Package ‘polyclip’

May 27, 2015

Version 1.3-2
Date 2015-05-27
Title Polygon Clipping
Author Angus Johnson. Ported to R by Adrian Baddeley and Brian Ripley.
Maintainer Adrian Baddeley <Adrian.Baddeley@uwa.edu.au>
Depends R (>= 3.0.0)
Description R port of the Clipper library. Performs polygon clipping operations (intersection, union, set minus, set difference) for polygonal regions of arbitrary complexity, including holes. Also computes offset polygons (spatial buffer zones, morphological dilations, Minkowski dilations) for polygonal regions and polygonal lines.
License BSL
LazyData true
LazyLoad true
ByteCompile true
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-05-27 09:01:34

R topics documented:

polyclip ................................................................. 2
polylineoffset .......................................................... 4
polyoffset ............................................................... 6
Index 9
polyclip

Polygon Clipping

Description

Find intersection, union or set difference of two polygonal regions.

Usage

polyclip(A, B, op=c("intersection", "union", "minus", "xor"),
  ..., 
  eps, x0, y0,
  fillA=c("evenodd", "nonzero", "positive", "negative"),
  fillB=c("evenodd", "nonzero", "positive", "negative"))

Arguments

A, B  Data specifying polygons. See Details.
op  Set operation to be performed to combine A and B.
...  Ignored.
eps  Spatial resolution for coordinates.
x0, y0  Spatial origin for coordinates.
fillA, fillB  Polygon-filling rule for A and B.

Details

This is an interface to the polygon-clipping library Clipper written by Angus Johnson.

Given two polygonal regions A and B the function polyclip performs one of the following geometrical operations:

- op="intersection": set intersection of A and B.
- op="union": set union of A and B.
- op="minus": set subtraction (sometimes called set difference): the region covered by A that is not covered by B.
- op="xor": exclusive set difference (sometimes called exclusive-or): the region covered by exactly one of the sets A and B.

Each of the arguments A and B represents a region in the Euclidean plane bounded by closed polygons. The format of these arguments is either

- a list containing two components x and y giving the coordinates of the vertices of a single polygon. The last vertex should not repeat the first vertex.
- a list of list(x,y) structures giving the coordinates of the vertices of several polygons.
Note that calculations are performed in integer arithmetic: see below.

The interpretation of the polygons depends on the polygon-filling rule for A and B that is specified by the arguments fillA and fillB respectively.

**Even-Odd:** The default rule is *even-odd* filling, in which every polygon edge demarcates a boundary between the inside and outside of the region. It does not matter whether a polygon is traversed in clockwise or anticlockwise order. Holes are determined simply by their locations relative to other polygons such that outers contain holes and holes contain outers.

**Non-Zero:** Under the *nonzero* filling rule, an outer boundary must be traversed in clockwise order, while a hole must be traversed in anticlockwise order.

**Positive:** Under the positive filling rule, the filled region consists of all points with positive winding number.

**Negative:** Under the negative filling rule, the filled region consists of all points with negative winding number.

**Calculations are performed in integer arithmetic** after subtracting $x_0, y_0$ from the coordinates, dividing by $\text{eps}$, and rounding to the nearest integer. Thus, $\text{eps}$ is the effective spatial resolution. The default values ensure reasonable accuracy.

**Value**

Data specifying polygons, in the same format as A and B.

**Author(s)**

Angus Johnson. Ported to R by Adrian Baddeley <Adrian.Baddeley@uwa.edu.au>.

**References**

Clipper Website: [http://www.angusj.com](http://www.angusj.com)


**Examples**

```r
A <- list(list(x=1:10, y=c(1:5,5:1)))
B <- list(list(x=c(2,8,8,2), y=c(0,0,10,10)))
plot(c(0,10),c(0,10), type="n", axes=FALSE, xlab="", ylab="")
polygon(A[[1]])
polygon(B[[1]])
C <- polyclip(A, B)
polygon(C[[1]], lwd=3, col=3)
```
polylineoffset  

**Polygonal Line Offset**

**Description**

Given a list of polygonal lines, compute the offset region (guard region, buffer region, morphological dilation) formed by shifting the boundary outwards by a specified distance.

**Usage**

```
polylineoffset(A, delta,  
    ...,  
    eps, x0, y0,  
    miterlim=2, arctol=abs(delta)/100,  
    jointype=c("square", "round", "miter"),  
    endtype = c("closedpolygon", "closedline",  
                "openbutt", "opensquare", "openround",  
                "closed", "butt", "square", "round"))
```

**Arguments**

- `A`  
  Data specifying polygons. See Details.

- `delta`  
  Distance over which the boundary should be shifted.

- `...`  
  Ignored.

- `eps`  
  Spatial resolution for coordinates.

- `x0, y0`  
  Spatial origin for coordinates.

- `miterlim, arctol`  
  Tolerance parameters: see Details.

- `jointype`  
  Type of join operation to be performed at each vertex. See Details.

- `endtype`  
  Type of geometrical operation to be performed at the start and end of each line. See Details.

**Details**

This is part of an interface to the polygon-clipping library Clipper written by Angus Johnson.

Given a list of polygonal lines `A`, the function `polylineoffset` computes the offset region (also known as the morphological dilation, guard region, buffer region, etc) obtained by shifting the boundary of `A` outward by the distance `delta`.

The argument `A` represents a polygonal line (broken line) or a list of broken lines. The format is either

- a list containing two components `x` and `y` giving the coordinates of successive vertices of the broken line.
- a list of list(`x, y`) structures giving the coordinates of the vertices of several broken lines.
Lines may be self-intersecting and different lines may intersect each other. Note that calculations are performed in integer arithmetic: see below.

The argument jointype determines what happens at the vertices of each line. See the Examples for illustrations.

- jointype=“round”: a circular arc is generated.
- jointype=“square”: the circular arc is replaced by a single straight line.
- jointype=“miter”: the circular arc is omitted entirely, or replaced by a single straight line.

The argument endtype determines what happens at the beginning and end of each line. See the Examples for illustrations.

- endtype=“closedpolygon”: ends are joined together (using the jointype value) and the path filled as a polygon.
- endtype=“closedline”: ends are joined together (using the jointype value) and the path is filled as a polyline.
- endtype=“openbutt”: ends are squared off abruptly.
- endtype=“opensquare”: ends are squared off at distance delta.
- endtype=“openround”: ends are replaced by a semicircular arc.

The values endtype=“closed”, “butt”, “square” and “round” are deprecated; they are equivalent to endtype=“closedpolygon”, “openbutt”, “opensquare” and “openround” respectively.

The arguments miterlim and arctol are tolerances.

- if jointype=“round”, then arctol is the maximum permissible distance between the true circular arc and its discretised approximation.
- if jointype=“miter”, then miterlimit * delta is the maximum permissible displacement between the original vertex and the corresponding offset vertex if the circular arc were to be omitted entirely. The default is miterlimit=2 which is also the minimum value.

Calculations are performed in integer arithmetic after subtracting x0, y0 from the coordinates, dividing by eps, and rounding to the nearest integer. Thus, eps is the effective spatial resolution. The default values ensure reasonable accuracy.

Value

Data specifying polygons, in the same format as \texttt{A}.

Author(s)

Angus Johnson. Ported to \texttt{R} by Adrian Baddeley \texttt{<Adrian.Baddeley@uwa.edu.au>}.

References

Clipper Website: \url{http://www.angusj.com}


**Examples**

```r
A <- list(list(x=c(4,8,8,2), y=c(3,3,8,6)))

plot(c(0,10),c(0,10), type="n",
     main="jointype=square, endtype=opensquare",
     axes=FALSE, xlab="", ylab="")
lines(A[[1]], col="grey", lwd=3)
C <- polylineoffset(A, 0.5, jointype="square", endtype="opensquare")
polygon(C[[1]], lwd=3, border="blue")

plot(c(0,10),c(0,10), type="n",
     main="jointype=round, endtype=openround",
     axes=FALSE, xlab="", ylab="")
lines(A[[1]], col="grey", lwd=3)
C <- polylineoffset(A, 0.5, jointype="round", endtype="openround")
polygon(C[[1]], lwd=3, border="blue")
```

---

**polyoffset**  

documented  

**Polygon Offset**

**Description**

Given a polygonal region, compute the offset region (aka: guard region, buffer region, morphological dilation) formed by shifting the boundary outwards by a specified distance.

**Usage**

```r
polyoffset(A, delta,
           ...,
           eps, x0, y0,
           miterlim=2, arctol=abs(delta)/100,
           jointype=c("square", "round", "miter"))
```

**Arguments**

- `A`  
  Data specifying polygons. See Details.
- `delta`  
  Distance over which the boundary should be shifted.
- `...`  
  Ignored.
- `eps`  
  Spatial resolution for coordinates.
polyoffset

x0, y0  Spatial origin for coordinates.
miterlim, arctol  Tolerance parameters: see Details.
jointype  Type of join operation to be performed at each vertex. See Details.

Details

This is part of an interface to the polygon-clipping library Clipper written by Angus Johnson. Given a polygonal region A, the function polyoffset computes the offset region (also known as the morphological dilation, guard region, buffer region, etc) obtained by shifting the boundary of A outward by the distance delta.

The argument A represents a region in the Euclidean plane bounded by closed polygons. The format is either

- a list containing two components x and y giving the coordinates of the vertices of a single polygon. The last vertex should not repeat the first vertex.
- a list of list(x, y) structures giving the coordinates of the vertices of several polygons.

Note that calculations are performed in integer arithmetic: see below.

The argument jointype determines what happens at the convex vertices of A. See the Examples for illustrations.

- jointype="round": a circular arc is generated.
- jointype="square": the circular arc is replaced by a single straight line.
- jointype="miter": the circular arc is omitted entirely, or replaced by a single straight line.

The arguments miterlim and arctol are tolerances.

- if jointype="round", then arctol is the maximum permissible distance between the true circular arc and its discretised approximation.
- if jointype="miter", then miterlim * delta is the maximum permissible displacement between the original vertex and the corresponding offset vertex if the circular arc were to be omitted entirely. The default is miterlim=2 which is also the minimum value.

Calculations are performed in integer arithmetic after subtracting x0, y0 from the coordinates, dividing by eps, and rounding to the nearest 64-bit integer. Thus, eps is the effective spatial resolution. The default values ensure reasonable accuracy.

Value

Data specifying polygons, in the same format as A.

Author(s)

Angus Johnson. Ported to R by Adrian Baddeley <Adrian.Baddeley@uwa.edu.au>.
References

Clipper Website: http://www.angusj.com

http://portal.acm.org/citation.cfm?id=129906


Examples

```r
A <- list(list(x=c(4,8,8,2,6), y=c(3,3,8,8,6)))
plot(c(0,10),c(0,10), type="n", main="jointype=square", axes=FALSE, xlab="", ylab="")
polygon(A[[1]], col="grey")
C <- polyoffset(A, 1, jointype="square")
polygon(C[[1]], lwd=3, border="blue")
plot(c(0,10),c(0,10), type="n", main="jointype=round", axes=FALSE, xlab="", ylab="")
polygon(A[[1]], col="grey")
C <- polyoffset(A, 1, jointype="round")
polygon(C[[1]], lwd=3, border="blue")
plot(c(0,10),c(0,10), type="n", main="jointype=miter", axes=FALSE, xlab="", ylab="")
polygon(A[[1]], col="grey")
C <- polyoffset(A, 1, jointype="miter")
polygon(C[[1]], lwd=3, border="blue")
```
Index

*Topic **math**
  polyclip, 2
  polylineoffset, 4
  polyoffset, 6
*Topic **spatial**
  polyclip, 2
  polylineoffset, 4
  polyoffset, 6

polyclip, 2
polylineoffset, 4
polyoffset, 6