpha = 0.6))
}
```

`rgl.isomap` *Dynamic 3D plot of isomap ordination.*

Description

Function displays a dynamic 3D plot from `isomap` ordination.

Usage

```
rgl.isomap(x, web = "white", ...)
```

Arguments

`x`	Result from `isomap`.
`web`	Colour of the web. If this is a vector matching the number of points, the colour of links is a mixture of joined points. `NA` skips drawing the web.
`...`	Other parameters passed to `ordirgl` and `scores`.

Details

Function `rgl.isomap` displays dynamic 3D plots that can be rotated on the screen. The functions is based on `ordirgl`, but it adds the connecting lines. The function passes extra arguments to `scores` or `ordirgl` functions so that you can select axes, or define colours and sizes of points.

Value

Function returns nothing.

Note

This is a support function for `isomap` ordination in the **vegan** package.

Author(s)

Jari Oksanen.

See Also

`isomap`, `ordirgl`, `scores`.

Examples

```
if (interactive() && require(rgl, quietly = TRUE)) {
data(BCI)
dis <- vegdist(BCI)
## colour points and links by the dominant species
dom <- factor(make.cepnames(names(BCI))[apply(BCI, 1, which.max)])
ord <- isomap(dis, k=3)
rgl.isomap(ord, col = as.numeric(dom), web = as.numeric(dom), lwd=2)
}
```

`rgl.renyiaccum` *Dynamic Perspective Plot of Renyi Diversity Accumulation*

Description

Function `rgl.renyiaccum` displays a dynamic 3D plot of the result of `renyiaccum` function in the **vegan** package. Function `persp.renyiaccum` (in **vegan**) produces similar static plots.

Usage

```
rgl.renyiaccum(x, rgl.height = 0.2, ...)
```

Arguments

`x`	A `renyiaccum` result.
`rgl.height`	Vertical scaling of the plot.
`...`	Other arguments passed to the function (ignored).

Details

This is a graphical support function to `renyiaccum` in **vegan**. Similar static plots can be produced by `persp.renyiaccum`.

Value

Function returns nothing.

Author(s)

Roeland Kindt.

See Also

`renyiaccum`, `persp.renyiaccum`, `rgl`.

Examples

```
if (interactive() && require(rgl, quietly = TRUE)){
data(BCI)
mod <- renyiaccum(BCI[1:12,])
persp(mod)
rgl.renyiaccum(mod)
}
```

Index