Package 'RSurvey'

June 1, 2011

Version 0.6-3

Date 2011-05-29

Title Analysis of Spatially Distributed Data

Author Jason C. Fisher <jfisher@usgs.gov>

Maintainer Jason C. Fisher <jfisher@usgs.gov>

Depends R (>= 2.12.0), tcltk, sp, gpclib, rgl, MBA, tripack

Suggests rgdal

SystemRequirements Tcl/Tk (>= 8.5), Tktable (>= 2.9, optional)

Description This package is a processing program for spatially distributed data. It features graphing tools, query building,and polygon clipping. A graphical user interface is provided.

License GPL (>= 2)

URL http://rsurvey.org

R topics documented:

`RSurvey-package` *Analysis of Spatially Distributed Data*

Description

This package is a processing program for spatially distributed data. It features graphing tools, query building, and polygon clipping. A graphical user interface (GUI) is provided.

Details

Package:	RSurvey
Type:	Package
Version:	0.6-3
Date:	2011-05-29
License:	GPL (>= 2)

Note

The **RSurvey** GUI requires R operate as an SDI application, using multiple top-level windows for the console, graphics, and pager. Files can be one of four types as indicated by their extension: tables (‘`.txt`’, ‘`.csv`’, ‘`.dat`’, or ‘`.shp`’), grids (‘`.grd`’), polygons (‘`.ply`’), or binary project images (‘`.rda`’).Tables (‘`.txt`’, ‘`.csv`’, ‘`.dat`’) can be compressed by gzip with additional extension ‘`.gz`’. Shapefiles (‘`.shp`’) and interpolated grid files (‘`.grd`’) are limited to data export. Support for programmatic manipulation of measurement units is only provided for date and time values; therefore, the bulk of unit consistency is tasked to the user. Time zones, spatial datum’s and projections are not supported.

The set of standards used for coding **RSurvey** is documented in Google’s R Style Guide.

Author(s)

Jason C. Fisher

Maintainer: <`<jfisher@usgs.gov>`>

Examples

```
library(RSurvey)
```

AddAxis — *Add an Axis to a Plot*

Description

Adds an axis to the current plot.

Usage

```
AddAxis(side, lim, ticks.inside = FALSE, minor.ticks = FALSE, ...)
```

Arguments

`side`	integer; a vector of values specifying the plot sides for the axis to be drawn.
`lim`	numeric or POSIXt; the axis limits (`x1`, `x2`) of the plot.
`ticks.inside`	logical; if `TRUE` tickmarks are placed inside the plot region; its default is `FALSE`.
`minor.ticks`	logical; if `TRUE` minor tickmarks are added to the plot; its default is `FALSE`.
`...`	other graphical parameters may also be passed as arguments to this function.

Details

The plot sides are designated as: 1 = below, 2 = left, 3 = above, and 4 = right.

Author(s)

Fisher, J.C.

See Also

`axis`, `axis.POSIXct`, `seq`, `pretty`

Examples

```
x <- as.POSIXlt("2001/1/1") + 700 * sort(runif(10))
y <- rnorm(10)
xlim <- extendrange(x, f = 0.02)
ylim <- extendrange(y, f = 0.02)
plot(x, y, axes = FALSE)
box()
AddAxis(side = 1, lim = xlim)
AddAxis(side = 2, lim = ylim, ticks.inside = TRUE)
AddAxis(side = 3, lim = xlim, minor.ticks = TRUE)
AddAxis(side = 4, lim = ylim, ticks.inside = TRUE, minor.ticks = TRUE)
```

`Autocrop` *Autocrop Spatial Domain*

Description

Approximate the shape of an area defined by a set of points in a plane.

Usage

```
Autocrop(mesh, max.len, max.itr = 10000)
```

Arguments

`mesh`	tri; a Delaunay triangulation.
`max.len`	numeric; maximum arc length for an outer triangle.
`max.itr`	integer; maximum number of iterations.

Details

This subroutine uses a Delaunay triangulation to approximate the shape of an area defined by a set of arbitrarily distributed points in a plane. All triangles with arc lengths greater than an established maximum length are removed; a polygon is created from the union of the remaining triangles.

Value

Returns a polygon object of class `gpc.poly`.

Author(s)

Fisher, J.C.

See Also

`AutocropPolygon`, `tri.mesh`

Examples

```
data(tritest)
mesh <- tri.mesh(tritest$x, tritest$y)
plot(mesh)
ply <- Autocrop(mesh, max.len = 0.5, max.itr = 100)
plot(ply, add = TRUE, poly.args = list(col = 2))
```

`AutocropPolygon` *Set Autocrop Input Parameters*

Description

A GUI for specifying input parameters for the `Autocrop` function.

Usage

```
AutocropPolygon(parent = NULL)
```

Arguments

`parent`	tkwin; the parent window (optional).

Details

This subroutine calls on the `x` and `y` components of `data.raw`, a data table stored in `Data` (see `ReadData`). A Delaunay triangulation is created from the set of arbitrarily distributed points and the area defining these points is approximated using the `Autocrop` function. The default maximum arc length is the maximum outer arc length for the mesh. Entering arc lengths less than the default value will result in a reduced area for the polygon. A point plot is drawn showing the resulting polygon based on the user defined input parameters. Plotting parameters are queried from `Data`.

Value

Returns a polygon object of class `gpc.poly`.

Author(s)

Fisher, J.C.

See Also

`tri.mesh`, `Autocrop`, `Plot2d`

Examples

```
data(tritest)
Data("data.pts", as.data.frame(tritest))
Data("vars", list(x = 1, y = 2))
AutocropPolygon()
```

`CheckEntry` *Content Control within Entry Widget*

Description

Content control for character strings based on an expected entry type.

Usage

```
CheckEntry(ent.typ, ent.str = "")
```

Arguments

`ent.typ`	character; the entry type.
`ent.str`	character; the entry value.

Details

The entry types include: *real*, *integer*, *hour*, *minute*, *second*, and *date*.

Value

A character string with strict adherence to the specified format of the entry type.

Author(s)

Fisher, J.C.

See Also

`tkentry`

Examples

```
CheckEntry("numeric", "3.14ab")
## [1] "3.14"

CheckEntry("integer", "3.")
## [1] "3"

CheckEntry("hour", "13")
## [1] "13"

CheckEntry("hour", "25")
## [1] "23"
```

`CutoutPolygon` *Determine Grid Points within Polygon*

Description

This function excludes gridded data lying outside of a given polygon.

Usage

```
CutoutPolygon(dat, ply = NULL)
```

Arguments

`dat`	list; with components `x`, `y`, and `z`, see 'Value'.
`ply`	gpc.poly; the polygon defining the crop region for the gridded data.

Details

Values of `z` corresponding to coordinates (`x`, `y`) located outside the polygon will be set to `NA`.

Value

Returns a list containing the following components:

`x`	numeric; a vector of `x` coordinates.
`y`	numeric; a vector of `y` coordinates.
`z`	matrix; the state variable corresponding to coordinates in the grid.

Author(s)

Fisher, J.C.

See Also

`point.in.polygon`

Examples

```
x11()

ply <- as(cbind(c(2, 8, 9, 6, 3), c(3, 1, 4, 8, 6)), "gpc.poly")
x <- seq(0, 10, 0.1)
y <- seq(0, 10, 0.1)
z <- matrix(runif(length(x) * length(y)), nrow = length(y),
            ncol = length(x))

d <- list(x = x, y = y, z = z)
filled.contour(d, plot.axes = {axis(1); axis(2); plot(ply, add = TRUE)})

d <- CutoutPolygon(d, ply)
filled.contour(d, color.palette = terrain.colors)
```

`Data` *Set or Query Data and Parameters*

Description

A function to set or query all data and parameters used in **RSurvey**.

Usage

```
Data(option, value, clear.all = FALSE, clear.proj = FALSE,
     clear.data = FALSE)
```

Arguments

`option`	character; the parameter name, see 'Parameters'.
`value`	a parameter value specified for `option`.
`clear.all`	logical; if `TRUE` all parameters are cleared from `Data`, its default is `FALSE`.
`clear.proj`	logical; if `TRUE` basic GUI preferences will be saved and all other data removed, its default is `FALSE`.
`clear.data`	logical; if `TRUE` only data sets will be removed, its default is `FALSE`.

Value

If `value` is given the object specified by `option` is returned. A `NULL` value is returned for objects not yet assigned a value and where no default value is available.

Data

Imported raw data is saved to the data frame `data.raw` (see `ReadData`). Processed point data is saved to the data frame `data.pts` and interpolated surface data to the list `data.grd` (see `ProcessData`).

Parameters

Parameters undefined elsewhere in this documentation include:

`ver` character; the package version number.

`win.loc` character; the default horizontal and vertical location for GUI placement in pixels.

Author(s)

Fisher, J.C.

Examples

```
# To set a parameter
Data("test1", 3.14159265)
Data("test2", list(id = "PI", val = 3.14159265))
# To retrieve a parameter value
Data("test1")
Data("test2")
Data(c("test2", "id"))
Data(c("test2", "val"))
# To get all parameter values
d <- Data()
# To clear all parameters, use at your own risk
## Not run: Data(clear.all = TRUE)
```

`EditDateFormat` *A GUI for constructing date and time formats.*

Description

A GUI for converting between character representations and objects of class "POSIXt" representing calendar dates and times.

Usage

```
EditDateFormat(spec = "", parent = NULL)
```

Arguments

`spec`	character; the conversion specification for date-time values.
`parent`	tkwin; the parent window (optional).

Value

Returns a character string representing the formatted time.

Author(s)

Fisher, J.C.

See Also

`strptime`, `format`

Examples

```
EditDateFormat(spec = "%d/%m/%Y")
```

`EditFunction` *Function editor for table data*

Description

A GUI for defining functions in the R language.

Usage

```
EditFunction(cols, index = NULL, parent = NULL)
```

Arguments

`cols`	lsit; see `ManageData`.
`index`	integer; an element index number in `cols` (optional).
`parent`	tkwin; the parent window (optional).

Details

This GUI is appropriate for defining new variables in a pre-existing data frame.

Value

Results in a character string of the edited function; when evaluated, this text must be parseable and result in a vector of length equal to the number of rows in the `data.raw` data frame (see `ReadData`).

Author(s)

Fisher, J.C.

See Also

`parse`, `EvalFunction`

Examples

```
data(tritest)
Data("data.raw", as.data.frame(tritest))
cols <- list()
cols[[1]] <- list(id = "X", index = 1, fun = "DATA[[\"X\"]]")
cols[[2]] <- list(id = "Y", index = 2, fun = "DATA[[\"Y\"]]")
cols[[3]] <- list(id = "New Variable",
                  fun = "DATA[[\"X\"]] + DATA[[\"Y\"]]")
EditFunction(cols, index = 3)
```

`EditLimits` *Set Limits for Data and Axes*

Description

A GUI for specifying data and axes limits.

Usage

```
EditLimits(lim = NULL, win.title = "Limits", parent = NULL)
```

Arguments

`lim`	list; contains the current plotting limits, see 'Value'.
`win.title`	character; the title of the main window (optional).
`parent`	tkwin; the parent window (optional).

Value

Returns a list containing the following components:

`x1, x2`	numeric; the minimum and maximum `x` value.
`y1, y2`	numeric; the minimum and maximum `y` value.
`z1, z2`	numeric; the minimum and maximum `z` value.
`t1, t2`	POSIXct; the minimum and maximum `t` value.
`x1.chk, x2.chk`	logical; if `TRUE` a default value is used for the minimum and maximum `x` value.
`y1.chk, y2.chk`	logical; if `TRUE` a default value is used for the minimum and maximum `y` value.
`z1.chk, z2.chk`	logical; if `TRUE` a default value is used for the minimum and maximum `z` value.
`t1.chk, t2.chk`	logical; if `TRUE` a default value is used for the minimum and maximum `t` value.
`x`	numeric; a vector of `x` limits `(x1,x2)`, default is `(NA,NA)`.
`y`	numeric; a vector of `y` limits `(y1,y2)`, default is `(NA,NA)`.
`z`	numeric; a vector of `z` limits `(z1,z2)`, default is `(NA,NA)`.

Author(s)

Fisher, J.C.

Examples

```
EditLimits()
```

`EvalFunction` *Evaluates an R Statement*

Description

Evaluates a character string representation of an R statement.

Usage

```
EvalFunction(txt, cols)
```

Arguments

`txt`	character; a string representation of an R function; see 'Details'.
`cols`	list; see `ManageData`.

Details

The "DATA" identifier is a reserved word within the `txt` argument. "DATA" is used to reference the `data.raw` data frame, a component of `Data` with variable names keyed to column index numbers in `data.raw` using the `vars` argument.

Value

The result of evaluating the `txt` object after the appropriate substitutions for "DATA" has been made. `Inf`, `-Inf`, and `NaN` values are converted to `NA` in numeric vectors.

Author(s)

Fisher, J.C.

See Also

`parse`, `eval`, `round`, `is.infinite`, `is.nan`

Examples

```
data(tritest)
Data("data.raw", as.data.frame(tritest))
cols <- list()
cols[[1]] <- list(id = "X", index = 1, fun = "DATA[[\"X\"]]")
cols[[2]] <- list(id = "Y", index = 2, fun = "DATA[[\"Y\"]]")
EvalFunction("DATA[[\"X\"]]", cols)
cols[[1]]$digits <- 0
EvalFunction("DATA[[\"X\"]]", cols)
EvalFunction("DATA[[\"X\"]] + DATA[[\"Y\"]]", cols)
EvalFunction("rnorm(12)", cols)
```

GetFile — *Select a File to Open or Save As*

Description

A GUI for selecting files to open or save.

Usage

```
GetFile(cmd = "Open", file = NULL, exts = NULL, initialdir = NULL,
        initialfile = NULL, defaultextension = NULL,
        win.title = cmd, multi = FALSE, parent = NULL)
```

Arguments

`cmd`	character; specifies if an `"Open"` or `"Save As"` file management pop up dialog box is implemented.
`file`	character; the name of the file which the data are to be read from. Alternatively, `file` can be a readable text-mode `connection` (optional).
`exts`	character; a vector of default file extensions.
`initialdir`	character; specifies that the files in this directory should be displayed when the dialog pops up.
`initialfile`	character; the filename to be displayed in the dialog when it pops up.
`defaultextension`	character; the string that will be appended to the filename if the user enters a filename without an extension.
`win.title`	character; a string to display as the title of the dialog box.
`multi`	logical; if `TRUE` multiple files may be selected, its default is `FALSE`.
`parent`	tkwin; the parent window (optional).

Value

If `multi` is `FALSE` returns a list containing the following components:

`path`	character; the `file` path
`dir`	character; the directory that contains the `file`
`name`	character; the `file` name
`ext`	character; the `file` extension
`type`	character; the `file` type

Otherwise, a list is returned containing list components for each file.

Author(s)

Fisher, J.C.

Examples

```
GetFile()
```

`ImportData` *Import Data*

Description

A GUI for reading table formatted data.

Usage

```
ImportData(parent = NULL)
```

Arguments

`parent`	tkwin; the parent window (optional).

Details

This GUI lets you specify the format and connection type for table data. Data connections are defined as the path to the file to be opened or a complete URL (e.g. http://, ftp:// or file://), or clipboard. Files are limited to text format (‘`.txt`’ ‘`.csv`’, or ‘`.dat`’); however, they can be compressed by `gzip` with additional extension ‘`.gz`’.

Value

Queries and sets the following components of `Data` (see `ReadData`): `headers`, `skip`, `sep`, `nrows`, `na.strings`, `quote`, `comment.char`; and

`data.source`	character; a description of the connection (i.e. a file pathname).

Note

Requires the Tcl package Tktable. If Tktable is not available the `ReadData` function is called directly using default argument values.

Author(s)

Fisher, J.C.

See Also

`ReadData`, `read.table`, `connections`

Examples

```
tclRequire("Tktable", warn = TRUE)
ImportData()
```

`LoadPackages` *Load Required Packages for RSurvey*

Description

This function installs R packages required by **RSurvey**. If a required package is unavailable on the local computer an attempt is made to acquire the package from CRAN using an existing network connection.

Usage

```
LoadPackages(repo = "http://cran.r-project.org")
```

Arguments

`repo`	character; the base URL of the repositories to use for package installation.

Author(s)

Fisher, J.C.

See Also

`install.packages`, `require`

Examples

```
LoadPackages()
```

`ManageData` *Manage Data*

Description

A GUI for managing, querying, and formatting data.

Usage

```
ManageData(cols, vars, parent = NULL)
```

Arguments

`cols`	list; see ‘Value’.
`vars`	list; see ‘Value’.
`parent`	tkwin; the parent window (optional).

Details

This GUI lets you: (1) specify the names, measurement units, and decimal precision of variables; (2) add new variables based on user defined functions (see `EditFunction`); and (3) remove and (or) reorder variables in the data table.

Value

Queries and sets the `cols` and `vars` components of `Data`. The `cols` object is a list whose length is equal to the current number of data variables. Each component in `cols` is linked to a specific variable, and contains the following components:

`name`	character; variable name.
`unit`	character; measurement units (optional); programmatic manipulation of measurement units is only supported for date and time variables.
`id`	character; a unique identifier that is typically created from a string concatenation of `name` and `unit`.
`fun`	character; the expression evaluated when computing the vector of values for a variable.
`index`	integer; the variables component index number in the `data.raw` data frame (see `ImportData`).
`class`	character; the class of the data vector object.
`digs`	integer; the precision of the variable, defined as the number of fractional digits or decimal places (optional).
`summary`	list; a summary of the variables descriptive statistics (see `SummarizeData`).
`comments`	character; user comments (optional).

The `vars` object is a list with components:

`x, y, z, t, vx, vy`	integer; the index number of the corresponding state variable in `cols`.

Note

The `vars` object is only updated to reflect the removal and (or) reordering of variables.

Author(s)

Fisher, J.C.

See Also

`EditFunction`

Examples

```
ManageData()
```

`ManagePolygons` *Manage Polygons*

Description

A GUI for managing and manipulating polygons that is based on the **gpclib** package.

Usage

```
ManagePolygons(ply = NULL, encoding = getOption("encoding"),
               parent = NULL)
```

Arguments

`ply`	list; its components are objects of class `gpc.poly-class`.
`encoding`	character; encoding to be assumed for input strings. If the value is `"latin1"` or `"UTF-8"` it is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.
`parent`	tkwin; the parent window (optional).

Details

The text file representation of a polygon is of the following format:

```
<number of contours>
<number of points in first contour>
<hole flag>
x1 y1
x2 y2
...
<number of points in second contour>
<hole flag>
x1 y1
x2 y2
...
```

The hole flag is either 1 to indicate a hole, or 0 for a regular contour. See `read.polyfile` within the **gpclib** package for details.

Value

Queries and sets the `ply` (see ‘Arguments’), `poly.data`, and `poly.crop` (see `SetPolygonLimits`) components of `Data`.

Author(s)

Fisher, J.C.

See Also

`polyfile`, `union`, `setdiff`, `intersect`

Examples

```
ManagePolygons()
```

`OpenRSurvey` *Open Main Graphical User Interface*

Description

This function activates the main GUI for **RSurvey**.

Usage

```
OpenRSurvey()
```

Details

All functions within **RSurvey** are accessible in this GUI.

Value

Quaries and sets the `vars` component of `Data`. The `vars` object is a list with components:

`x, y, z, t, vx, vy`
integer; the index number of the corresponding state variable in `cols`.

Author(s)

Fisher, J.C.

Examples

```
OpenRSurvey()
```

`Plot2d` *Plot Points or Interpolated Surface*

Description

Draws a scatter plot or contour plot with arrows. A key showing how the colors map to state variable values is shown to the right of the plot.

Usage

```
Plot2d(x = NULL, y = NULL, z = NULL, vx = NULL, vy = NULL,
       type = "p", xlim = NULL, ylim = NULL, zlim = NULL,
       xlab = NULL, ylab = NULL, zlab = NULL, asp = NA,
       csi = NA, width = 7, pointsize = 12, cex.pts = 1,
       nlevels = 20, rkey = FALSE,
       color.palette = terrain.colors,
       vuni = FALSE, vmax = NULL, vxby = NULL, vyby = NULL,
       axis.side = 1:2, minor.ticks = FALSE,
       ticks.inside = FALSE, add.contour.lines = FALSE)
```

Arguments

`x`	numeric; a vector of `x` coordinates for the plot. If `x` is a list, its components `x$x`, `x$y`, `x$z`, `x$vx`, and `x$vy` are used for `x`, `y`, `z`, `vx`, and `vy`, respectively.
`y`	numeric; a vector of `y` coordinates for the plot.
`z`	matrix; the state variable values to be plotted, NAs allowed (optional if `type = "p"`).
`vx`	numeric; a vector of arrow component lengths in the `x` direction (optional).
`vy`	numeric; a vector of arrow component lengths in the `y` direction (optional).
`type`	character; a 1-character string giving the type of plot desired. The following values are possible: "p" for points, "l" for level contour, "g" for grid contour.
`xlim`	numeric; a vector of `x` limits `(x1,x2)` for the plot.
`ylim`	numeric; a vector of `y` limits `(y1,y2)` for the plot.
`zlim`	numeric; a vector of `z` limits `(z1,z2)` for the plot.
`xlab`	character; the label for the `x` axis.
`ylab`	character; the label for the `y` axis.
`zlab`	character; the label for the `z` legend.
`asp`	numeric; the `y/x` aspect ratio.
`csi`	numeric; height of text characters in inches.
`width`	numeric; the width of the plotting window canvas in inches.
`pointsize`	integer; the point size of plotted text.
`cex.pts`	numeric; the amount by which point symbols should be magnified relative to the default.
`nlevels`	integer; number of contour levels desired.
`rkey`	logical; if `TRUE` the legend key is reversed with `z` values descending from top to bottom.
`color.palette`	function; a color palette to be used to assign colors in the plot.
`vuni`	logical; if `TRUE` all arrow lengths are set equal.
`vmax`	numeric; the maximum length of arrows in inches.
`vxby`	integer; increment for the sequence of arrows in the `x` direction.
`vyby`	integer; increment for the sequence of arrows in the `y` direction.
`axis.side`	integer; the side of the plot the axis is to be drawn on. The axis is placed as follows: 1 = below, 2 = left, 3 = above and 4 = right.
`minor.ticks`	logical; if `TRUE` minor tickmarks are added to the plot; its default is `FALSE`.
`ticks.inside`	logical; if `TRUE` tickmarks are placed inside the plot region; its default is `FALSE`.
`add.contour.lines`	logical; if `TRUE` and `type` is either "l" or "g" than contour lines are drawn; its default is `FALSE`.

Author(s)

Fisher, J.C.

See Also

filled.contour, image, arrows, AddAxis

Examples

```
data(project)

d <- Data("data.pts")
Plot2d(d, type = "p")

d <- Data("data.grd")
Plot2d(d, type = "l")
Plot2d(d, type = "g")
```

Plot3d *Plot Surface using OpenGL*

Description

Draws a three-dimensional (3D) surface plot.

Usage

```
Plot3d(x = NULL, y = NULL, z = NULL,
       px = NULL, py = NULL, pz = NULL,
       xlim = NULL, ylim = NULL, zlim = NULL,
       vasp = NA, hasp = NA, width = 7, ppi = 96,
       cex.pts = 1, nlevels = 20,
       color.palette = terrain.colors,
       mouse.mode = c("trackball", "zAxis", "zoom"),
       bg = "white")
```

Arguments

`x, y`	numeric; locations of grid lines at which the values in `z` are measured. These must be in ascending order. If `x` is a list, its components `x$x` and `x$y` are used for `x` and `y`, respectively. If the list has component `x$z` this is used for `z`.
`z`	matrix; the values to be plotted.
`px`	numeric; a vector of `x` coordinates for points in the plot. If `px` is a list, its components `px$px`, `px$py` and `px$pz` are used for `px`, `py` and `pz`, respectively.
`py`	numeric; a vector of `y` coordinates for points in the plot.
`pz`	numeric; a vector of `z` coordinates for points in the plot.
`xlim`	numeric; a vector of `x` limits `(x1,x2)` for the plot.
`ylim`	numeric; a vector of `y` limits `(y1,y2)` for the plot.
`zlim`	numeric; a vector of `z` limits `(z1,z2)` for the plot.
`vasp`	numeric; the `z/x` aspect ratio.
`hasp`	numeric; the `y/x` aspect ratio.
`width`	numeric; the width of the plotting window canvas in inches.

`ppi`	integer; screen resolution in points per inch.
`cex.pts`	numeric; the amount by which point symbols should be magnified relative to the default.
`nlevels`	integer; number of contour levels desired.
`color.palette`	function; a color palette to be used to assign colors in the plot.
`mouse.mode`	character; a vector of 3 strings describing what the 3 mouse buttons do, see `par3d`.
`bg`	character; the primary color for the background.

Details

The interpolated surface data is rendered using **rgl**, a 3D visualization device system for R based on OpenGL. The mouse is used for interactive viewpoint navigation where the left, right, and center mouse buttons rotate the scene, rotate the scene around the x-axis, and zooms the display, respectively.

Author(s)

Fisher, J.C.

See Also

`surface3d`, `points3d`

Examples

```
data(project)
d <- Data("data.grd")
Plot3d(d)
rgl.quit()
```

`PlotTimeSeries` *Plot Temporal Data*

Description

Draws a time-series plot with points and connecting lines.

Usage

```
PlotTimeSeries(x, y = NULL, xlim = NULL, ylim = NULL, ylab = NULL,
               tgap = NULL, width = 7, cex.pts = 1,
               pointsize = 12, fmt = NULL, axis.side = 1:2,
               minor.ticks = FALSE, ticks.inside = FALSE)
```

Arguments

`x`	POSIXct; a vector specifying `x` values.
`y`	numeric; a vector specifying `y` values.
`xlim`	POSIXct; the `x` limits `(x1,x2)` of the plot.
`ylim`	numeric; the `y` limits `(y1,y2)` of the plot.
`ylab`	character; the label for the `y` axis.
`tgap`	numeric; time gap exceedance level in seconds.
`width`	numeric; the width of the plotting window canvas in inches.
`cex.pts`	numeric; the amount by which point symbols should be magnified relative to the default.
`pointsize`	integer; the point size of plotted text.
`fmt`	character; date-time format for `x` axis tic mark labels, see `strptime`.
`axis.side`	integer; the side of the plot the axis is to be drawn on. The axis is placed as follows: 1 = below, 2 = left, 3 = above and 4 = right.
`minor.ticks`	logical; if `TRUE` minor tickmarks are added to the plot; its default is `FALSE`.
`ticks.inside`	logical; if `TRUE` tickmarks are placed inside the plot region; its default is `FALSE`.

Details

Line segments will not be drawn where time differences between consecutive points are greater than `tgap`.

Author(s)

Fisher, J.C.

See Also

`plot`

Examples

```
data(project)
d <- Data("data.pts")
PlotTimeSeries(x = d$t, y = d$z)
PlotTimeSeries(x = d$t, y = d$z, tgap = 3000)
```

`ProcessData` *Process Data*

Description

This function performs data processing on the state variables.

Usage

```
ProcessData()
```

Details

The raw data being processed is queried from the `data.raw` component of `Data`. Processing control parameters are also queried from `Data` and include: `cols`, `vars`, `lim.data`, `poly`, `poly.data`, `poly.crop`, `grid.dx`, `grid.dy`, `mba.m`, `mba.n`, and `mba.h`.

A data frame based on the user-defined state variables is first created. Records outside the user-defined spatial and temporal domains are then removed using: (1) data limits, where the `x`, `y`, `z`, and `t` limits are specified in the `xlim`, `ylim`, `zlim`, and `tlim` components of the `lim.data` list, respectively; and (2) a two-dimensional polygon defining the spatial domain within the `xy`-plane.

An interpolated grid of `z` values in constructed using a Multilevel B-spline approximation. The spatial extent of the interpolated surface is constrained using `polyLimit`, a polygon that sets spatial limits for grided data; where `z` values corresponding to grid nodes located outside this polygon are set to `NA`.

Value

Sets the `data.pts` and `data.grd` components of `Data`. The `data.pts` component is a data frame with variables:

`x, y`	numeric; a vector of `x` and `y` coordinates.
`z`	numeric; a vector of state variable values (optional).
`t`	POSIXct; a vector of time stamps (optional).
`vx, vy`	numeric; a vector of velocity components in the `x` and `y` directions, respectively (optional).

The `data.grd` component is a list with components:

`x, y`	numeric; a vector of grid line locations at which the values in `z` are measured.
`z`	matrix; interpolated surface of state variable with rows and columns corresponding to grid lines in the `x` and `y` directions, respectively.
`vx, vy`	matrix; interpolated surface of velocity components with rows and columns corresponding to grid lines in the `x` and `y` directions, respectively (optional).

Row names in `data.pts` coincide with row indexes in `data.raw`.

Author(s)

Fisher, J.C.

See Also

`mba.points`

Examples

```
data(project)
ProcessData()
```

`ReadData` *Read Data*

Description

Reads table formatted data from a connection and creates a data frame from it.

Usage

```
ReadData(con, headers = c(FALSE, FALSE, FALSE), sep = "\t",
         quote = "\"'", nrows = -1, na.strings = c("", "NA"),
         skip = 0, comment.char = "#", encoding = getOption("encoding"))
```

Arguments

`con`	connection; a `connection` object.
`headers`	logical; a vector of length three that indicates whether the data table contains header lines: see 'Details'.
`sep`	character; the field separator string. Values on each line of the file are separated by this string.
`quote`	character; the set of quoting characters.
`nrows`	integer; the maximum number of rows to read in. Negative and other invalid values are ignored (optional).
`na.strings`	character; a vector of strings which are to be interpreted as `NA` values. Blank fields are also considered to be missing values.
`skip`	integer; the number of lines to skip before beginning to read data.
`comment.char`	character; a vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.
`encoding`	character; encoding to be assumed for input strings. If the value is `"latin1"` or `"UTF-8"` it is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.

Format

The table formatted data is required to have at least two numeric variables. Measurement units associated with date and time values are based on format character strings described in `strptime`, for example "02/26/2010 02:05:39 PM" is represented using "%d/%m/%Y %I:%M:%S %p".

Details

This function is the primary method for importing table formatted data. The `headers` argument, a logical vector of length three, indicates whether the file contains the names, measurement units, and decimal precision of variables as its initial lines of text. For example, a `headers = c(TRUE, FALSE, TRUE)` indicates that the first and second lines will contain the names and decimal precision of variables, respectively; measurement units are not included. If `headers = c(FALSE, FALSE, FALSE)` (the default), no header information is contained within the data table.

Value

Returns a list with the following components:

`dat`	data.frame; a data table with headers and comments removed.
`cols`	list; of length equal to the current number of data variables. Each component in `cols` is linked to a specific variable (see `ManageData`).
`vars`	list; an initial guess of the state variables. Integer components `x`, `y`, `z`, and `t` specify the index number in `cols` that correspond to the respective state variable.

Author(s)

Fisher, J.C.

See Also

`read.table`

Examples

```
f <- system.file("extdata/DataExample.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
ans <- ReadData(con, headers = c(TRUE, TRUE, TRUE))
close(con)
```

`Rename` *Rename Values in Character Vector*

Description

A GUI for renaming values in a vector of character strings.

Usage

```
Rename(names = NULL, cur.name = NULL, win.title = NULL,
       parent = NULL)
```

Arguments

`names`	character; a vector of character strings.
`cur.name`	character; sets the combobox value, name must be included in `names` (optional).
`win.title`	character; the title of the main window (optional).
`parent`	tkwin; the parent window (optional).

Value

Returns a character vector with updated values of `names`.

Author(s)

Fisher, J.C.

Examples

```
Rename(names = c("Name1", "Name2", "Name3"), cur.name = "Name2")
```

`SetConfiguration` *Set Window and Plotting Parameters*

Description

A GUI for specifying window geometry and universal plotting parameters.

Usage

```
SetConfiguration(parent = NULL)
```

Arguments

`parent`	tkwin; the parent window (optional).

Value

Queries and sets the following components of `Data`:

`nlevels`	integer; approximate number of contour levels desired; its default is 20.
`width`	numeric; the width of the plotting window canvas in inches; its default is 7.
`cex.pts`	numeric; the amount by which point symbols should be magnified relative to the default value, `1.0`. For example, `cex.pts = 0.5` reduces the point symbol to half of its default size.
`asp.yx, asp.zx`	numeric; the `y/x` and `z/x` aspect ratios, respectively (optional).
`vmax`	numeric; the maximum length of arrows in inches (optional).
`vxby, vyby`	integer; increment for the sequence of arrows in the `x` and `y` directions, respectively (optional).
`tgap`	numeric; the time gap exceedance level in seconds. A break in the linear segments of the time-series plot will occur where differences between sequential temporal records is greater than `tgap`.
`rkey`	logical; if `TRUE` the legend key is reversed with `z` values descending from top to bottom; its default is `FALSE`.
`img.contour`	logical; if `TRUE` the `image` function is used to plot the interpolated surface; if `FALSE` (the default) the `filled.contour` function is used.
`show.arrows`	logical; if `TRUE` the vector arrows will be plotted; its default is `FALSE`.
`show.lines`	logical; if `TRUE` the line contours will be plotted on the two-dimensional interpolated surface; its default is `FALSE`.
`show.points`	logical; if `TRUE` the point values associated with `(x,y)` will be plotted on the interpolated surface; its default is `FALSE`.

`show.poly`	logical; if `TRUE` the polygons describing the spatial domain are added to the scatter plot and two-dimensional surface plot; its default is `FALSE`.
`vuni`	logical; if `TRUE` a constant arrow length specified by `vmax` is used; its default is `FALSE`.
`show.2.axes`	logical; if `TRUE` axes tickmarks will be drawn on all sides, its default is `FALSE`.
`minor.ticks`	logical; if `TRUE` minor tickmarks are added to the plot; its default is `FALSE`.
`ticks.inside`	logical; if `TRUE` tickmarks are placed inside the plot region; its default is `FALSE`.

Note

Re-importing data does not affect values specified in this GUI.

Author(s)

Fisher, J.C.

Examples

```
SetConfiguration()
```

`SetPolygonLimits` *Set Polygon Limits*

Description

A GUI for specifying polygon limits.

Usage

```
SetPolygonLimits(poly.names = NULL, poly.data = NULL,
                 poly.crop = NULL, parent = NULL)
```

Arguments

`poly.names`	character; the vector of names corresponding to polygons contained within `ply` (see `ManagePolygons`).
`poly.data`	character; the name of the polygon that defines the data limits boundary (optional).
`poly.crop`	character; the name of the polygon that defines the crop region for interpolated data (optional).
`parent`	tkwin; the parent window (optional).

Value

Returns a list with components `poly.data` and `poly.crop` (see ‘Arguments’).

Author(s)

Fisher, J.C.

See Also

`AutocropPolygon`, `tri.mesh`

Examples

```
SetPolygonLimits(c("Polygon1", "Polygon2", "Polygon3"))
```

`SetPreferences` *Set Data Preferences*

Description

A GUI for specifying the interpolation algorithms input parameters.

Usage

```
SetPreferences(parent = NULL)
```

Arguments

`parent`	tkwin; the parent window (optional).

Value

Queries and sets the following components in `Data`:

`grid.dx, grid.dy`	numeric; the grid spacing along the `x`- and `y`-axis for the interpolated surface, respectively (optional).
`mba.n, mba.m`	integer; initial size of the spline space in the hierarchical construction along the `x`- and `y`-axis, respectively (optional).
`mba.h`	integer; number of levels in the hierarchical construction, its default is `11`.

Note

If data is re-imported, parameters in this GUI are set to default values.

Author(s)

Fisher, J.C.

See Also

`mba.points`

Examples

```
SetPreferences()
```

`SummarizeData` *Summarize Object*

Description

A summary of the descriptive statistics of an array object.

Usage

```
SummarizeData(obj, digits = NULL, units = NULL)
```

Arguments

`obj`	an array object for which the summary is desired.
`digits`	integer; the decimal precision of the object (only pertains to objects of class dQuotenumeric).
`units`	integer; the date and time format, its default is "%Y-%m-%d %H:%M:%S".

Value

Results are dependent on the class of `obj`. Returns a list with the following components:

`Count`	integer; array length.
`NAs`	integer; number of `NA` values.
`Class`	character; the objects `class` attribute.
`Min., Max.`	numeric; extreme values with `NA` values ignored.
`1st Qu., Median, 3rd Qu.`	numeric; estimates of the underlying distribution quantiles with `NA` values ignored.
`Mean`	numeric; arithmetic mean with `NA` values ignored.
`St.Dev.`	numeric; standard deviation with `NA` values ignored.
`Sum`	numeric; sum with `NA` values ignored.
`Hist`	histogram; an object of class histogram, see `hist` documentation for details.
`Unique`	integer; number of unique factors.
`TRUE, FALSE`	integer; number of `TRUE` and `FALSE` values, respectively.
`String`	character; a formatted text summary of the descriptive statistics.
`Time Per.`	character; a formatted time duration.

Author(s)

Fisher, J.C.

See Also

`quantile`, `hist`

Examples

```
summary(attenu$dist, digits = 4)
SummarizeData(attenu$dist, digits = 4)
```

ViewData — *View Data*

Description

A GUI for viewing table formatted data.

Usage

```
ViewData(d, col.names = NULL, col.units = NULL, col.digs = NULL,
         parent = NULL)
```

Arguments

`d`	data.frame; data used to populate the table.
`col.names`	character; a vector giving the column names for the data table (optional).
`col.units`	character; a vector giving the measurement units for each column of the data table (optional).
`col.digs`	integer; a vector giving the decimal places of numeric columns in the data table (optional).
`parent`	tkwin; the parent window (optional).

Details

Column titles are a concatenation of variables `col.names` and `col.units`. Row titles are taken from the row names attribute of the data frame.

Note

Requires the Tcl package Tktable.

Author(s)

Fisher, J.C.

See Also

`tclArray`, `row.names`

Examples

```
tclRequire("Tktable", warn = TRUE)

n <- 1000
V1 <- sample(c(1:9, NA), n, replace = TRUE)
V2 <- sample(LETTERS, n, replace = TRUE)
V3 <- as.POSIXct(rnorm(n, mean = 0, sd = 1e6), origin = "2010-01-01")
V4 <- sample(V1 * pi, n)
d <- data.frame(V1, V2, V3, V4)
col.names <- c("Integers", "Letters", "POSIXt", "Numeric")
col.units <- c("units", NA, "%m/%d/%Y %H:%M", NA)
col.digs <- c(0, NA, NA, 3)
```

```
ViewData(d, col.names, col.units, col.digs)

row.names(d) <- 1:n + n
ViewData(d, col.names, col.units = NULL, digs)
```

`WriteFile` *Write Data File*

Description

Exports post-processed data to a file.

Usage

```
WriteFile(ext = "txt")
```

Arguments

`ext`	character; the default `file` extension.

Value

The format and type of data written is based on the file type chosen within the file management pop-up dialog box. A selection of *Text Files* (‘`*.txt`’, ‘`*.csv`’, ‘`*.dat`’) writes the contents of `data.pts` to a text file; selecting *Interpolated Grid Test Files* (‘`*.grd`’) writes the contents of `data.grd` to a text file; and a selection of *ESRI Shapefiles* (‘`*.shp`’) writes the contents of `data.pts` to a binary file.

Author(s)

Fisher, J.C.

See Also

`GetFile`, `write.table`, `writeOGR`

Examples

```
data(project)
WriteFile("txt")
```

Index