

# Package ‘RSurvey’

January 2, 2012

**Version** 0.6-8

**Date** 2011-12-27

**Title** Analysis of Spatially Distributed Data

**Author** Jason C. Fisher

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

**Depends** R (>= 2.12.0), tcltk, sp, gplotlib, rgl, MBA, tripack, colorspace

**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** <https://github.com/stormplot/RSurvey>

**BugReports** <https://github.com/stormplot/RSurvey/issues>

**ByteCompile** yes

**Repository** CRAN

**Date/Publication** 2011-12-28 13:50:21

## R topics documented:

RSurvey-package . . . . .	2
AddAxis . . . . .	3
Autocrop . . . . .	4
AutocropPolygon . . . . .	5
CheckEntry . . . . .	6
ChooseColor . . . . .	7

ChoosePalette . . . . .	8
ChoosePch . . . . .	9
CutoutPolygon . . . . .	9
Data . . . . .	10
EditFunction . . . . .	12
EditLimits . . . . .	13
EvalFunction . . . . .	14
ExportData . . . . .	15
Format . . . . .	15
FormatDateTime . . . . .	16
GetBitmapImage . . . . .	17
GetFile . . . . .	18
ImportData . . . . .	19
LoadPackages . . . . .	20
ManageData . . . . .	20
ManagePolygons . . . . .	22
OpenRSurvey . . . . .	23
Plot2d . . . . .	24
Plot3d . . . . .	26
PlotTimeSeries . . . . .	27
ProcessData . . . . .	28
project . . . . .	30
ReadData . . . . .	31
Rename . . . . .	32
SetConfiguration . . . . .	33
SetPolygonLimits . . . . .	34
SetPreferences . . . . .	35
SummarizeData . . . . .	36
ViewData . . . . .	37
WriteFile . . . . .	38

<b>Index</b>	<b>40</b>
--------------	-----------

---

RSurvey-package	<i>Analysis of Spatially Distributed Data</i>
-----------------	---

---

## 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. 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](#).

## Examples

```
library(RSurvey)
OpenRSurvey()
```

---

AddAxis	<i>Add an Axis to a Plot</i>
---------	------------------------------

---

## Description

Adds an axis to the current plot.

## Usage

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

## Arguments

<code>side</code>	integer; a vector of values specifying the plot sides for the axis to be drawn.
<code>lim</code>	numeric or POSIXt; the axis limits ( $x_1$ , $x_2$ ) of the plot.
<code>ticks.inside</code>	logical; if TRUE tickmarks are placed inside the plot region; its default is FALSE.
<code>minor.ticks</code>	logical; if TRUE minor tickmarks are added to the plot; its default is FALSE.
<code>...</code>	other graphical parameters may also be passed as arguments to this function, see <a href="#">axis</a> .

## 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](#), [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 [gpc.poly-class](#).

### Author(s)

J.C. Fisher

### 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 of the [Autocrop](#) function.

## Usage

```
AutocropPolygon(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 <a href="#">Plot2d</a> .

## 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

## See Also

[tri.mesh](#)

## Examples

```
data(tritest)
AutocropPolygon(as.data.frame(tritest), asp = 1)
```

---

CheckEntry

*Content Control within Entry Widget*

---

## Description

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

## Usage

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

## Arguments

<code>ent.typ</code>	character; the entry type.
<code>ent.str</code>	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 data format.

## Author(s)

J.C. Fisher

## 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"
```

---

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

```
ChooseColor(col = "#669933")
```

---

`ChoosePalette`*Set Color Palette*

---

**Description**

A GUI for viewing or manipulating color palettes.

**Usage**

```
ChoosePalette(pal = terrain_hcl, n = 7L, parent = NULL)
```

**Arguments**

<code>pal</code>	function; the initial palette, see ‘Value’ below.
<code>n</code>	integer; the number of colors in the palette.
<code>parent</code>	tkwin; the GUI parent window.

**Details**

Computes palettes based on the HCL ([polarLUV](#)) color space.

**Value**

Returns a function that takes an integer argument and returns that number of colors interpolating the given sequence.

**Author(s)**

J.C. Fisher

**References**

Zeileis A., Hornik K., and Murrell P., 2009, *Escaping RGBland: Selecting Colors for Statistical Graphics*, Computational Statistics & Data Analysis, 53, p. 3259-3270.

**See Also**

[rainbow\\_hcl](#)

**Examples**

```
pal <- ChoosePalette(terrain_hcl)
filled.contour(volcano, color.palette = pal, asp = 1)
```

---

ChoosePch	<i>Set Graphic Symbol</i>
-----------	---------------------------

---

**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**

```
ChoosePch(pch = "+")
```

---

CutoutPolygon	<i>Determine Grid Points within Polygon</i>
---------------	---

---

**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**

```
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 parameters used in **RSurvey**.

**Usage**

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

**Arguments**

<code>option</code>	character; the parameter name, see 'Parameters'.
<code>value</code>	a parameter value specified for <code>option</code> .
<code>clear.proj</code>	logical; if TRUE basic GUI preferences will be saved and all other data removed; its default is FALSE.
<code>clear.data</code>	logical; if TRUE only data sets will be removed, its default is FALSE.
<code>replace.all</code>	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 [ReadData](#). 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)
```

---

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	list; see <a href="#">ManageData</a> .
index	integer; an element index number in cols.
parent	tkwin; the GUI parent window.

## Details

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

## Value

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

## Author(s)

J.C. Fisher

## See Also

[EvalFunction](#), [parse](#)

## 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)
```

**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; a string to display as the title of the dialog box.
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.
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)**

J.C. Fisher

**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 and 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)

J.C. Fisher

## See Also

[parse](#), [eval](#), [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)
EvalFunction("DATA[[\"X\"]] + DATA[[\"Y\"]]", cols)
EvalFunction("rnorm(12)", cols)
```

---

ExportData	<i>Set Parameters for Data Export</i>
------------	---------------------------------------

---

**Description**

A GUI for exporting data to text files or shapefiles.

**Usage**

```
ExportData(col.ids, file.type = "text", parent = NULL)
```

**Arguments**

col.ids	character; a vector of strings identifying the variables to include in the exported data table.
file.type	character; the output file type: either "text" for <i>Text Files</i> or "shape" for <i>ESRI Shapefiles</i> .
parent	tkwin; the GUI parent window.

**Author(s)**

J.C. Fisher

**See Also**

[WriteFile](#)

**Examples**

```
ExportData(col.ids = c("V1", "V2", "V3"), file.type = "text")
```

---

Format	<i>Build C-Style String Formats</i>
--------	-------------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

sample	logical, integer, numeric, character, or factor; a sample value.
fmt	character; the conversion specification format, see <a href="#">sprintf</a> .
parent	tkwin; the GUI parent window.

**Value**

Returns a character string.

**Author(s)**

J.C. Fisher

**See Also**

[sprintf](#), [format](#)

**Examples**

```
Format(sample = pi, fmt = "%3.8f")
Format(sample = 3L)
Format(sample = TRUE)
Format(sample = "string")
```

---

FormatDateTime

*Build Date-Time String Formats*

---

**Description**

A GUI for converting between character representations and objects of class “POSIXt”, calendar dates and times.

**Usage**

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

**Arguments**

sample	POSIXct or POSIXlt; a calendar date and time object.
fmt	character; the conversion specification format for date-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**

```
FormatDateTime(fmt = "%d/%m/%Y")
```

---

GetBitmapImage	<i>Icon Bitmap Image</i>
----------------	--------------------------

---

**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*, and *print*. 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**

```
image.obj <- GetBitmapImage("left")
tt <- tkoplevel(padx = 50, pady = 50)
button <- ttkbutton(tt, width = 2, image = image.obj)
tkpack(button)
```

---

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, defaulttextextension = 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 <a href="#">connection</a> .
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.
defaulttextextension	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 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)**

J.C. Fisher

**Examples**

```
GetFile()
```

---

ImportData

*Import Data*

---

**Description**

A GUI for reading table formatted data.

**Usage**

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

**Arguments**

parent                    tkwin; the GUI parent window.

**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, a complete URL (e.g. `http://`, `ftp://` or `file://`), or windows 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 `Data` components: `headers`, `skip`, `sep`, `nrows`, `na.strings`, `quote`, `comment.char`, see `ReadData`; and `data.source`, where

`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)**

J.C. Fisher

**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)**

J.C. Fisher

**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 GUI parent window.

**Details**

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

**Value**

Queries and sets the cols component of [Data](#), 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.
format	character; the conversion specification format (optional).
id	character; a unique identifier that is created from a string concatenation of name and unit.
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 <a href="#">ImportData</a> . Only required for variables directly linked to data columns in data.raw.
class	character; the data class of the vector object.
summary	list; a summary of the variables descriptive statistics (see <a href="#">SummarizeData</a> ).
comments	character; user comments.

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.
--------------------	--

The vars component of [Data](#) is updated to reflect the removal and (or) reordering of variables.

**Author(s)**

J.C. Fisher

**Examples**

```
data(project)
ManageData(project$cols, project$vars)
```

---

 ManagePolygons

*Manage Polygons*


---

### Description

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

### Usage

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

### Arguments

ply	list; its components are objects of <code>gpc.poly-class</code> .
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 **gpplib** package for details.

### Value

Queries and sets the `ply`, `poly.data`, and `poly.crop` components of `Data`, see [SetPolygonLimits](#).

### Author(s)

J.C. Fisher

**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 through 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`, see [ManageData](#).

**Author(s)**

J.C. Fisher

**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,
       rm.pnt.line = FALSE)
```

**Arguments**

x	numeric; a vector of x coordinates for the plot. If x is a data frame, 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	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.

<code>nlevels</code>	integer; number of contour levels desired.
<code>rkey</code>	logical; if TRUE the legend key is reversed with z values descending from top to bottom; its default is FALSE.
<code>color.palette</code>	function; a color <a href="#">palette</a> to be used to assign colors in the plot.
<code>vuni</code>	logical; if TRUE all arrow lengths are set equal; its default is FALSE.
<code>vmax</code>	numeric; the maximum length of arrows in inches.
<code>vxby, vyby</code>	integer; increment for the sequence of arrows in the x and y direction.
<code>axis.side</code>	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.
<code>minor.ticks</code>	logical; if TRUE minor tickmarks are added to the plot; its default is FALSE.
<code>ticks.inside</code>	logical; if TRUE tickmarks are placed inside the plot region; its default is FALSE.
<code>add.contour.lines</code>	logical; if TRUE and <code>type</code> is either "l" or "g" than contour lines are drawn; its default is FALSE.
<code>rm.pnt.line</code>	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(project)

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

d <- project$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. 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 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 <a href="#">palette</a> 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

## See Also

[surface3d](#), [points3d](#)

## Examples

```
data(project)
d <- project$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,
               rm.pnt.line = 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.

<code>width</code>	numeric; the width of the plotting window canvas in inches.
<code>cex.pts</code>	numeric; the amount by which point symbols should be magnified relative to the default.
<code>pointsize</code>	integer; the point size of plotted text.
<code>fmt</code>	character; date-time format for x axis tickmark labels, see <a href="#">strptime</a> .
<code>axis.side</code>	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.
<code>minor.ticks</code>	logical; if TRUE minor tickmarks are added to the plot; its default is FALSE.
<code>ticks.inside</code>	logical; if TRUE tickmarks are placed inside the plot region; its default is FALSE.
<code>rm.pnt.line</code>	logical; if TRUE the line boundary on point symbols is not drawn; its default is FALSE.

### Details

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

### Author(s)

J.C. Fisher

### See Also

[points](#), [lines](#), [AddAxis](#)

### Examples

```
data(project)
d <- project$data.pts
PlotTimeSeries(x = d$t, y = d$z)
PlotTimeSeries(x = d$t, y = d$z, tgap = 3000, rm.pnt.line=TRUE)
```

---

ProcessData

*Process Data*

---

### Description

This function performs data processing on the state variables.

### Usage

```
ProcessData(d, type = "p", lim = 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.
lim	list; contains numeric vector components giving the minimum and maximum values for each variable in d.
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 <a href="#">SetPreferences</a> .
grid.mba	list; integer components giving the input parameters for the multilevel B-splines algorithm, see <a href="#">SetPreferences</a> .

**Details**

Any row in the d data table with NA values for either x, y, or t is removed. The spatial and temporal domains are constrained using data limits for x, y, z, and t. Additionally, 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).
t	POSIXct; a vector of time stamps (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")

lim <- list(x = c(NA, 5), t = c(as.POSIXct("4/24/2009 12",
                                         format = "%m/%d/%Y %H"), NA))
data.pts <- ProcessData(data.raw, type = "p", lim = lim)

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)
```

---

 project

*Example Project Data Set*


---

**Description**

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

**Usage**

```
project
```

**Format**

A list containing typical components in a project; these components are described throughout this documentation and include: data sets, graphical parameters, and options to control the way in which **RSurvey** processes data.

**Examples**

```
data(project)
# To load a project data set into RSurvey
Data(replace.all = project)
```

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 <a href="#">connection</a> object.
headers	logical; a vector of length 3 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 interpreted as <a href="#">NA</a> 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 imported data table requires at least two numeric variables.

**Details**

This function is the primary method for importing table formatted data from a text file. The `headers` argument, a logical vector of length 3, indicates whether the file contains the names, measurement units, and conversion specification formats of the variables as its initial lines. For example, `headers = c(TRUE, FALSE, TRUE)` indicates that the first and second lines contain the names and formats of variables, respectively; with measurement units excluded. If `headers = c(FALSE, FALSE, FALSE)`, the default, no header information is contained within the file.

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".

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

Returns a list with the following components:

dat	data.frame; a data table with headers and comments removed.
cols	list; length equal to the current number of data variables. Each component in cols is linked to a specific variable, see <a href="#">ManageData</a> .
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)

J.C. Fisher

### 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.
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**

```
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 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, <code>cex.pts = 0.5</code> 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.

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 <code>image</code> function is used to plot interpolated surfaces; if FALSE, the default, the <code>filled.contour</code> 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**

```
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, a list of polygon components, see <a href="#">ManagePolygons</a> .
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

**See Also**

[AutocropPolygon](#), [tri.mesh](#)

**Examples**

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

---

SetPreferences	<i>Set Data Preferences</i>
----------------	-----------------------------

---

**Description**

A GUI for specifying the interpolation algorithms input parameters.

**Usage**

```
SetPreferences(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**

```
SetPreferences()
```

---

SummarizeData

*Summarize Object*

---

**Description**

A summary of the descriptive statistics of an array object.

**Usage**

```
SummarizeData(obj, fmt = NULL)
```

**Arguments**

`obj` an array object for which the summary is desired.  
`fmt` character; the conversion specification format, see [sprintf](#).

**Value**

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

Count	integer; array length.
NAs	integer; number of <code>NA</code> values.
Class	character; the objects <code>class</code> attribute.
Min., Max.	numeric; extreme values with <code>NA</code> values ignored.
1st Qu., Median, 3rd Qu.	numeric; estimates of the underlying distribution quantiles with <code>NA</code> values ignored.
Mean	numeric; arithmetic mean with <code>NA</code> values ignored.
St.Dev.	numeric; standard deviation with <code>NA</code> values ignored.
Sum	numeric; sum with <code>NA</code> values ignored.

Hist	histogram; an object of class histogram, see <code>hist</code> 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)**

J.C. Fisher

**See Also**

[quantile](#), [hist](#)

**Examples**

```
summary(attenu$dist, digits = 4)
SummarizeData(attenu$dist, fmt = "%.2f")
SummarizeData(as.POSIXct(Sys.time() + 1:10), fmt = "%a %H:%M:%S")
```

---

ViewData

*View Data*


---

**Description**

A GUI for viewing table formatted data.

**Usage**

```
ViewData(d, column.names = NULL, column.units = NULL,
         column.formats = NULL, parent = NULL)
```

**Arguments**

<code>d</code>	data.frame; the data used to populate the table.
<code>column.names</code>	character; a vector giving the column names for the data table (optional).
<code>column.units</code>	character; a vector giving the measurement units for each column of the data table (optional).
<code>column.formats</code>	character; the conversion specification format (optional).
<code>parent</code>	tkwin; the GUI parent window.

**Details**

Column titles are a concatenation of variables `column.names` and `column.units`. Row titles are taken from the row names attribute of `d`. Pattern searches are performed using [grep](#) with GUI options available for setting its fixed and perl arguments.

**Note**

Requires the Tcl package [Tktable](#).

**Author(s)**

J.C. Fisher

**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)
column.names <- c("Integers", "Letters", "DateTime", "Numeric")
column.units <- c("units", NA, NA, NA)
column.formats <- c("%d", "%s", "%m/%d/%Y %H:%M", NA)
ViewData(d, column.names, column.units, column.formats)

row.names(d) <- 1:n + n
ViewData(d, column.names, column.formats = column.formats)
```

---

WriteFile

*Write Data File*

---

**Description**

Writes data to a file.

**Usage**

```
WriteFile(file.type = "text", file.name = NULL, col.ids = NULL,
          headers = c(FALSE, FALSE, FALSE), sep = "\t",
          is.processed = TRUE, is.compressed = FALSE,
          encoding = getOption("encoding"))
```

**Arguments**

file.type	character; the output file type: either “text” for <i>Text Files</i> , “shape” for <i>ESRI Shapefiles</i> , or “grid” for <i>Interpolated Grid Text Files</i> .
file.name	character; the name of the file which the data are to be written to.
col.ids	character; a vector of strings identifying the variables to include in the exported data table.
headers	logical; a vector of length 3 that indicates whether the data table contains header lines.
sep	character; the field separator string.
is.processed	logical; if TRUE, only include record numbers in the processed data table, see <a href="#">ProcessData</a> ; its default is TRUE.
is.compressed	logical; if TRUE, the file will be compressed by <a href="#">gzip</a> ; its default is FALSE.
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.

**Author(s)**

J.C. Fisher

**See Also**

[write.table](#), [writeOGR](#)

**Examples**

```
data(project)
Data(replace.all = project)
WriteFile(file.type = "grid")
```

# Index

- \*Topic **aplot**
    - AddAxis, 3
  - \*Topic **datasets**
    - project, 30
  - \*Topic **file**
    - GetFile, 18
    - ReadData, 31
    - WriteFile, 38
  - \*Topic **hplot**
    - Plot2d, 24
    - Plot3d, 26
    - PlotTimeSeries, 27
  - \*Topic **manip**
    - CheckEntry, 6
    - CutoutPolygon, 9
    - ProcessData, 28
  - \*Topic **misc**
    - AutocropPolygon, 5
    - ChooseColor, 7
    - ChoosePalette, 8
    - ChoosePch, 9
    - EditFunction, 12
    - EditLimits, 13
    - ExportData, 15
    - Format, 15
    - FormatDateTime, 16
    - GetBitmapImage, 17
    - ImportData, 19
    - LoadPackages, 20
    - ManageData, 20
    - ManagePolygons, 22
    - OpenRSurvey, 23
    - Rename, 32
    - SetConfiguration, 33
    - SetPolygonLimits, 34
    - SetPreferences, 35
    - SummarizeData, 36
    - ViewData, 37
  - \*Topic **package**
    - RSurvey-package, 2
  - \*Topic **symbolmath**
    - Autocrop, 4
  - \*Topic **sysdata**
    - Data, 10
  - \*Topic **utilities**
    - EvalFunction, 14
- AddAxis, 3, 25, 28
- arrows, 25
- Autocrop, 4, 5
- AutocropPolygon, 4, 5, 35
- axis, 3
- axis.POSIXct, 3
- CheckEntry, 6
- ChooseColor, 7
- ChoosePalette, 8
- ChoosePch, 9
- col2rgb, 7
- connection, 18, 31
- connections, 19
- CutoutPolygon, 9, 29
- Data, 10, 14, 19, 21–23, 33, 35
- EditFunction, 12, 21
- EditLimits, 13
- eval, 14
- EvalFunction, 12, 14
- ExportData, 15
- filled.contour, 25, 34
- Format, 15
- format, 16, 17
- FormatDateTime, 16
- GetBitmapImage, 17
- GetFile, 18
- gpc.poly-class, 4, 5, 22
- grep, 37

hist, 37

image, 25, 34

ImportData, 19, 21

install.packages, 20

intersect, 23

is.infinite, 14

is.nan, 14

lines, 28

LoadPackages, 20

ManageData, 12, 14, 20, 23, 32

ManagePolygons, 22, 35

mba.points, 29, 36

NA, 29, 31, 36

OpenRSurvey, 23

palette, 25, 26

parse, 12, 14

Plot2d, 5, 24

Plot3d, 26

PlotTimeSeries, 27

point.in.polygon, 10, 29

points, 9, 28

points3d, 27

polarLUV, 8

polyfile, 23

pretty, 3

ProcessData, 11, 28, 39

project, 30

quantile, 37

rainbow\_hcl, 8

read.table, 19, 32

ReadData, 11, 12, 19, 31

Rename, 32

require, 20

row.names, 38

RSurvey-package, 2

seq, 3

SetConfiguration, 33

setdiff, 23

SetPolygonLimits, 22, 34

SetPreferences, 29, 35

sprintf, 16, 32, 36

strftime, 32

strptime, 17, 28

SummarizeData, 21, 36

surface3d, 27

tclArray, 38

tclObj, 17

tkentry, 6

tkimage.create, 17

union, 23

ViewData, 21, 37

write.table, 39

WriteFile, 15, 38