Package 'gpclib'

July 2, 2014

Depends R (>= 2.14.0), methods

Imports graphics

LazyLoad yes

Version 1.5-5

Date 2013-04-01

Title General Polygon Clipping Library for R

Author Roger D. Peng <rpeng@jhsph.edu> with contributions from Duncan Murdoch and Barry Rowlingson; GPC library by Alan Murta

Maintainer Roger D. Peng <rpeng@jhsph.edu>

Description General polygon clipping routines for R based on Alan Murta's C library

License file LICENSE

URL http://www.cs.man.ac.uk/~toby/gpc/,http://github.com/rdpeng/gpclib

NeedsCompilation yes

License_restricts_use yes

Repository CRAN

Date/Publication 2013-04-01 20:03:33

R topics documented:

gpc.poly-class 2
gpc.poly.nohole-class 4
new-generics 5
polyfile 6

Index **9**

gpc.poly-class *Class "gpc.poly"*

Description

A class for representing polygons composed of multiple contours, some of which may be holes.

Objects from the Class

Objects can be created by calls of the form `new("gpc.poly", ...)` or by reading in from a file using `read.polyfile`.

Slots

pts Object of class "list". Actually, `pts` is a list of lists with length equal to the number of contours in the `"gpc.poly"` object. Each element of `pts` is a list of length 3 with names `x`, `y`, and `hole`. `x` and `y` are vectors containing the `x` and `y` coordinates, respectively, while `hole` is a logical indicating whether or not the contour is a hole.

Methods

[`signature(x = "gpc.poly")`: ...

append.poly `signature(x = "gpc.poly", y = "gpc.poly")`: ...

area.poly `signature(object = "gpc.poly")`: ...

coerce `signature(from = "matrix", to = "gpc.poly")`: ...

coerce `signature(from = "data.frame", to = "gpc.poly")`: ...

coerce `signature(from = "gpc.poly", to = "matrix")`: ...

coerce `signature(from = "gpc.poly", to = "numeric")`: ...

coerce `signature(from = "numeric", to = "gpc.poly")`: ...

coerce `signature(from = "list", to = "gpc.poly")`: ...

get.bbox `signature(x = "gpc.poly")`: ...

get.pts `signature(object = "gpc.poly")`: ...

intersect `signature(x = "gpc.poly", y = "gpc.poly")`: ...

plot `signature(x = "gpc.poly")`: The argument `poly.args` can be used to pass a list of additional arguments to be passed to the underlying `polygon` call.

scale.poly `signature(x = "gpc.poly")`: ...

setdiff `signature(x = "gpc.poly", y = "gpc.poly")`: ...

show `signature(object = "gpc.poly")`: Scale `x` and `y` coordinates by amount `xscale` and `yscale`. By default `xscale` equals `yscale`.

union `signature(x = "gpc.poly", y = "gpc.poly")`: ...

tristrip `signature(x = "gpc.poly")`: ...

triangulate `signature(x = "gpc.poly")`: ...

Note

The class "gpc.poly.nohole" is identical to "gpc.poly" except the hole flag for each contour of a "gpc.poly.nohole" object is always FALSE.

Author(s)

Roger D. Peng

Examples

```
## Make some random polygons
set.seed(100)
a <- cbind(rnorm(100), rnorm(100))
a <- a[chull(a), ]

## Convert `a' from matrix to "gpc.poly"
a <- as(a, "gpc.poly")

b <- cbind(rnorm(100), rnorm(100))
b <- as(b[chull(b), ], "gpc.poly")

## More complex polygons with an intersection
p1 <- read.polyfile(system.file("poly-ex/ex-poly1.txt", package = "gpclib"))
p2 <- read.polyfile(system.file("poly-ex/ex-poly2.txt", package = "gpclib"))

## Plot both polygons and highlight their intersection in red
plot(append.poly(p1, p2))
plot(intersect(p1, p2), poly.args = list(col = 2), add = TRUE)

## Highlight the difference p1 \ p2 in green
plot(setdiff(p1, p2), poly.args = list(col = 3), add = TRUE)

## Highlight the difference p2 \ p1 in blue
plot(setdiff(p2, p1), poly.args = list(col = 4), add = TRUE)

## Plot the union of the two polygons
plot(union(p1, p2))

## Take the non-intersect portions and create a new polygon
## combining the two contours
p.comb <- append.poly(setdiff(p1, p2), setdiff(p2, p1))
plot(p.comb, poly.args = list(col = 2, border = 0))

## Coerce from a matrix
x <-
structure(c(0.0934073560027759, 0.192713393476752, 0.410062456627342,
0.470020818875781, 0.41380985426787, 0.271408743927828, 0.100902151283831,
0.0465648854961832, 0.63981588032221, 0.772382048331416,
0.753739930955121, 0.637744533947066, 0.455466052934407,
0.335327963176065, 0.399539700805524,
0.600460299194476), .Dim = c(8, 2))
y <-
```

```
structure(c(0.404441360166551, 0.338861901457321, 0.301387925052047,
0.404441360166551, 0.531852879944483, 0.60117973629424, 0.625537820957668,
0.179976985040276, 0.341542002301496, 0.445109321058688,
0.610817031070196, 0.596317606444189, 0.459608745684695,
0.215189873417722), .Dim = c(7, 2))

x1 <- as(x, "gpc.poly")
y1 <- as(y, "gpc.poly")

plot(append.poly(x1, y1))
plot(intersect(x1, y1), poly.args = list(col = 2), add = TRUE)

## Show the triangulation
plot(append.poly(x1, y1))
triangles <- triangulate(append.poly(x1,y1))
for (i in 0:(nrow(triangles)/3 - 1))
    polygon(triangles[3*i + 1:3,], col="lightblue")
```

`gpc.poly.nohole-class` *Class "gpc.poly.nohole"*

Description

A class for representing polygons with multiple contours but without holes.

Objects from the Class

Objects can be created by calls of the form 'new("gpc.poly.nohole", ...) or by calling `read.polyfile`'.

Slots

pts Object of class "list". See the help for "gpc.poly" for details.

Extends

Class "gpc.poly", directly.

Methods

coerce `signature(from = "numeric", to = "gpc.poly.nohole")`: ...

Note

This class is identical to ""gpc.poly"" and is needed because the file formats for polygons without holes is slightly different from the file format for polygons with holes. For a "gpc.poly.nohole" object, the hole flag for each contour is always `FALSE`.

Also, `write.polyfile` will write the correct file format, depending on whether the object is of class "gpc.poly" or "gpc.poly.nohole".

Author(s)

Roger D. Peng

See Also

gpc.poly-class

Examples

```
## None
```

`new-generics` *Generics/Methods for polygon objects*

Description

Some generic functions and methods for polygon objects

Usage

```
append.poly(x, y)
area.poly(object, ...)
get.pts(object)
get.bbox(x)
scale.poly(x, ...)
tristrip(x)
triangulate(x)
```

Arguments

`x,object`	A polygon object
`y`	A polygon object
`...`	Other arguments passed to methods

Details

The result of `tristrip(x)` is a list of two-column matrices. Each matrix is a tristrip, i.e. the rows are vertices of triangles, with each overlapping triple of rows corresponding to a separate triangle.

The result of `triangulate(x)` is a single two-column matrix. The rows are vertices of triangles, taken in non-overlapping triples.

Methods

append.poly `signature(x = "gpc.poly", y = "gpc.poly")`: Combine all contours of two `"gpc.poly"` objects and return the combined polygon as a `"gpc.poly"` object.

area.poly `signature(object = "gpc.poly")`: Compute and return the sum of the areas of all contours in a `"gpc.poly"` object.

scale.poly `signature(x = "gpc.poly")`: Scale (divide) the x and y coordinates of a `"gpc.poly"` object by the amount `xscale` and `yscale`, respectively. Return a scaled `"gpc.poly"` object.

get.pts `signature(object = "gpc.poly")`: Return the list of x and y coordinates of the vertices of a `"gpc.poly"` object.

get.bbox `signature(x = "gpc.poly")`: Return the bounding box for a `"gpc.poly"` object.

tristrip `signature(x = "gpc.poly")`: Return a tristrip list for a `"gpc.poly"` object.

triangulate `signature(x = "gpc.poly")`: Return a matrix of vertices of a triangulation of a `"gpc.poly"` object.

Author(s)

Roger D. Peng; GPC Library by Alan Murta; tristrip additions by Duncan Murdoch

See Also

`"gpc.poly"` class documentation.

Examples

```
holepoly <- read.polyfile(system.file("poly-ex/hole-poly.txt", package =
"gpclib"), nohole = FALSE)
area.poly(holepoly)
stopifnot(area.poly(holepoly) == 8)
```

---

polyfile *Read/Write polygon data*

Description

Read/Write polygon and contour information from/to a text file.

Usage

```
read.polyfile(filename, nohole = TRUE)
write.polyfile(poly, filename = "GPCpoly.txt")
```

Arguments

`filename`	the name of the file (a character string) from/to which data should be read/written.
`nohole`	Is this a polygon without holes?
`poly`	an object of class `"gpc.poly"`

Details

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

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

For example, a data file for a polygon with 2 contours (a four-sided object and a triangle) might look like:

2
4
1.0 1.0
1.0 2.0
3.4 3.21
10 11.2
3
21.0 11.2
22.3 99.2
4.5 5.4

The vertices of the polygon can be ordered either clockwise or counter-clockwise.

If a polygon has contours which are holes, then the format is slightly different. Basically, a flag is set to indicate that a particular contour is a hole. The format is

<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. For example, a four-sided polygon with a triangular hole would be written as:

2
3
1

4.0 4.0
6.0 5.0
5.0 6.0
4
0
2.0 1.0
8.0 2.0
7.0 9.0
1.0 7.0

Value

If `nohole` is `TRUE` (the default) `read.polyfile` returns an object of class `"gpc.poly.nohole"`. This object has the hole flag set to `FALSE` for all contours. If `nohole` is `FALSE`, then an object of class `"gpc.poly"` is returned.

`write.polyfile` does not return anything useful.

Author(s)

Roger D. Peng

See Also

`gpc.poly-class`, `gpc.poly.nohole-class`

Examples

```
## None right now.
```

Index

∗Topic **IO**
    polyfile, 6
∗Topic **classes**
    gpc.poly-class, 2
    gpc.poly.nohole-class, 4
∗Topic **methods**
    new-generics, 5
[,gpc.poly,ANY,ANY-method
        (gpc.poly-class), 2
[,gpc.poly-method *(gpc.poly-class)*, 2

append.poly *(new-generics)*, 5
append.poly,gpc.poly,gpc.poly-method
        (gpc.poly-class), 2
append.poly-methods *(new-generics)*, 5
area.poly *(new-generics)*, 5
area.poly,gpc.poly-method
        (gpc.poly-class), 2
area.poly-methods *(new-generics)*, 5

coerce,data.frame,gpc.poly-method
        (gpc.poly-class), 2
coerce,gpc.poly,matrix-method
        (gpc.poly-class), 2
coerce,gpc.poly,numeric-method
        (gpc.poly-class), 2
coerce,list,gpc.poly-method
        (gpc.poly-class), 2
coerce,matrix,gpc.poly-method
        (gpc.poly-class), 2
coerce,numeric,gpc.poly-method
        (gpc.poly-class), 2
coerce,numeric,gpc.poly.nohole-method
        (gpc.poly.nohole-class), 4

get.bbox *(new-generics)*, 5
get.bbox,gpc.poly-method
        (gpc.poly-class), 2
get.bbox-methods *(new-generics)*, 5
get.pts *(new-generics)*, 5
get.pts,gpc.poly-method
        (gpc.poly-class), 2
get.pts-methods *(new-generics)*, 5
gpc.poly-class, 2
gpc.poly.nohole-class, 4

intersect,gpc.poly,gpc.poly-method
        (gpc.poly-class), 2

new-generics, 5

plot,gpc.poly-method *(gpc.poly-class)*, 2
polyfile, 6

read.polyfile *(polyfile)*, 6

scale.poly *(new-generics)*, 5
scale.poly,gpc.poly-method
        (gpc.poly-class), 2
scale.poly-methods *(new-generics)*, 5
setdiff,gpc.poly,gpc.poly-method
        (gpc.poly-class), 2
show,gpc.poly-method *(gpc.poly-class)*, 2

triangulate *(new-generics)*, 5
triangulate,gpc.poly-method
        (gpc.poly-class), 2
triangulate-methods *(new-generics)*, 5
tristrip *(new-generics)*, 5
tristrip,gpc.poly-method
        (gpc.poly-class), 2
tristrip-methods *(new-generics)*, 5

union,gpc.poly,gpc.poly-method
        (gpc.poly-class), 2

write.polyfile *(polyfile)*, 6