Package 'RSurvey'

August 29, 2016

Version 0.8-3

Date 2015-02-20

Title Analysis of Spatially Distributed Data

Author Jason C. Fisher

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

Depends R (>= 3.1.0)

Imports tcltk, sp, rgeos, MBA

Suggests rgdal, tripack, colorspace, dichromat, rgl, XML

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

Description 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 https://github.com/jfisher-usgs/RSurvey

BugReports https://github.com/jfisher-usgs/RSurvey/issues

ByteCompile yes

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-21 01:02:54

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.

Note

The **RSurvey** GUI requires R operate as an SDI application, using multiple top-level windows for the console, graphics, and pager.

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

Examples

```
## Not run: 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, see axis.

Details

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

Author(s)

J.C. Fisher

See Also

axis, axis.POSIXct

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 gpc.poly-class.

Author(s)

J.C. Fisher

See Also

AutocropRegion

Examples

```
## Not run:
data(tritest, package = "tripack")
mesh <- tripack::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))

## End(Not run)
```

`AutocropRegion` *Set Autocrop Input Parameters*

Description

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

Usage

```
AutocropRegion(d, parent = NULL, ...)
```

Arguments

`d`	data.frame; a data table with required coordinate components `d$x` and `d$y`.
`parent`	tkwin; the GUI parent window.
`...`	other graphical parameters, see `Plot2d`.

Details

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 user defined input parameters.

Value

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

Author(s)

J.C. Fisher

Examples

```
## Not run:
data(tritest, package = "tripack")
AutocropRegion(as.data.frame(tritest), asp = 1)

## End(Not run)
```

`BuildHistogram` *Set Histogram Input Parameters*

Description

A GUI for specifying input parameters of the `hist` function.

Usage

```
BuildHistogram(d, var.names = NULL, var.default = 1L, parent = NULL)
```

Arguments

`d`	list, data.frame, matrix, or numeric; vector(s) of values for which the histogram is desired.
`var.names`	character; names corresponding to each vector (column) in `d`.
`var.default`	character or integer; vector name or index in `d`, defaults to `1L`.
`parent`	tkwin; the GUI parent window.

Details

Plot histogram and view results.

Value

Does not return any value.

Author(s)

J.C. Fisher

See Also

`hist`, `plot.histogram`

Examples

```
## Not run:
data(ex.project)
BuildHistogram(ex.project$data.pts)

## End(Not run)
```

CheckEntry *Control Content in Entry Widget*

Description

Content control for character strings based object class.

Usage

```
CheckEntry(obj.class, ent.str = "")
```

Arguments

obj.class	character; object class.
ent.str	character; entry value.

Details

The allowed object classes include: *real*, *integer*, and *logical*.

Value

A character string with strict adherence to the format for the object class.

Author(s)

J.C. Fisher

Examples

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

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

ChooseColor *Set Graphic Color*

Description

A GUI for selecting a graphic color.

Usage

```
ChooseColor(col, parent = NULL)
```

Arguments

col	character; the initial color, see 'Value'.
parent	tkwin; the GUI parent window.

Value

Returns a selected color in terms of its RGB components with a string of the form "#RRGGBB" where each of the pairs RR, GG, BB consist of two hexadecimal digits giving a value in the range 00 to FF.

Author(s)

J.C. Fisher

See Also

col2rgb

Examples

```
## Not run: ChooseColor(col = "#669933")
```

ChoosePch *Set Graphic Symbol*

Description

A GUI for selecting a graphic symbol (pch).

Usage

```
ChoosePch(pch = NA, parent = NULL)
```

Arguments

pch	numeric or character; the initial selection of pch.
parent	tkwin; the GUI parent window.

Value

Returns a numeric or integer value based on a user selection.

Author(s)

J.C. Fisher

See Also

points

Examples

```
## Not run: ChoosePch(pch = "+")
```

`CutoutPolygon` *Exclude Gridded Data Outside Polygon*

Description

This function excludes gridded data located outside 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, y`	numeric; a vector of `x` and `y` coordinates.
`z`	matrix; the state variable corresponding to coordinates in the grid.

Author(s)

J.C. Fisher

See Also

`point.in.polygon`

Examples

```
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)})

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 parameters and their attributes.

Usage

```
Data(option, value, which.attr = NULL, clear.proj = FALSE, clear.data = FALSE,
     replace.all = NULL)
```

Arguments

option	character; the parameter name, see 'Parameters'.
value	a parameter value specified for option (optional).
which.attr	character, a non-empty character string specifying which attribute is to be accessed.
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.
replace.all	list; a replacement list of parameter values.

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. Default values are specified internally within this function.

Data

Imported raw data is saved to the data frame data.raw, see ImportText. Processed point data is saved to the data frame data.pts and interpolated surface data is saved 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)

J.C. Fisher

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 remove all saved parameter values
Data(replace.all = list())
# To recover saved parameter values
Data(replace.all = d)
```

`EditData` *Edit Data*

Description

A GUI for viewing and editing table formatted data.

Usage

```
EditData(d, col.names = names(d), row.names = NULL, col.formats = NULL,
         read.only = FALSE, changelog = NULL, win.title = "Data",
         parent = NULL)
```

Arguments

`d`	list, matrix, or data.frame; the data used to populate the table.
`col.names`	character; a vector of column names.
`row.names`	character; a vector of row names.
`col.formats`	character; a vector of format conversion specification strings, see `sprintf` and `strftime`.
`read.only`	logical; specifies whether the table is in read only mode.
`changelog`	data.frame; history of all data table edits (see 'Value').
`win.title`	character; a string to display as the title of the dialog box.
`parent`	tkwin; the GUI parent window.

Details

Row titles are taken from the row names attribute of `d`. Pattern searches are performed using `grep`. Edits are reflected in the `changelog`.

Value

NULL is returned if no edits were made; otherwise, new values of d and changelog are returned as components in a list. The changelog data frame contains the following variables:

timestamp	POSIXct; identifies when the edit event occurred.
record	character; row name.
variable	character; column name.
old	character; value before editing.
new	character; value after editing.

Note

Requires the Tcl package Tktable.

Author(s)

J.C. Fisher

See Also

BuildHistogram

Examples

```
## Not run:
tcltk::tclRequire("Tktable", warn = TRUE)

n <- 1000L
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", "DateTime", "Numeric")
col.formats <- c("%d", "%s", "%m/%d/%Y %H:%M", "")
ans <- EditData(d, col.names, col.formats)
str(ans)

rownames(d) <- paste0(sample(LETTERS, n, replace = TRUE), seq_len(n))
EditData(d, read.only = TRUE)

colnames(d) <- NULL
rownames(d) <- NULL
EditData(d, read.only = TRUE)

## End(Not run)
```

EditFunction *Edit Function*

Description

A GUI for defining functions in the R language.

Usage

```
EditFunction(cols, index = NULL, fun = NULL, value.length = NULL,
             value.class = NULL, win.title = "Edit Function", parent = NULL)
```

Arguments

cols	list; see ManageVariables.
index	integer; an element index number in cols.
fun	character; existing function, only used if index=NULL.
value.length	integer; the required length for the evaluated function.
value.class	character; the required class for the evaluated function.
win.title	character; a string to display as the title of the dialog box.
parent	tkwin; the GUI parent window.

Details

This GUI is appropriate for deriving new variables in a pre-existing data frame or query building.

Value

Returns a list with two four components: fun, the user defined function (when evaluated, this string must be parseable); class, the object class for the evaluated function; summary, the default summary for the evaluated function; and sample, the first non-missing value for the evaluated function.

Author(s)

J.C. Fisher

See Also

EvalFunction

Examples

```
## Not run:
d <- list(x = 1:10, y = 10:1)
Data("data.raw", d)
cols <- list()
cols[[1]] <- list(id = "X", index = 1, fun = "\"X\"")
cols[[2]] <- list(id = "Y", index = 2, fun = "\"Y\"")
cols[[3]] <- list(id = "New Variable", fun = "\"X\" + \"Y\"")
EditFunction(cols, index = 3)

## End(Not run)
```

`EditText` *Edit Text*

Description

A GUI for viewing and editing text.

Usage

```
EditText(txt, read.only = FALSE, win.title = "View Text",
         is.fixed.width.font = FALSE, parent = NULL)
```

Arguments

`txt`	character; the data used to populate the table.
`read.only`	logical; specifies whether the text is read only.
`win.title`	character; a string to display as the title of the dialog box.
`is.fixed.width.font`	logical; should a fixed-width font be used?
`parent`	tkwin; the GUI parent window.

Author(s)

J.C. Fisher

Examples

```
## Not run:
txt <- c("\"Hills cherish the ambition",
         "    to turn into partial",
         "   differential equations\"",
         "",
         "        -Donald Hall")
new.txt <- EditText(txt, is.fixed.width.font = TRUE)

EditText(txt, read.only = TRUE)
```

```
## End(Not run)
```

`EvalFunction` *Parse and Evaluate an RSurvey Expression*

Description

Parses and evaluates a character string representation of an RSurvey expression.

Usage

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

Arguments

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

Value

The result of evaluating the text epression.

Author(s)

J.C. Fisher

See Also

`parse`, `eval`

Examples

```
d <- list(x = 1:10, y = 10:1)
Data("data.raw", d)
cols <- list()
cols[[1]] <- list(id = "X", index = 1, fun = "\"X\"")
cols[[2]] <- list(id = "Y", index = 2, fun = "\"Y\"")
EvalFunction("\"Y\"", cols)
EvalFunction("\"X\" + \"Y\"", cols)
EvalFunction("rnorm(12)", cols)
```

`ex.data` *Example Data Set*

Description

This is an example data set that is typically used to test **RSurvey**.

Usage

```
ex.data
```

Format

A data frame with 50 rows and the following columns:

T.date.time POSIXct; vector of date-time values.

X.numeric..m numeric; vector.

Y.numeric..m numeric; vector.

Z.numeric..ft numeric; vector.

Z.numeric..ft..1. numeric; vector.

Z.numeric..m numeric; vector.

Unknown numeric; vector.

Logical integer; vector.

Integer..psia integer; vector.

Date Date; vector.

Time POSIXct; vector.

Character factor; vector.

Examples

```
## Not run:
data(ex.data)
summary(ex.data)

## End(Not run)
```

ex.project *Example Project*

Description

This is an example project for **RSurvey**.

Usage

```
ex.project
```

Format

A list containing typical components in a project; these components are described throughout the **RSurvey** documentation.

Examples

```
## Not run:
data(ex.project)
str(ex.project)

# Programmatically load a project into RSurvey using the following command:
Data(replace.all = ex.project)

## End(Not run)
```

ExportData *Export Data*

Description

A GUI for exporting data to text files, shapefiles, or R data files.

Usage

```
ExportData(file.type = "txt", parent = NULL)
```

Arguments

file.type	character; the output file type: either "txt" for *Text Files*, "rda" for *R Data Files*, or "shp" for *ESRI Shapefiles*.
parent	tkwin; the GUI parent window.

Value

Saves the GUI options in the `export` component of `Data`. List components of `export` include:

`processed`	logical; are exported records limited to processed data?
`fmts`	logical; indicates whether a header line of conversion specification format strings is written (text only).
`cols`	logical; indicates whether a header line of column names is written (text only).
`rows`	logical; indicates whether the row names are written (text only).
`comment`	logical; indicates whether to write comments using the comment character, `com` (text only).
`sep`	character; the field separator character (text only).
`dec`	character; string used for decimal points (text only).
`nas`	character; string interpreted as `NA` value (text only).
`com`	character; comment character (text only).
`qmethod`	character; a string specifying how to deal with embedded double quote characters when quoting strings (text only).
`quote`	logical; if `TRUE`, any character or factor columns will be surrounded by double quotes (text only).
`encoding`	character; declares the encoding to be used on the file (text only).
`eol`	character; the character to print at the end of each line (text only).
`zip`	character; indicate whether the file should be compressed using gzip, bzip2, or xz (text only).
`changelog`	logical; indicate if a separate text file should be written with the change log (text only).
`ascii`	logical; if `TRUE`, an ASCII representation of the data is written (R data only).

Author(s)

J.C. Fisher

See Also

`write.table`, `save`, `writeOGR`

Examples

```
## Not run:
data(ex.project)
Data(replace.all = ex.project)
ExportData(file.type = "txt")

## End(Not run)
```

Format *Build C-Style String Formats*

Description

A GUI for the system `sprintf` C-library function.

Usage

```
Format(sample = pi, fmt = "", parent = NULL)
```

Arguments

`sample`	logical, integer, numeric, character, or factor; a sample value.
`fmt`	character; the conversion specification format, see `sprintf`.
`parent`	tkwin; the GUI parent window.

Value

Returns a character string.

Author(s)

J.C. Fisher

See Also

`sprintf`, `format`

Examples

```
## Not run:
Format(sample = pi, fmt = "%3.8f")
Format(sample = 3L)
Format(sample = TRUE)
Format(sample = "string")

## End(Not run)
```

FormatDateTime *Build Date-Time String Formats*

Description

A GUI for converting between character representations and objects of class "POSIXt" or "Date".

Usage

```
FormatDateTime(sample = as.POSIXct("1991-08-25 20:57:08"), fmt = "",
               parent = NULL)
```

Arguments

sample	POSIXt or Date; a time object.
fmt	character; the format conversion specification string for time values.
parent	tkwin; the GUI parent window.

Value

Returns a character string representing the formatted date-time value.

Author(s)

J.C. Fisher

See Also

strptime, format

Examples

```
## Not run:
new.fmt <- FormatDateTime(fmt = "%A %B %d %I:%M %p")
FormatDateTime(Sys.Date())

## End(Not run)
```

GetBitmapImage *Create Icon Bitmap Image*

Description

Create a small TK bitmap image.

Usage

```
GetBitmapImage(type)
```

Arguments

type	character; icon image type, see 'Details'.

Details

Icon image types include: *left*, *right*, *up*, *down*, *top*, *bottom*, *upleft*, *upright*, *downleft*, *downright*, *next*, *previous*, *copy*, *paste*, *find*, *delete*, *view*, *info*, *plus*, *minus*, *print*, and *histogram*. A recommended editor for bitmap design is Paul Obermeier's poBitmap tool.

Value

An image of class tclObj.

Author(s)

J.C. Fisher

See Also

tkimage.create

Examples

```
## Not run:
types <- c("left", "right", "up", "down", "top", "bottom", "upleft", "upright",
           "downleft", "downright", "next", "previous", "copy", "paste", "find",
           "delete", "view", "info", "plus", "minus", "print", "histogram")
Fun <- function(k) print(types[k])
tt <- tcltk::tktoplevel(padx = 50, pady = 50)
i <- 0
j <- 0
d <- 5
for (k in seq_along(types)) {
  img <- paste("img", k, sep = ".")
  but <- paste("but", k, sep = ".")
  assign(img, GetBitmapImage(types[k]))
  assign(but, tcltk::ttkbutton(tt, width = 2, image = get(img),
```

```
                                command = local({k <- k; function() Fun(k)})))
  tcltk::tkgrid(get(but), row = i, column = j, padx = 5, pady = 5)
  i <- k %/% d
  j <- ifelse(j < d - 1, j + 1, 0)
}

## End(Not run)
```

GetFile *Select 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 file name which the data are to be read from. Alternatively, `file` can be a readable text-mode `connection`.
`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 file name to be displayed in the dialog when it pops up.
`defaultextension`	character; the string that will be appended to the file name if the user enters a file name 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 GUI parent window.

Value

If `multi` is `FALSE` returns the file path as a charcter object with the following attributes:

`directory`	character; the directory that contains the file
`name`	character; the file name
`extension`	character; the file extension
`type`	character; the file type

Otherwise, a list is returned containing a character object for each file.

Author(s)

J.C. Fisher

Examples

```
## Not run: GetFile()
```

`ImportPackage` *Import Data from R Package*

Description

A GUI for loading selected data sets from R packages.

Usage

```
ImportPackage(classes = NULL, parent = NULL)
```

Arguments

`classes`	character; the object classes of data sets that can be loaded. Set to `NULL` to enable loading for all object classes.
`parent`	tkwin; the GUI parent window.

Value

Returns a list with the following components:

`d`	data.frame or matrix; data set.
`src`	character; a vector of length 3 that includes the dataset name, package name, and access date.

Author(s)

J.C. Fisher

See Also

`data`

Examples

```
## Not run: ans <- ImportPackage(c("data.frame", "matrix"))
```

ImportSpreadsheet *Import Data from XML Spreadsheet File*

Description

A GUI for loading selected data sets from an Open XML Spreadsheet file (*.xlsx).

Usage

```
ImportSpreadsheet(parent = NULL)
```

Arguments

parent	tkwin; the GUI parent window.

Value

Returns a list with the following components:

d	data.frame; data set.
src	character; a vector of length 2 that includes the pathname of the spreadsheet file and access date.

Author(s)

J.C. Fisher

References

This code was derived with permission from Schaun Wheeler's xlsxToR function.

Examples

```
## Not run: ans <- ImportSpreadsheet()
```

ImportText *Import Data from Text File*

Description

A GUI for reading table formatted data from a text file.

Usage

```
ImportText(parent = NULL)
```

Arguments

parent	tkwin; the GUI parent window.

Details

This GUI is a wrapper for the read.table function. Data connections are defined as the path to the file to be opened, a complete URL (e.g. http://, ftp:// or file:///), or windows clipboard. Files are limited to text format (e.g., '.tsv' '.csv', or '.txt'); however, they can be compressed by gzip, bzip2, or xz with additional extension '.gz', '.bz2', or '.xz', respectively.

Conversion specification formats are the character representation of object types used to: identify column classes prior to reading in data, and format values for printing. Conversion specifications are based on C-style string formatting commands for numeric, integer, and character object classes, see sprintf; for example, a format string of "%.5f" applied to the mathematical constant *pi* results in "3.14159". Calendar date and time objects of class POSIXct are defined by the ISO C99 / POSIX standard, see strftime; for example, "02/26/2010 02:05:39 PM" is represented using "%d/%m/%Y %I:%M:%S %p".

Comments located above data records and header lines are preserved; all other comments are ignored. Requires the specification of a comment character.

Performance issues associated with reading in large files can be alleviated by specifying formats in a header line, and giving the maximum number of rows to read in.

Value

Sets the following components in Data:

data.raw	data.frame; the imported (or raw) data table.
cols	list; length equal to the current number of data variables. Each component in cols is linked to a specific variable, see ManageVariables.
comment	character; a vector of comment strings.
import	list; saved GUI options.

Components of import include:

source	character; a vector of length 2 that includes the pathname of the text file and access date.
fmts	logical; indicates whether the file contains the conversion specification format strings of the variables.
cols	logical; indicates whether the file contains the names of the variables.
skip	integer; the number of lines skiped before data is read.
sep	character; the field separator string.
dec	character; used in the file for decimal points.
na	character; string interpreted as NA values.
quote	character; the set of quoting characters.
comment	character; comment character.
encoding	character; encoding that was assumed for input strings, see Encoding.
str.as.fact	logical; if TRUE, character variables are converted to factors.

Note

Requires the Tcl package Tktable.

Author(s)

J.C. Fisher

See Also

`read.table`

Examples

```
## Not run: ImportText()
```

LoadPackages *Load Suggested Packages*

Description

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

Usage

```
LoadPackages()
```

Author(s)

J.C. Fisher

See Also

`install.packages`, `require`

Examples

```
## Not run: LoadPackages()
```

ManagePolygons *Manage Polygons*

Description

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

Usage

```
ManagePolygons(polys = NULL, poly.data = NULL, poly.crop = NULL,
               encoding = getOption("encoding"), parent = NULL)
```

Arguments

polys	list; its components are objects of gpc.poly-class.
poly.data	character; the name of the polygon that defines the data limits boundary.
poly.crop	character; the name of the polygon that defines the crop region for interpolated data.
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 GUI parent window.

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 **rgeos** package for details.

Value

A list with components polys, poly.data, and poly.crop (see 'Arguments').

Author(s)

J.C. Fisher

See Also

polyfile, union, setdiff, intersect

Examples

```
## Not run: ManagePolygons()
```

ManageVariables *Manage Variables*

Description

A GUI for managing variables in the data table.

Usage

```
ManageVariables(cols, vars, query, changelog, parent = NULL)
```

Arguments

cols	list; see 'Value'.
vars	list; see 'Value'.
query	character; see 'Value'.
changelog	data.frame; see 'Value'.
parent	tkwin; the GUI parent window.

Details

This GUI lets you: (1) specify the names and format of variables; (2) add new variables based on user defined functions, see EditFunction; (3) display data in a spreadsheet, see EditData; and (4) remove and (or) reorder variables in the data table.

Value

Returns a list with components cols and vars. 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.
format	character; the conversion specification format (optional).
id	character; a unique identifier that is created from name.

fun	character; the expression evaluated when computing the variables vector of values.
index	integer; the variables component index number in the data.raw data frame, see ImportText. Only required for variables directly linked to data columns in data.raw.
class	character; the data class of the vector object.
summary	summaryDefault; a summary of the variables descriptive statistics (see summary).
comments	character; user comments.

The vars object is a list with components:

x, y, z, vx, vy, sort.on	integer; the index number of the corresponding state variable in cols. These indexes are updated to reflect the removal and (or) reordering of variables in cols.
query	character; if required, variable names are updated.
changelog	data.frame; if required, names in variable component are updated.

Author(s)

J.C. Fisher

Examples

```
## Not run:
data(ex.project)
Data(replace.all = ex.project)
ManageVariables(ex.project$cols, ex.project$vars, ex.project$query, ex.project$changelog)

## End(Not run)
```

OpenRSurvey *Open RSurvey*

Description

Opens the main graphical user interface (GUI) for **RSurvey**.

Usage

```
OpenRSurvey()
```

Details

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

Value

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

x, y, z, vx, vy
: integer; the index number of the corresponding state variable in cols, see ManageVariables.

Author(s)

J.C. Fisher

Examples

```
## Not run: 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,
       rm.pnt.line = FALSE)
```

Arguments

x	numeric; a vector of x coordinates for the plot. If x is a data frame, its components xx, xy, xz, xvx, and x$vy are used for x, y, z, vx, and vy, respectively.
y	numeric; a vector of y coordinates for the plot.
z	numeric or matrix; the state variable values to be plotted, NAs allowed. A matrix is required for contour plots.
vx, vy	numeric; a vector of arrow component lengths in the x and y directions.
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, ylab	character; the label for the x and 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; its default is FALSE.
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; its default is FALSE.
vmax	numeric; the maximum length of arrows in inches.
vxby, vyby	integer; increment for the sequence of arrows in the x and 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.
rm.pnt.line	logical; if TRUE the line boundary on point symbols is not drawn; its default is FALSE.

Details

The length of x and y should be equal to the nrow(z) and ncol(z), respectively.

Author(s)

J.C. Fisher

See Also

filled.contour, image, arrows, AddAxis

Examples

```
data(ex.project)

d <- ex.project$data.pts
Plot2d(d, type = "p")

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

Plot3d *Plot 3D 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$z this is used for z.
z	matrix; the values to be plotted. The number of rows and columns should be equal to the length(x) and length(y), respectively.
px	numeric; a vector of x coordinates for points in the plot. If px is a list, its components pxpx, pxpy and px$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 background color.

Details

The interpolated surface 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)

J.C. Fisher

Examples

```
## Not run:
data(ex.project)
Plot3d(ex.project$data.grd)
rgl::rgl.quit()

## End(Not run)
```

POSIXct2Character *Convert POSIXct to Character*

Description

Convert objects from POSIXct to character class.

Usage

```
POSIXct2Character(x, fmt = "%Y-%m-%d %H:%M:%OS3")
```

Arguments

x	POSIXct; vector of calendar dates and times.
fmt	character; the conversion specification format.

Value

Returns a character vector representing time.

Note

R incorrectly formats objects of calss `POSIXct` with fractional seconds. For example, a POSIXct time with fractional part `.3` seconds (stored as `0.29999`) is printed as `.2` when represented with one decimal digit. Note that the fractional part on outputs is not rounded. Decimal precision is down to milliseconds on Windows, and down to (almost) microseconds on the other operating systems.

Author(s)

J.C. Fisher

References

A detailed explanation of the problem is provided here; solution provided here.

See Also

`strptime`

Examples

```
txt <- c("11/10/2011 07:49:36.3", "04/01/2013 17:22:08.123",
         "01/06/2013 01:02:16.123", "12/14/2038 15:42:04.123456")
date.time <- as.POSIXct(txt, format = "%m/%d/%Y %H:%M:%OS")

options("digits.secs" = 3)
format(date.time, fmt = "%d/%m/%Y %H:%M:%OS")
format(date.time, fmt = "%d/%m/%Y %H:%M:%OS3")

POSIXct2Character(date.time, fmt = "%d/%m/%Y %H:%M:%OS3")
POSIXct2Character(date.time, fmt = "%d/%m/%Y %H:%M:%OS4")
POSIXct2Character(date.time, fmt = "%d/%m/%Y %H:%M:%OS2")

POSIXct2Character(date.time, fmt = "%H:%M:%OS3 %Y-%m-%d")
```

`ProcessData` *Process Data*

Description

This function performs data processing on the state variables.

Usage

```
ProcessData(d, type = "p", coerce.rows = NULL, ply = NULL,
            grid.res = list(x = NA, y = NA),
            grid.mba = list(n = NA, m = NA, h = 11))
```

Arguments

d	data.frame; the data to be processed, a table with variables x, y, z, t, vx, vy, and vz.
type	character; a 1-character string giving the resulting output type. The following values are possible: "p" for points and "g" for interpolated grid.
coerce.rows	logical; a vector indicating rows in d to include in the processed data set. NAvalues are considered FALSE.
ply	gpc.poly; if type = "p" a polygon that defines either the data limits, else if type = "g" a crop region for gridded data.
grid.res	list; numeric components giving the grid spacing along the x- and y-axis for interpolated values, see SetInterpolation.
grid.mba	list; integer components giving the input parameters for the multilevel B-splines algorithm, see SetInterpolation.

Details

Any row in the d data table with NA values for either x or y is removed. Rows are also removed where coerce.rows is FALSE. The spatial domain may be limited using a polygon defined in the xy-plane. Interpolated grid values corresponding to grid nodes located outside the polygon boundary are set to NA.

Value

If type = "p" returns a data frame with variables:

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

If type = "g" returns 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, vz	matrix; interpolated surface of velocity components with rows and columns corresponding to grid lines in the x and y directions (optional).
vf	numeric; volumetric flux (optional).

Author(s)

J.C. Fisher

See Also

point.in.polygon, CutoutPolygon, mba.points

Examples

```
x <- c(7.8, 5, 2.2, 3.7, NA, -1.6, -7.5)
y <- c(-2.3, -4.7, -2.2, -2.3, -3.4, -1.6, -7.5)
z <- c(-0.9, -1.2, -2.4, -2.4, -0.4, 0.1, 2)
t <- c("4/23/2009 10:43", "4/24/2009 11:20", "4/24/2009 12:08", "4/24/2009 12:50",
       "4/24/2009 13:51", "4/24/2009 14:24", "4/24/2009 14:44")
t <- as.POSIXct(t, format = "%m/%d/%Y %H:%M")

data.raw <- as.data.frame(list(x = x, y = y, z = z, t = t))
data.pts <- ProcessData(data.raw, type = "p")
data.grd <- ProcessData(data.pts, type = "g")

ply <- as(cbind(c(-4, 2, -6), c(-7, -3, -3)), "gpc.poly")
grid.res <- list(x = 0.2, y = 0.5)
data.grd <- ProcessData(data.pts, type = "g", ply = ply, grid.res = grid.res)
```

`ProgressBar` *Progress Bar*

Description

A progress bar that shows the status of long-running operations.

Usage

```
ProgressBar(win.title = "Progress Bar", label = "", maximum = 100,
            nsteps = NULL, min.nsteps = 10L, parent = NULL)

SetProgressBar(pb, value, label = NULL, step = NULL)
```

Arguments

`win.title`	character; a string to display as the title of the dialog box.
`label`	character; a string to display in the dialog box.
`maximum`	numeric; the maximum value for the progress bar. The minimum value is `0`.
`nsteps`	numeric; the total number of increments the progress bar will make.
`min.nsteps`	numeric; the minimum number of increments. If greater than `nsteps`, the dialog box is not opened.
`parent`	tkwin; the GUI parent window.
`pb`	ProgressBar; an object returned from `ProgressBar` (see 'Value').
`value`	numeric; the value for the progress bar, between `0` and `maximum`.
`step`	numeric; the number of progress bar increments. If equal to `nsteps`, the dialog box will close.

Value

For ProgressBar an object of class "ProgressBar" and mode list is returned. Components of the list object include:

GetValue	function; returns the value of the progress bar.
MoveProgressBar	function; moves progress bar, passes a numeric argument.
SetLabel	function; sets label in the dialog box, passes a character argument.
DestroyWindow	function; closes the dialog box.
GetWindowState	function; returns FALSE if the dialog box has been closed, otherwise TRUE.
nsteps	numeric; see 'Arguments'.

For SetProgressBar, the previous value of the progress bar. An error is returned if the progress has terminated prematurely.

Author(s)

J.C. Fisher

References

This code was derived from the tkProgressBar function.

Examples

```
## Not run:
maximum <- 10
label <- "Estimated time to completion is being calculated\u2026"
pb <- ProgressBar(label = label, maximum = maximum, nsteps = maximum)

for (i in seq_len(maximum)) {
  est.time <- system.time(Sys.sleep(1))["elapsed"] * (maximum - i)
  label <- paste("Estimated time to completion is", round(est.time), "secs")
  ans <- try(SetProgressBar(pb, value = i, label = label, step = i))
  if (inherits(ans, "try-error"))
    break
}

## End(Not run)
```

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.
win.title	character; a string to display as the title of the dialog box.
parent	tkwin; the GUI parent window.

Value

Returns a character vector with updated values of names.

Author(s)

J.C. Fisher

Examples

```
## Not run:
Rename(names = c("Name1", "Name2", "Name3"), cur.name = "Name2")

## End(Not run)
```

Search *Search*

Description

A GUI for establishing find and replace arguments in a data table.

Usage

```
Search(is.replace = FALSE, defaults = NULL, parent = NULL)
```

Arguments

is.replace	logical; if TRUE, the replace component is included.
defaults	list; see 'Value'.
parent	tkwin; the GUI parent window.

Value

A `list` with the following components:

`find.what`	character; the string to search for.
`replace.with`	character; the replace string.
`is.match.word`	logical; should matches be restricted to whole words only?
`is.match.case`	logical; is the search case sensitive?
`is.reg.exps`	logical; if `TRUE`, the search is made using regular expression (that is, a pattern that describes a set of strings).
`is.search.col`	logical; is the search limited to a single column?
`is.perl`	logical; should Perl style regular expressions be used?
`is.replace.first`	logical; replace only the first instance?
`is.search.sel`	logical; is the search limited to selected cells?

Author(s)

J.C. Fisher

Examples

```
## Not run: ans <- Search()
```

`SetAxesLimits` *Set Axes Limits*

Description

A GUI for specifying axes limits.

Usage

```
SetAxesLimits(lim = NULL, parent = NULL)
```

Arguments

`lim`	list; contains the current plotting limits, see 'Value'.
`parent`	tkwin; the GUI parent window.

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.
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.
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)

J.C. Fisher

Examples

```
## Not run: SetAxesLimits()
```

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 GUI parent window.

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.
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 interpolated surfaces; if FALSE, the default, the filled.contour function is used.
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 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.
rm.pnt.line	logical; if TRUE the line boundary on point symbols is not drawn; its default is FALSE.

Note

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

Author(s)

J.C. Fisher

Examples

```
## Not run: SetConfiguration()
```

SetInterpolation *Set Interpolation Preferences*

Description

A GUI for specifying the interpolation algorithms input parameters.

Usage

```
SetInterpolation(parent = NULL)
```

Arguments

parent	tkwin; the GUI parent window.

Value

Queries and sets the following components in Data:

grid.res	list; numeric components x and y giving the grid spacing along the x- and y-axis of the interpolated surface, respectively.
grid.mba	list; integer components m, n, and h giving the initial size of the spline space in the hierarchical construction along the x- and y-axis, and the 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)

J.C. Fisher

See Also

mba.points

Examples

```
## Not run: SetInterpolation()
```

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; a vector of polygon names.
poly.data	character; the name of the polygon that defines the data limits boundary.
poly.crop	character; the name of the polygon that defines the crop region for interpolated data.
parent	tkwin; the GUI parent window.

Value

Returns a list with components `poly.data` and `poly.crop`.

Author(s)

J.C. Fisher

Examples

```
## Not run: SetPolygonLimits(c("Polygon1", "Polygon2", "Polygon3"))
```

`SetSortOrder` *Set Sort Order*

Description

A GUI for specifying the variable used to sort the data set.

Usage

```
SetSortOrder(col.ids, sort.on = NULL, parent = NULL)
```

Arguments

`col.ids`	character; a vector of variable names.
`sort.on`	integer; the index of the variable used to sort the data set.
`parent`	tkwin; the GUI parent window.

Value

Returns an integer object that specifies the index of the variable used to sort the data set. Attributes for this object include: `decreasing`, a logical value indicating if the sort order is increasing or decreasing; and `na.last`, a logical value for controlling the treatment of NAs during sorting. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed.

Author(s)

J.C. Fisher

See Also

`order`

Examples

```
## Not run:
col.ids <- c("Variable1", "Variable2", "Variable3")
sort.on <- 2
attr(sort.on, "decreasing") <- TRUE
attr(sort.on, "na.last") <- FALSE
SetSortOrder(col.ids, sort.on)

## End(Not run)
```

Index