Package 'polyclip'

October 20, 2022

Version 1.10-4

Date 2022-10-20

Title Polygon Clipping

Maintainer Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

Depends R (>= 3.0.0)

Description
R port of Angus Johnson's open source library 'Clipper'. Performs polygon clipping operations (intersection, union, set minus, set difference) for polygonal regions of arbitrary complexity, including holes. Computes offset polygons (spatial buffer zones, morphological dilations, Minkowski dilations) for polygonal regions and polygonal lines. Computes Minkowski Sum of general polygons. There is a function for removing self-intersections from polygon data.

License BSL

URL http://www.angusj.com/delphi/clipper.php,
https://sourceforge.net/projects/polyclipping,
https://github.com/baddstats/polyclip

BugReports https://github.com/baddstats/polyclip/issues

ByteCompile true

Note built from Clipper C++ version 6.4.0

NeedsCompilation yes

Author Angus Johnson [aut] (C++ original,
http://www.angusj.com/delphi/clipper.php),
Adrian Baddeley [aut, trl, cre],
Kurt Hornik [ctb],
Brian D. Ripley [ctb],
Elliott Sales de Andrade [ctb],
Paul Murrell [ctb]

Repository CRAN

Date/Publication 2022-10-20 17:12:40 UTC

R topics documented:

`pointinpolygon` *Test Whether a Point Lies Inside a Polygon*

Description

Test whether each point lies inside a specified polygon.

Usage

```
pointinpolygon(P, A, eps, x0, y0)
```

Arguments

`P`	Spatial coordinates of the points to be tested. A list of two vectors named `x` and `y`.
`A`	A single polygon, specified as a list of two vectors named `x` and `y`.
`eps`	Spatial resolution for coordinates.
`x0,y0`	Spatial origin for coordinates.

Details

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

The argument `A` represents a closed polygon. It should be a list containing two components `x` and `y` giving the coordinates of the vertices. The last vertex should not repeat the first vertex.

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

An integer vector with one entry for each point in `P`. The result is 0 if the point lies outside `A`, 1 if the point lies inside `A`, and -1 if it lies on the boundary.

Author(s)

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

References

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

Vatti, B. (1992) A generic solution to polygon clipping. *Communications of the ACM* **35** (7) 56–63. https://dl.acm.org/doi/10.1145/129902.129906

Agoston, M.K. (2005) *Computer graphics and geometric modeling: implementation and algorithms.* Springer-Verlag. http://books.google.com/books?q=vatti+clipping+agoston

See Also

polyclip.

Examples

```
A <- list(x=1:10, y=c(1:5,5:1))
P <- list(x=4, y=2)
pointinpolygon(P, A)
```

---

polyclip *Polygon Clipping*

Description

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

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"),
        closed = TRUE)
```

Arguments

A,B	Data specifying polygons. See Details.
op	Set operation to be performed to combine A and B. One of the character strings "intersection", "union", "minus" or "xor" (partially matched).
...	Ignored.
eps	Spatial resolution for coordinates. A single positive numeric value.
x0,y0	Spatial origin for coordinates. Numeric values.
fillA,fillB	Polygon-filling rules for A and B. Each argument is one of the character strings "evenodd", "nonzero", "positive" or "negative" (partially matched).
closed	Logical value specifying whether A is a closed polygon (closed=TRUE, the default) or an open polyline (closed=FALSE).

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 `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 `A` and `B`.

Author(s)

Angus Johnson. Ported to R by Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>. Additional modification by Paul Murrell.

References

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

Vatti, B. (1992) A generic solution to polygon clipping. *Communications of the ACM* **35** (7) 56–63. https://dl.acm.org/doi/10.1145/129902.129906

Agoston, M.K. (2005) *Computer graphics and geometric modeling: implementation and algorithms.* Springer-Verlag. http://books.google.com/books?q=vatti+clipping+agoston

Chen, X. and McMains, S. (2005) Polygon Offsetting by Computing Winding Numbers. Paper no. DETC2005-85513 in *Proceedings of IDETC/CIE 2005* (ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference), pp. 565–575 https://mcmains.me.berkeley.edu/pubs/DAC05OffsetPolygon.pdf

See Also

polysimplify, polyoffset, polylineoffset, polyminkowski

Examples

```
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="", main="intersection of closed polygons")
invisible(lapply(A, polygon))
invisible(lapply(B, polygon))
C <- polyclip(A, B)
polygon(C[[1]], lwd=3, col=3)

plot(c(0,10),c(0,10), type="n", axes=FALSE,
  xlab="", ylab="", main="intersection of open polyline and closed polygon")
invisible(lapply(A, polygon))
invisible(lapply(B, polygon))
E <- polyclip(A, B, closed=FALSE)
invisible(lapply(E, lines, col="purple", lwd=5))
```

---

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 A.

Author(s)

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

References

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

Vatti, B. (1992) A generic solution to polygon clipping. *Communications of the ACM* **35** (7) 56–63. https://dl.acm.org/doi/10.1145/129902.129906

Agoston, M.K. (2005) *Computer graphics and geometric modeling: implementation and algorithms.* Springer-Verlag. http://books.google.com/books?q=vatti+clipping+agoston

Chen, X. and McMains, S. (2005) Polygon Offsetting by Computing Winding Numbers. Paper no. DETC2005-85513 in *Proceedings of IDETC/CIE 2005* (ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference), pp. 565–575 https://mcmains.me.berkeley.edu/pubs/DAC05OffsetPolygon.pdf

See Also

polyoffset, polysimplify, polyclip, polyminkowski

Examples

```
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, 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")
```

---

polyminkowski *Minkowski Sum of Polygon with Path*

Description

Compute the Minkowski Sum of a polygon with a path or paths.

Usage

```
polyminkowski(A, B, ..., eps, x0, y0,
              closed=TRUE)
```

Arguments

A	Data specifying a polygon or polygons. See Details.
B	Data specifying a path or paths. See Details.
...	Ignored.
eps	Spatial resolution for coordinates.
x0,y0	Spatial origin for coordinates.
closed	Logical value indicating whether the resulting path should be interpreted as closed (last vertex joined to first vertex and interior filled in).

Details

This is an interface to the function MinkowskiSum in Angus Johnson's C++ library **Clipper**.

It computes the Minkowski Sum of the polygon A (including its interior) with the path or paths B (*excluding* their interior).

The argument A should be 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.

The argument B should be either

- a list containing two components `x` and `y` giving the coordinates of the vertices of a single polygon or path. 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 or paths.

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 `A`.

Author(s)

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

References

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

See Also

polyclip, polyoffset, polylineoffset, polysimplify

Examples

```
A <- list(list(x=c(-3,3,3,-3),y=c(-3,-3,3,3)))
B <- list(list(x=c(-1,1,1,-1),y=c(-1,-1,1,1)))
C <- polyminkowski(A, B)
opa <- par(mfrow=c(1,3))
rr <- c(-4, 4)
plot(rr, rr, type="n", axes=FALSE, xlab="", ylab="", main="A")
polygon(A[[1]], col="blue")
plot(rr,rr, type="n", axes=FALSE, xlab="", ylab="", main="B")
polygon(B[[1]], border="red", lwd=4)
plot(rr,rr, type="n", axes=FALSE, xlab="", ylab="", main="A+B")
polygon(C[[1]], col="green")
polygon(C[[2]], col="white")
par(opa)
```

`polyoffset` *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

```
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.
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 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 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@curtin.edu.au>.

References

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

Vatti, B. (1992) A generic solution to polygon clipping. *Communications of the ACM* **35** (7) 56–63. https://dl.acm.org/doi/10.1145/129902.129906

Agoston, M.K. (2005) *Computer graphics and geometric modeling: implementation and algorithms.* Springer-Verlag. http://books.google.com/books?q=vatti+clipping+agoston

Chen, X. and McMains, S. (2005) Polygon Offsetting by Computing Winding Numbers. Paper no. DETC2005-85513 in *Proceedings of IDETC/CIE 2005* (ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference), pp. 565–575 https://mcmains.me.berkeley.edu/pubs/DAC05OffsetPolygon.pdf

See Also

`polylineoffset`, `polyclip`, `polysimplify`, `polyminkowski`

Examples

```
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")
```

polysimplify *Remove Self-Intersections from a Polygon*

Description

This function attempts to remove self-intersections and duplicated vertices from the given polygon.

Usage

```
polysimplify(A, ..., eps, x0, y0,
             filltype = c("evenodd", "nonzero", "positive", "negative"))
```

Arguments

A	Data specifying a polygon or polygons. See Details.
...	Ignored.
eps	Spatial resolution for coordinates.
x0,y0	Spatial origin for coordinates.
filltype	Polygon-filling rule. See Details.

Details

This is an interface to the function SimplifyPolygons in Angus Johnson's C++ library **Clipper**. It tries to remove self-intersections from the supplied polygon, by performing a boolean union operation using the nominated filltype. The result may be one or several polygons.

The argument A should be 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.

The argument filltype determines the polygon fill type.

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 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 A.

Author(s)

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

References

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

See Also

polyclip, polyoffset, polylineoffset, polyminkowski

Examples

```
theta <- 2 * pi * (0:5) * 2/5

A <- list(list(x=sin(theta), y=cos(theta)))
B <- polysimplify(A, filltype="nonzero")

opa <- par(mfrow=c(1,2))
plot(c(-1,1),c(-1,1), type="n", axes=FALSE, xlab="", ylab="")
with(A[[1]], segments(x[-6], y[-6], x[-1], y[-1], col="red"))
points(A[[1]], col="red")

with(A[[1]], text(x[1:5], y[1:5], labels=1:5, cex=2))
plot(c(-1,1),c(-1,1), type="n", axes=FALSE, xlab="", ylab="")
polygon(B[[1]], lwd=3, col="green")
with(B[[1]], text(x,y,labels=seq_along(x), cex=2))
par(opa)
```

Index