# Package 'pts2polys'

April 15, 2019

**Type** Package

**Title** Construct Polygons Summarising the Location and Variability of Point Sets

**Version** 0.1.1

**Date** 2019-04-11

**Description** Various applications in invasive species biology, conservation biology, epidemiology and elsewhere involve sampling of sets of 2D points from a posterior distribution. The number of such point sets may be large, say 1000 or 10000. This package facilitates visualisation of such output by constructing seven nested polygons representing the location and variability of the point sets. This can be used, for example, to visualise the range boundary of a species, and uncertainty in the location of that boundary.

**License** GPL (>= 2)

**Imports** Rcpp (>= 1.0.1)

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Author** Jonathan Keith [aut, cre],
Ken Clarkson [aut],
Eric Hufschmid [ctb],
AT&T [cph]

**Maintainer** Jonathan Keith <jonathan.keith@monash.edu>

**Repository** CRAN

**Date/Publication** 2019-04-15 08:02:38 UTC

## R topics documented:

pts2polys-package *Construct polygons from point sets*

## Description

Various applications in invasive species biology, conservation biology, epidemiology and elsewhere involve sampling of sets of 2D points from a posterior distribution. The number of such point sets may be large, say 1000 or 10000. This package facilitates visualisation of such output by constructing seven nested polygons representing the location and variability of the point sets. This can be used, for example, to visualise the range boundary of a species, and uncertainty in the location of that boundary.

The function pts2polys takes multiple sets of points as input and in Stage 1 constructs a chi-shape enclosing each point set. In Stage 2, it outputs chi-shapes enclosing grid points that are interior to a specified proportion of the chi-shapes from Stage 1.

The method implemented in this package is described in a manuscript entitled "Delimiting a species' geographic range using posterior sampling and computational geometry". The manuscript is currently under review.

The package depends on code for constructing a Delaunay triangulation originally written by Ken Clarkson, modified by Eric Hufschmid and further modified by Jonathan Keith. Ken Clarkson's original code included the following notice:

"Ken Clarkson wrote this. Copyright (c) 1995 by AT&T.. Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software."

## Usage

```
pts2polys(in_string,SAMPLESIZE,MINLEN,GRIDSIZE,MINX,MAXX,MINY,MAXY)
```

## Arguments

| | |
|---|---|
| in_string | name of input file containing point sets. This is a text file in which each point set begins with a line starting with 'P' followed by one line per point, with each line containing an x and y coordinate |
| SAMPLESIZE | number of point sets |
| MINLEN | minimum length of external edges to be removed |
| GRIDSIZE | spacing between points in a square tiling |
| MINX | minimum x co-ordinate |
| MAXX | maximum x co-ordinate |
| MINY | minimum y co-ordinate |
| MAXY | maximum y co-ordinate |

## Value

A list comprised of two elements: boundarylen and boundary. 'boundarylen' is an integer vector with 7 elements, representing the number of points in each of 7 polygons. 'boundary' is a list comprised of 7 numeric vectors, corresponding to alpha = 0.999, 0.990, 0.975, 0.750, 0.500, 0.250 and 0.025. Each vector is comprised of alternating x and y co-ordinates.

## Author(s)

Author and Maintainer: Jonathan Keith <<jonathan.keith@monash.edu>>

Author: Ken Clarkson

Contributor: Eric Hufschmid



## Examples

```
pol <- pts2polys(system.file("extdata", "test.pt.sets",
package = "pts2polys", mustWork = TRUE),
20,10000,100,293706,564106,6803848,7076948)

plot(pol$boundary[[1]][c((1:pol$boundarylen[1])*2-1,1)],
pol$boundary[[1]][c((1:pol$boundarylen[1])*2,2)],type="l",
xlab="Easting",ylab="Northing")

lines(pol$boundary[[2]][c((1:pol$boundarylen[2])*2-1,1)],
pol$boundary[[2]][c((1:pol$boundarylen[2])*2,2)],col=2)

lines(pol$boundary[[3]][c((1:pol$boundarylen[3])*2-1,1)],
pol$boundary[[3]][c((1:pol$boundarylen[3])*2,2)],col=3)

lines(pol$boundary[[4]][c((1:pol$boundarylen[4])*2-1,1)],
pol$boundary[[4]][c((1:pol$boundarylen[4])*2,2)],col=4)

lines(pol$boundary[[5]][c((1:pol$boundarylen[5])*2-1,1)],
pol$boundary[[5]][c((1:pol$boundarylen[5])*2,2)],col=5)

lines(pol$boundary[[6]][c((1:pol$boundarylen[6])*2-1,1)],
pol$boundary[[6]][c((1:pol$boundarylen[6])*2,2)],col=6)

lines(pol$boundary[[7]][c((1:pol$boundarylen[7])*2-1,1)],
pol$boundary[[7]][c((1:pol$boundarylen[7])*2,2)],col=7)
```

# Index