

# Package ‘RSurvey’

April 17, 2009

**Type** Package

**Title** Analysis of Spatially Distributed Data

**Version** 0.4.5

**Date** 2009-04-03

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

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

**Depends** R (>= 2.8.0), tcltk, rgl, MBA, gpplib, sp, mgcv, tripack

**Description** This package is a processing program for spatially distributed data. The program is capable of error corrections and data visualization. A graphical user interface is provided.

**License** GPL (>= 2)

**URL** <https://eng.ucmerced.edu/people/jfisher/software/rsurvey>

**Repository** CRAN

**Date/Publication** 2009-04-04 20:38:50

## R topics documented:

RSurvey-package . . . . .	2
autocrop . . . . .	3
axesLimits . . . . .	4
buildPackage . . . . .	5
confluence . . . . .	5
geoProj . . . . .	6
getFile . . . . .	6
keyEvent . . . . .	7
loadPackages . . . . .	8
mergeData . . . . .	9
minorTics . . . . .	10
plotPoints . . . . .	11

plotSurface2d . . . . .	11
plotSurface3d . . . . .	12
plotTime . . . . .	13
plotTransect . . . . .	13
polyAutocrop . . . . .	14
polyConstruct . . . . .	15
polyCutout . . . . .	16
readFile . . . . .	17
restoreSession . . . . .	19
srvy . . . . .	20
srvy.config . . . . .	21
srvy.dat . . . . .	21
srvy.export . . . . .	23
srvy.pref . . . . .	24
srvy.process . . . . .	25
srvy.vars . . . . .	26
tran . . . . .	27
tran.dat . . . . .	27
tran.edit . . . . .	28
tran.import-export . . . . .	29
tran.profile . . . . .	31
tran.spatial . . . . .	32
tran.vector . . . . .	33
tran.velocity . . . . .	34

<b>Index</b>	<b>35</b>
--------------	-----------

---

RSurvey-package      *Analysis of Spatially Distributed Data*

---

## Description

This package is a processing program for spatially distributed data. The application is capable of error corrections and data visualization. A graphical user interface (GUI) is provided.

## Details

Package: RSurvey  
 Type: Package  
 Version: 0.4.5  
 Date: 2009-04-03  
 License: GPL version 2 or newer.

**Note**

The **RSurvey** GUI requires that **R** operate as an SDI application with multiple top-level windows for the console, graphics and pager. Preservation of unit consistency within this package is tasked to the user. Parameter dimensions are included in the GUI entry labels to assist with unit consistency.

**Author(s)**

Jason C. Fisher

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

**Examples**

```
library(RSurvey)
```

---

autocrop

*Auto Crop Spatial Domain*

---

**Description**

Eliminate mesh cells with arc lengths greater than a user defined maximum.

**Usage**

```
autocrop(mesh, maxLength, maxItr)
```

**Arguments**

mesh	an object of class <code>tri</code> .
maxLength	the maximum arc length for a cell within the mesh.
maxItr	the maximum number of iterations.

**Value**

A polygon of class `gpc.poly`.

**Author(s)**

Fisher, J. C.

**See Also**

[polyAutocrop](#)

**Examples**

```

data(tritest)
mesh <- tri.mesh(tritest$x, tritest$y)
plot(mesh); axis(1); axis(2); box()
ply <- autocrop(mesh, maxLength = 0.5, maxItr = 100)
plot(ply, add = TRUE, poly.args = list(col = 2))

```

---

axesLimits

*Set Limits on Raster Data*


---

**Description**

A GUI for setting plotting limits associated with raster data.

**Usage**

```
axesLimits(tt2 = NULL, old = list())
```

**Arguments**

tt2	a Tk toplevel widget.
old	the current plotting limits, see Value section.

**Value**

A list containing 6 components:

z.min	the minimum z-axis value, default NA.
z.min.auto	the default value for the minimum z-axis, default TRUE.
z.max	the maximum z-axis value, default NA.
z.max.auto	the default value for the maximum z-axis, default TRUE.
z.lev	the number of contour levels, default NA.
z.lev.auto	the default value for the number of contour levels, default TRUE.

**Author(s)**

Fisher, J. C.

**Examples**

```
axesLimits()
```

---

buildPackage	<i>Package Builder</i>
--------------	------------------------

---

**Description**

This function is for package developers only. Package builder has been tested on a Windows operating system (OS) within the package source directory.

**Usage**

```
buildPackage
```

**Value**

The shell commands necessary for building **RSurvey** are printed to the R console. The user is tasked with copy/pasting the commands into a Windows command shell.

**Author(s)**

Fisher, J. C.

**Examples**

```
## Not run: buildPackage()
```

---

confluence	<i>Sample Project</i>
------------	-----------------------

---

**Description**

A sample **RSurvey** project containing list objects `srvy.dat` and `tran.dat`.

**Usage**

```
data(confluence)
```

**Author(s)**

Fisher, J. C., S. Villamizar, H. Pai

**Source**

The survey data was collected within the San Joaquin River Basin, CA, USA, at the confluence of the Merced and San Joaquin river in July, 2007.

**Examples**

```
data(confluence)
summary(srvy.dat())
summary(tran.dat())
```

---

geoProj                      *Geographical Projection*

---

**Description**

A description of the geographical projection for the current project.

**Usage**

```
geoProj(tt1 = NULL)
```

**Arguments**

tt1                      a Tk toplevel widget.

**Value**

A character string, the projection component of `srvy.dat`.

**Author(s)**

Fisher, J. C.

**Examples**

```
geoProj()
```

---

getFile                      *Select a File to Open or Save.*

---

**Description**

A pop up a dialog box for the user to select a file to open or save.

**Usage**

```
getFile(cmd = "Open", exts = NULL, directory = NULL, file = NULL, titleStr = cmd)
```

**Arguments**

<code>cmd</code>	determines if an "Open" or "Save As" file management pop up dialog box is implemented.
<code>exts</code>	an array of default file extensions.
<code>directory</code>	specifies that the files in <code>directory</code> should be displayed when the dialog pops up.
<code>file</code>	either a character string naming a file or a connection.
<code>titleStr</code>	a character string to display as the title of the dialog box.

**Value**

A list containing 4 components:

<code>dir</code>	the directory containing the file.
<code>path</code>	the file path
<code>name</code>	the file name
<code>ext</code>	the file extension

**Author(s)**

Fisher, J. C.

**Examples**

```
getFile()
```

---

keyEvent

*Content Control within Tk Entry Widgets*

---

**Description**

Controls the character string content within a Tk entry widget.

**Usage**

```
keyEvent(ent.typ, ent.str = "", rm.data = FALSE)
```

**Arguments**

<code>ent.typ</code>	the entry type
<code>ent.str</code>	the entry value
<code>rm.data</code>	a logical; if TRUE the existing processed data is removed.

**Details**

The entry types include: *real*, *second*, *integer*, *hour*, *minute*, and *date*. If `rm.data` is `TRUE` then both the `data.mod` and `data.tin` components of `srvy.dat` are removed.

**Value**

A character string with strict adherence to the entry type.

**Author(s)**

Fisher, J. C.

**See Also**

[tkentry](#)

**Examples**

```
keyEvent("date", "07/12/1971")  
## [1] "07/12/1971"
```

```
keyEvent("date", "7/2/19uq")  
## [1] "7/2/19"
```

```
keyEvent("hour", "13")  
## [1] "13"
```

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

---

loadPackages

*Load Required Packages*

---

**Description**

This function loads `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()
```

**Author(s)**

Fisher, J. C.

**Examples**

```
loadPackages()
```

---

`mergeData`*Merge Time-stamped Data*

---

**Description**

This function merges temporal data with the current time-stamped survey data.

**Usage**

```
mergeData(file = NULL)
```

**Arguments**

`file` either a character string naming a file or a connection.

**Format**

The tab delineated temporal file, '\*.txt', contains the following variables:

[, 1]	character	date/time
[, 2 : ]	numeric	state variables

The first [1, ] and second [2, ] rows are reserved for column descriptors and units, respectively. Units associated with date/time values are based on format character strings described in [strptime](#).

**Value**

The `cols` and `data.raw` components of `srvy.dat` are replaced by the merged data set.

**Note**

This function is not currently accessible through a GUI.

**Author(s)**

Fisher, J. C.

**See Also**

[approx](#)

**Examples**

```
f <- system.file("RSurvey-ex/river.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
d <- readfile(con)
srvy.dat("cols", d$cols)
```

```
srvy.dat("vars", d$vars)
srvy.dat("data.raw", d$dat)

f <- system.file("RSurvey-ex/river-temperature.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
mergeData(con)
srvy.dat("cols")
srvy.dat("data.raw")
```

---

minorTics

*Place Minor Tick Marks on Plot Axis*

---

### Description

Adds minor axis tic marks to the active plot.

### Usage

```
minorTics(side, loc.tics = NULL)
```

### Arguments

`side` an array of integers specifying the plot sides for the axis to be drawn.  
`loc.tics` the location of the major tic marks.

### Details

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

### Author(s)

Fisher, J. C. and J. Lewis

### See Also

[axis](#)

### Examples

```
plot(1:10)
minorTics(1)
minorTics(2, loc.tics=4:8)
minorTics(3, c(4, 6, 8))
```

---

`plotPoints`*Plot Spatial Locations and Corresponding State Variable*

---

**Description**

Draw a scatter plot of spatial coordinates (x,y) and their associated state variable (z) values. A key showing how the colors map to state variable values is shown to the right of the plot.

**Usage**

```
plotPoints()
```

**Author(s)**

Fisher, J. C.

**See Also**

[plot](#)

**Examples**

```
data(confluence)
plotPoints()
```

---

`plotSurface2d`*Plot 2D Interpolated Surface*

---

**Description**

A filled contour plot showing the distributed spatial data is drawn. A key showing how the colors map to state variable values is shown to the right of the plot.

**Usage**

```
plotSurface2d(const.poly = FALSE)
```

**Arguments**

`const.poly` a logical; if TRUE the user is required to interactively construct the spatial domain within the active plot.

**Author(s)**

Fisher, J. C.

**See Also**

`mba.surf`, `filled.contour`

**Examples**

```
data(confluence)
plotSurface2d()
```

---

`plotSurface3d`

*Plot 3D Interpolated Surface using OpenGL*

---

**Description**

A three-dimensional (3D) surface plot of the processed survey data is drawn.

**Usage**

```
plotSurface3d()
```

**Details**

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

**Author(s)**

Fisher, J. C.

**See Also**

`surface3d`

**Examples**

```
data(confluence)
plotSurface3d()

rgl.quit()
```

---

plotTime

*Plot the Raw and Corrected State Variable vs. Time*


---

**Description**

A scatter plot is drawn of raw and corrected state variable measurements versus time.

**Usage**

```
plotTime()
```

**Details**

This plotting routine is designed as a tool for error correction.

**Author(s)**

Fisher, J. C.

**See Also**

[plot](#)

**Examples**

```
## None at this time
```

---

plotTransect

*Plot Profile and Raster Scan Data*


---

**Description**

The profile along a transect and/or an interpolated raster distribution within the cross-sectional area of the transect is drawn.

**Usage**

```
plotTransect(id, rasterField = NULL, velPlot = FALSE, flow = "magnitude")
```

**Arguments**

id	the name of the transect.
rasterField	the raster field of interest, see the type component in <a href="#">tran.dat</a> .
velPlot	a logical; if TRUE velocity raster data is plotted.
flow	the velocity component types to use where "magnitude" is the principle direction of flow and "longTran" is the longitudinal and transverse directions.

**Details**

If the `rasterField` argument is missing a profile is plotted along the transect. The horizontal axis is representative of the distance between transect vertices and the vertical axis the distance from the vertex origin, `v.origin`, positive upward. The profile along a transect is by default based on the interpolated surface, `data.tin`. If the `data.tin` component of `srvy.dat` is NULL the function will attempt to access imported profile data in `prof`.

**Author(s)**

Fisher, J. C.

**See Also**

`mba.points`, `filled.contour`, `tran.profile`

**Examples**

```
data(confluence)
plotTransect("T1")
plotTransect("T1", rasterField = "SpCond (uS/cm)")
plotTransect("T1", velPlot = TRUE, flow = "longTran")
```

---

polyAutocrop

*Auto Crop Parameters*

---

**Description**

Control parameters associated with the autocrop algorithm.

**Usage**

```
polyAutocrop(ttl = NULL)
```

**Arguments**

`ttl` a Tk toplevel widget.

**Details**

Requires access to the following components within `srvy.dat`: `data.raw`, `win.width`, `csi`, and `vars[2:3]`. The default maximum arc length for a cell is determined from the maximum length within the Delaunay triangulation mesh and an iteration limit for the algorithm. Entering arc lengths less than the default value will reduce the polygons size.

**Value**

A polygon of class `gpc.poly`.

**Author(s)**

Fisher, J. C.

**See Also**`autocrop`, `tri.mesh`, `get.pts`**Examples**

```
data(confluence)
polyAutocrop()
```

---

polyConstruct      *Polygon Construction*

---

**Description**

Read polygon information from a text file.

**Usage**

```
polyConstruct(file = NULL, dl = NULL)
```

**Arguments**

<code>file</code>	either a character string naming a file or a connection.
<code>dl</code>	the maximum distance between polygon vertices. If <code>NULL</code> , no vertices are added.

**Format**

The tab delineated text file, `*.txt`, adheres to the following 4 column format:

<code>[, 1]</code>	integer	an index for vertices within a polygon.
<code>[, 2]</code>	numeric	vertex location in the x-direction.
<code>[, 3]</code>	numeric	vertex location in the y-direction.
<code>[, 4]</code>	integer	a code, see Details.

**Details**

The `code` argument is used to identify polygons and vertex types. Letting `code = 0` for two sequential vertices will override `dl` and prevent additional vertices from being added between them. A `code < 2` indicates a vertex point is within the outer polygon. Inner polygons or holes are specified with a `code > 1`.

**Value**

A polygon of class `gpc.poly-class`.

**Author(s)**

Fisher, J. C.

**See Also**

`polyfile`

**Examples**

```
f <- system.file("RSurvey-ex/confluence-poly.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
ply <- polyConstruct(file = con)
pts <- get.pts(ply)
plot(ply)
for(i in 1:length(pts)) points(get.pts(ply)[[i]], col = "red")

con <- file(f, open = "r", encoding = "latin1")
ply <- polyConstruct(file = con, dl = 1)
pts <- get.pts(ply)
for(i in 1:length(pts)) points(get.pts(ply)[[i]], col = "green")
```

---

`polyCutout`

*Cutout Grid Points within Polygon*

---

**Description**

This function excludes data points outside a given polygon.

**Usage**

```
polyCutout(dat, ply)
```

**Arguments**

`dat` a list containing x, y, and z components, see Value section.  
`ply` a polygon of class `gpc.poly-class`.

**Details**

Returns NA values for z when points (x, y) are located outside the polygon.

**Value**

A list containing the following components:

x	grid point locations in the x-direction.
y	grid point locations in the y-direction.
z	a matrix of state variable values corresponding to the grid.

**Author(s)**

Fisher, J. C.

**See Also**

`point.in.polygon`, `filled.contour`

**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))

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

dat.new <- polyCutout(dat.old, ply)
filled.contour(dat.new, color.palette = terrain.colors)
```

---

readFile

*Data Input*

---

**Description**

Reads a file in table format.

**Usage**

```
readFile(file, fields = TRUE, units = TRUE)
```

**Arguments**

file	either a character string naming a file or a connection.
fields	a logical value indicating whether the file contains the names of the variables as its first line.
units	a logical value indicating whether the file contains the units of the variables as its second line.

**Format**

A tab delineated file, `*.txt`, containing the following variables:

<code>[, 1]</code>	character	date/time (optional)
<code>[, 2]</code>	numeric	x values
<code>[, 3]</code>	numeric	y values
<code>[, 4:]</code>	numeric	z values, state variable(s)

The first `[1, ]` and second `[2, ]` rows are reserved for column descriptors and units, respectively. Units associated with date/time values are based on format character strings described in [strptime](#), for example `"%d/%m/%Y %H:%M:%OS"`. You can embed comments in the data file by using the `"#"` character. Anything after a `"#"` on a line will be ignored.

**Value**

A list with the following components: `cols`, a character array of column names. `vars`, a character array of column names associated with the date/time `[1]`, x `[2]`, y `[3]`, and z `[4]` variables. `data.raw`, a data frame of raw survey data.

**Author(s)**

Fisher, J. C.

**See Also**

[read.table](#)

**Examples**

```
## A sample file including a header and two rows of data.

## Date / Time Easting Northing      Depth
## %d/%m/%Y %H:%M:%OS      m      m      m
## 06/01/2001 01:01:59.000      727972.52      4138308.65      3.12
## 06/01/2001 01:01:59.200      727972.51      4138308.68      3.12

f <- system.file("RSurvey-ex/lake.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
d <- readfile(con)
srvy.dat("cols", d$cols)
srvy.dat("vars", d$vars)
srvy.dat("data.raw", d$dat)
```

---

restoreSession      *Restore Session from Source Code Files*

---

### Description

This function restores local objects within the current R session.

### Usage

```
restoreSession(path, save.objs, fun.call)
```

### Arguments

<code>path</code>	a full path name; if missing a folder browser pop up dialog box is implemented.
<code>save.objs</code>	the name of local objects to preserve during the recompiling process.
<code>fun.call</code>	the name of the function to call after restoring the session (optional).

### Value

An updated R session compiled from '.R' source code files located in `path` and its recursive directories.

### Note

Restoring `srvy.dat` and `tran.dat` clears their components.

### Author(s)

Fisher, J. C.

### See Also

[source](#)

### Examples

```
## Not run: restoreSession()
```

## Description

This function activates the main graphical user interface (GUI) for **RSurvey**.

## Usage

```
srvy()
```

## Details

Most functions within **RSurvey** are accessible within this GUI. The text file format of survey data is described in [readFile](#). 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

The following components within `srvy.dat` are set: `sd`, `ed`, `sh`, `sm`, `em`, `ss`, `min.x`, `max.x`, `min.y`, `max.y`, `min.z`, `max.z`, `off.z`, `off.t`.

## Author(s)

Fisher, J. C.

## Examples

```
srvy()
```

---

`srvy.config`*Window and Plotting Parameters*

---

**Description**

A GUI used to define window geometry and universal plotting parameters.

**Usage**

```
srvy.config(ttl = NULL)
```

**Arguments**

`ttl` a Tk toplevel widget.

**Details**

The output date/time format is based on character representations described in [strptime](#), for example "%d/%m/%Y %H:%M:%OS".

**Value**

The following components within [srvy.dat](#) are set: `n.levels`, `win.width`, `yx.ratio`, `zx.ratio`, `date.fmt`.

**Author(s)**

Fisher, J. C.

**Examples**

```
srvy.config()
```

---

`srvy.dat`*Set or Query Survey Data*

---

**Description**

A function used to set or query survey parameters and data.

**Usage**

```
srvy.dat(option, value, clearAll = FALSE)
```

**Arguments**

<code>option</code>	the parameter name, described in the Survey Parameters section.
<code>value</code>	a parameter value specified for <code>option</code> .
<code>clearAll</code>	a logical; if <code>TRUE</code> all parameters are cleared from <code>srvy.dat</code> .

**Details**

The data frame `data.raw` has the following components:

<code>[,1]</code>	numeric	date/time, "%Y-%m-%d %H:%M:%OS" (optional).
<code>[,2]</code>	numeric	locations in the x-direction.
<code>[,3]</code>	numeric	locations in the y-direction.
<code>[,4:]</code>	numeric	state variables.

The data frame `data.mod` has the following components:

<code>[,1]</code>	x	numeric	locations in the x-direction.
<code>[,2]</code>	y	numeric	locations in the y-direction.
<code>[,3]</code>	z	numeric	state variable.

The list `data.tin` contains the following components:

<code>x</code>	numeric	locations in the x-direction.
<code>y</code>	numeric	locations in the y-direction.
<code>z</code>	matrix	rows and columns corresponding to grid lines in the x and y directions, respectively.

**Survey Parameters**

<code>cols</code>	character	fields within the imported survey data, a concatenation of column type and units.
<code>csi</code>	numeric	the height of plotted characters (in).
<code>data.file</code>	character	the imported text file containing survey data.
<code>data.raw</code>	data.frame	raw survey data, see Details section.
<code>data.mod</code>	data.frame	processed survey data, see Details.
<code>data.tin</code>	list	the interpolated surface derived from processed survey data, see Details.
<code>date.fmt</code>	character	the date/time format for exported data, see <a href="#">srvy.export</a>
<code>default.dir</code>	character	the default directory for saving and opening files.
<code>depth</code>	logical	if <code>TRUE</code> depths are converted to elevations during the interpolation.
<code>ed</code>	character	end date
<code>eh</code>	numeric	end hour
<code>em</code>	numeric	end minute
<code>encoding</code>	character	encoding to be assumed for input strings. Mark character strings as Latin-1 or UTF-8.
<code>es</code>	numeric	end second
<code>font.gui</code>	character	the GUI font, family and size.
<code>grad.tol</code>	numeric	tolerance value for state variable correction.
<code>grid.res</code>	numeric	the grid resolution for interpolation.
<code>max.vect</code>	numeric	the maximum length for plotting velocity vector arrows.
<code>max.x</code>	numeric	maximum x value

max.y	numeric	maximum y value
max.z	numeric	maximum z value
min.x	numeric	minimum x value
min.y	numeric	minimum y value
n.levels	numeric	the approximate number of contour intervals.
off.t	numeric	time offset in seconds.
off.z	numeric	state variable offset
poly	list	a polygon describing the spatial domain, class "gpc.poly".
poly.file	character	the imported polygon file.
proj.file	character	the binary project file.
projection	character	a description of the geographical projection in use.
sd	character	start date
sep	character	the field separator character. Values on each line of the file are separated by this character.
sh	numeric	start hour
sm	numeric	start minute
ss	numeric	start second
time.gap	numeric	the time gap exceedance level between sequential data records (sec).
vars	character	field names corresponding to data/time, x, y, and z.
ver	character	the version of <b>RSurvey</b> .
win.loc	character	the default horizontal and vertical location for GUI placement (pixels).
win.width	numeric	the default width for windows devices (in).
wtr.elev	numeric	the water surface elevation.
yx.ratio	numeric	the the y/x aspect ratio.
zx.ratio	numeric	the the z/x aspect ratio.

**Author(s)**

Fisher, J. C.

**Examples**

```
# To set a parameter
srvy.dat("grad.tol", 0.01)
srvy.dat("projection", "UTM")
# To get a parameter value
srvy.dat("grad.tol")
# To get all parameter values
srvy.dat()
# To clear all parameters
srvy.dat(clearAll = TRUE)
```

**Description**

This function exports survey data.

**Usage**

```
srvy.export(file = NULL)
```

**Arguments**

`file` either a character string naming a file or a connection.

**Value**

The output of `srvy.export` is dependent on the file type chosen within the file management pop up dialog box. Choosing a file type of `*.txt` writes the contents of `data.mod` to a file, whereas a choice of `*.tin` writes the contents of `data.tin` to a file.

**Author(s)**

Fisher, J. C.

**See Also**

[write.table](#)

**Examples**

```
data(confluence)
f <- paste(getwd(), "/test.txt", sep="")
con <- file(f, open = "w", encoding = "latin1")
srvy.export(file = con)
```

---

srvy.pref

*Error Correction, Interpolation, and Water Surface Parameters*

---

**Description**

A GUI defining error correction, interpolation, and water surface parameters and flags.

**Usage**

```
srvy.pref(ttl = NULL)
```

**Arguments**

`ttl` a Tk toplevel widget.

**Value**

The following components within `srvy.dat` are set: `grad.tol`, `time.gap`, `grid.res`, `wtr.elev`, `depth`.

**Author(s)**

Fisher, J. C.

**Examples**

```
srvy.pref()
```

`srvy.process`                      *Process Survey Data*

**Description**

This function limits the spatial and temporal domains of raw data and corrects for erroneous state variable measurements.

**Usage**

```
srvy.process(const.tin = FALSE)
```

**Arguments**

`const.tin`            a logical; if TRUE interpolation is used to construct a surface map of the state variable.

**Details**

Erroneous state variable measurements are identified by calculating the change in z over the change in time for sequential data records. Those gradients exceeding a user defined tolerance, `grad.tol` (see `srvy.dat`), are identified as erroneous measurements. After excluding these measurements the process is repeated until all gradients are less than the user specified tolerance. An exception to the tolerance constraint is made for sequential records whose time difference is greater than a specified time gap, `time.gap` (see `srvy.dat`).

**Value**

A data frame `data.mod` and list `data.tin` are stored within `srvy.dat`.

**Author(s)**

Fisher, J. C.

**See Also**

`mba.surf`

## Examples

```
data(confluence)
srvy.process(const.tin = TRUE)
```

---

srvy.vars

*Data Variables*

---

## Description

A GUI for identifying temporal and spatial variables associated with survey data.

## Usage

```
srvy.vars(cols, vars, ttl = NULL)
```

## Arguments

cols	those components within <code>srvy.dat</code> corresponding to temporal and spatial data, a concatenation of column type and units.
vars	the components corresponding to data/time, x, y, and z data.
ttl	a Tk toplevel widget.

## Details

The vars component is updated in `srvy.dat`.

## Author(s)

Fisher, J. C.

## Examples

```
data(confluence)
srvy.vars(srvy.dat("cols"), srvy.dat("vars"))
```

---

tran	<i>Transect Management</i>
------	----------------------------

---

**Description**

This function activates the main GUI for managing transect data.

**Usage**

```
tran(ttl = NULL)
```

**Arguments**

ttl                    a Tk toplevel widget.

**Author(s)**

Fisher, J. C.

**Examples**

```
tran()
```

---

tran.dat	<i>Set or Query Transect Data</i>
----------	-----------------------------------

---

**Description**

This function is used to set or query transect data.

**Usage**

```
tran.dat(id, option, value, clearId = FALSE, clearAll = FALSE)
```

**Arguments**

id                    the transect name.  
option                the parameter name, see Transect Parameters section.  
value                 a parameter value specified for option.  
clearId               a logical; if TRUE all data associated with transect id is cleared.  
clearAll              a logical; if TRUE all components are cleared from tran.dat.

### Transect Parameters

asp.ratio	numeric	the local z/x aspect ratio.
data.file	character	the file name associated with the raster data.
data.ras	data.frame	rows corresponding to raster point records and columns to the raster fields.
fix.zero	character	"R" for establishing the right vertex as the origin and "L" for the left.
grid.dx	numeric	the spatial resolution of the raster interpolation grid along the local x-axis.
grid.dy	numeric	the spatial resolution of the raster interpolation grid along the local z-axis.
h.offset	numeric	the local x-axis offset measured from the vertex origin.
id	character	the name of the transect.
limits	data.frame	rows corresponding to the raster types and columns to the raster plotting limits.
prof	data.frame	rows corresponding to points along the transect and columns to local x and z values.
type	character	fields within the imported raster data, a concatenation of column type and units.
v.offset	numeric	the local z-axis offset, positive upward.
v.origin	numeric	the z value corresponding to the local z-axis origin.
vel.vect	character	fields corresponding to velocity vector components: x, y, z, and principle direction.
vertices	matrix	rows corresponding to the transect vertices and columns to x and y values.
hv.fields	character	raster fields corresponding to the local-x and local-z directions.

### Author(s)

Fisher, J. C.

### Examples

```
# To set a parameter within a transect
tran.dat("T1", "h.offset", 10)
tran.dat("T1", "v.origin", 3.14)
tran.dat("T2", "h.offset", 9)
# To get a parameter value
tran.dat("T1", "h.offset")
# To get all transect data
tran.dat()
# To get transect data for T1
tran.dat("T1")
# To clear all data associated with a transect
tran.dat("T1", clearId = TRUE)
# To clear all transect data
tran.dat(clearAll = TRUE)
```

---

tran.edit

*Interpolation and Error Correction Parameters*

---

### Description

A GUI for defining parameters associated with a transect.

**Usage**

```
tran.edit(id = NULL, tt2 = NULL)
```

**Arguments**

id	the name of the transect.
tt2	a Tk toplevel widget.

**Value**

The following components within `tran.dat` are set: `x1`, `x2`, `y1`, `y2`, `fix.zero`, `v.origin`, `v.offset`, `h.offset`, `grid.dx`, `grid.dy`, `zero.int`, `asp.ration`.

**Author(s)**

Fisher, J. C.

**Examples**

```
tran.edit()
```

---

`tran.import-export` *Import and Export Transect Data*

---

**Description**

This function imports and exports transect data.

**Usage**

```
tran.import(type, id, file = NULL)
tran.export(type, id, file = NULL)
```

**Arguments**

type	a character string giving the desired data type ( <i>transect</i> , <i>profile</i> , and <i>raster</i> ). See Format section.
id	the name of the transect.
file	either a character string naming a file or a connection.

**Format**

For *transect* data the text file, '\*.txt', contains the deparsed `tran.dat` object.

For *profile* data the tab delineated text file, '\*.txt', reserves the first [1, ] row for column descriptors and contains the following components:

[, 1]	numeric	the distance along the local x-axis measured from the vertex origin.
[, 2]	numeric	z values along the profile.

For *raster* data the tab delimited text file, '\*.txt', reserves the first [1, ] and second [2, ] rows for column descriptors and contains the following components:

```
[, 1]    numeric  the distance along the local x-axis measured from the vertex origin.
[, 2]    numeric  local z values.
[, 3:]   numeric  raster state variable(s).
```

### Value

The *transect* option replaces or creates the *id* component in *tran.dat*. The *profile* and *raster* options replace the *prof* and *data.ras* components in *tran.dat*, respectively.

### Author(s)

Fisher, J. C.

### See Also

[dput](#), [read.table](#)

### Examples

```
## A sample transect file:

## structure(list(id = "T1", vertices = structure(c(679163.534, 679238.561,
## 4135735.518, 4135730.43), .Dim = c(2L, 2L), .Dimnames = list(c("1", "2"),
## c("x", "y"))), fix.zero = "R", h.offset = 21, grid.dx = 0.05, grid.dy = 0.005,
## asp.ratio = 5, v.origin = 11.379, v.offset = -0.18, data.file = "T1.txt"),
## .Names = c("id", "vertices", "fix.zero", "h.offset", "grid.dx", "grid.dy",
## "asp.ratio", "v.origin", "v.offset", "data.file"))

f <- system.file("RSurvey-ex/confluence-T1.txt", package = "RSurvey")
tran.import("transect", "T1", f)

f <- paste(getwd(), "/test.txt", sep="")
tran.export("transect", "T1", f)

## A sample profile file including a header and two rows of data.

## Distance      Elevation
## 14      10.499
## 16      10.429

f <- system.file("RSurvey-ex/confluence-profile-T1.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
tran.import(type = "profile", id = "T1", file = con)

f <- paste(getwd(), "/test.txt", sep="")
con <- file(f, open = "w")
tran.export(type = "profile", id = "T1", file = con)
```

```
## A sample raster file including a header and two rows of data.

## motoX      motoY  temp   pH      SpCond
## m         m      degC   units  uS/cm
## 0         -0.11  24.55  8.00   569.38
## 2         0      24.58  7.99   646.01

f <- system.file("RSurvey-ex/confluence-raster-T1.txt", package = "RSurvey")
con <- file(f, open = "r", encoding = "latin1")
tran.import(type = "raster", id = "T1", file = con)

f <- paste(getwd(), "/test.txt", sep="")
con <- file(f, open = "w")
tran.export("T1", type="raster", file = con)
```

---

tran.profile	<i>Calculate the Profile Along a Transect</i>
--------------	---

---

## Description

This function determines the profile spatial coordinates along a transect.

## Usage

```
tran.profile(vertices)
```

## Arguments

`vertices` a matrix with rows corresponding to the transect vertices and columns to spatial coordinates.

## Value

A matrix with rows corresponding to locations along a profile. Variables associated with each location include:

[, 1]	x	numeric	locations on the x-axis
[, 2]	y	numeric	locations on the y-axis
[, 3]	h	numeric	locations on the local x-axis.
[, 4]	z	numeric	locations on the z-axis

## Author(s)

Fisher, J. C.

## Examples

```
data(confluence)
```

```
vert <- matrix(c(679164, 4135733, 679239, 4135730), nrow = 2, ncol = 2,  
              byrow = TRUE, dimnames = list(c("1", "2"), c("x", "y")))  
prof <- tran.profile(vert)  
prof
```

---

tran.spatial	<i>Spatial Fields for Raster Data</i>
--------------	---------------------------------------

---

### Description

Identify raster fields corresponding to spatial coordinates.

### Usage

```
tran.spatial(rasterFields, spatialFields = NULL, tt2 = NULL)
```

### Arguments

`rasterFields` an array giving all raster fields.  
`spatialFields` current raster fields corresponding to the spatial coordinates  $(x, y)$ .  
`tt2` a Tk toplevel widget.

### Value

A character array containing field names associated with the local x-axis [1] and local z-axis [2] coordinates.

### Author(s)

Fisher, J. C.

### Examples

```
tran.spatial(c("X (m)", "Z (m)", "T (degC)"), c("X (m)", "Z (m)"))
```

---

tran.vector                      *Longitudinal and Transverse Velocity Calculations*

---

### Description

Transform velocity components from x and y directions to longitudinal and transverse directions.

### Usage

```
tran.vector(vertices, vel.vect, arrow.max)
```

### Arguments

vertices	a matrix with rows corresponding to the transect vertices and columns to x and y values.
vel.vect	a data frame with rows corresponding to the raster point records and columns to vector components, see Format section for further details.
arrow.max	the maximum transverse velocity arrow symbol length.

### Format

The components associated with `vel.vect` are as follows:

[,1]	x	numeric	velocity component in the x-direction.
[,2]	y	numeric	velocity component in the y-direction.
[,3]	z	numeric	velocity component in the z-direction, positive upward.

### Value

A data frame with rows corresponding to points along a profile. Components associated with each point include:

[,1]	magn	numeric	the magnitude of the velocity within the principle direction.
[,2]	long	numeric	longitudinal velocity component, perpendicular to the local x-direction.
[,3]	arrow.x	numeric	the velocity component in the local x-direction.
[,4]	arrow.y	numeric	the velocity component in the local z-direction.

### Author(s)

Fisher, J. C.

### Examples

```
vert <- matrix(c(0, 0, 10, 10), nrow = 2, ncol = 2,
              byrow = TRUE, dimnames = list(c("1", "2"), c("x", "y")))
vect <- data.frame(cbind(x = c(1, -2, -3), y = c(-1, 2, -3), z = c(-1, 1, -1)))
tran.vector(vert, vect, 20)
```

```
##          magn          long          arrow.x  arrow.y
## [1,] 1.732051 -1.414214  2.809440e-16 -4.588315
## [2,] 3.000000  2.828427 -1.299754e-15  4.588315
## [3,] 4.358899  0.000000 -1.946657e+01 -4.588315
```

---

tran.velocity	<i>Velocity Components for Raster Data</i>
---------------	--

---

### Description

A GUI for identifying raster fields corresponding to velocity components.

### Usage

```
tran.velocity(rasterFields, velocityFields = NULL, tt2 = NULL)
```

### Arguments

`rasterFields` an array containing all raster field names.

`velocityFields`  
current raster fields corresponding to the x, y, z, and principle velocity components.

`tt2` a Tk toplevel widget.

### Value

A character array of field names associated with velocity components in the x [1], y [2], z [3], and principle [4] directions.

### Author(s)

Fisher, J. C.

### Examples

```
tran.velocity(c("VelE (m/s)", "VelN (m/s)", "VelU (m/s)", "VelM (m/s)"))
```

# Index

- \*Topic **aplot**
  - minorTics, 9
- \*Topic **datasets**
  - confluence, 4
- \*Topic **file**
  - readFile, 16
  - srvy.export, 22
  - tran.import-export, 28
- \*Topic **hplot**
  - plotPoints, 10
  - plotSurface2d, 10
  - plotSurface3d, 11
  - plotTime, 12
  - plotTransect, 12
  - tran.profile, 30
- \*Topic **manip**
  - polyCutout, 15
  - srvy.process, 24
- \*Topic **misc**
  - autocrop, 2
  - axesLimits, 3
  - geoProj, 5
  - getFile, 5
  - keyEvent, 6
  - loadPackages, 7
  - mergeData, 8
  - polyAutocrop, 13
  - polyConstruct, 14
  - srvy, 19
  - srvy.config, 20
  - srvy.pref, 23
  - srvy.vars, 25
  - tran, 26
  - tran.edit, 27
  - tran.spatial, 31
  - tran.vector, 32
  - tran.velocity, 33
- \*Topic **package**
  - RSurvey-package, 1
- \*Topic **programming**
  - buildPackage, 4
  - restoreSession, 18
- \*Topic **sysdata**
  - srvy.dat, 20
  - tran.dat, 26
- approx, 8
- autocrop, 2, 14
- axesLimits, 3
- axis, 9
- buildPackage, 4
- confluence, 4
- dput, 29
- filled.contour, 11, 13
- geoProj, 5
- getFile, 5
- keyEvent, 6
- loadPackages, 7
- mergeData, 8
- minorTics, 9
- plot, 10, 12
- plotPoints, 10
- plotSurface2d, 10
- plotSurface3d, 11
- plotTime, 12
- plotTransect, 12
- polyAutocrop, 3, 13
- polyConstruct, 14
- polyCutout, 15
- read.table, 17, 29
- readFile, 16, 19

restoreSession, 18  
RSurvey-package, 1

source, 18  
srvy, 19  
srvy.config, 20  
srvy.dat, 4, 5, 7, 8, 13, 18, 19, 20, 20, 24,  
25  
srvy.export, 21, 22  
srvy.pref, 23  
srvy.process, 24  
srvy.vars, 25  
strptime, 8, 17, 20

tkentry, 7  
tran, 26  
tran.dat, 4, 12, 18, 26, 28, 29  
tran.edit, 27  
tran.export (*tran.import-export*),  
28  
tran.import (*tran.import-export*),  
28  
tran.import-export, 28  
tran.profile, 13, 30  
tran.spatial, 31  
tran.vector, 32  
tran.velocity, 33

write.table, 23