Package ‘RTriangle’

May 7, 2026

Copyright 1993, 1995, 1997, 1998, 2002, 2005 Jonathan Richard Shewchuk; 2011-2025 David Sterratt

License CC BY-NC-SA 4.0

Title Triangle - A 2D Quality Mesh Generator and Delaunay Triangulator

Description This is a port of Jonathan Shewchuk's Triangle library to R. From his description: ``Triangle generates exact Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, Voronoi diagrams, and high-quality triangular meshes. The latter can be generated with no small or large angles, and are thus suitable for finite element analysis."

Version 1.6-0.15

URL https://github.com/davidcsterratt/RTriangle, http://www.cs.cmu.edu/~quake/triangle.html

Date 2025-04-13

Depends R (>= 3.5.0)

RoxygenNote 7.2.3

Suggests testthat, geometry

BugReports https://github.com/davidcsterratt/RTriangle/issues

Encoding UTF-8

Language en-GB

NeedsCompilation yes

Author Jonathan Richard Shewchuk [ctb, cph],
David C. Sterratt [cph, aut, cre],
Elias Pipping [ctb],
Michael Sumner [ctb]

Maintainer David C. Sterratt <david.c.sterratt@ed.ac.uk>

Repository CRAN

Date/Publication 2025-04-14 08:50:02 UTC

Contents

RTriangle-package 2
plot.pslg 3
plot.triangulation 4
pslg 4
read.pslg 5
triangulate 6

Index **9**

`RTriangle-package` *Generate 2D Quality meshes and constrained Delaunay triangulations*

Description

Generate 2D Quality meshes and constrained Delaunay triangulations

Details

This package is a wrapper of Jonathan Richard Shewchuk's Triangle package. `triangulate` triangulates a *Planar Straight Line Graph* (PSLG), a collection of vertices and segments created with `pslg`. A mesh in the can be created within an arbitrary closed outline and the maximum area and minimum angle of the triangles in the mesh can be specified.

Author(s)

David C. Sterratt <david.c.sterratt@ed.ac.uk>

References

- http://www.cs.cmu.edu/~quake/triangle.html
- Jonathan Richard Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, in "Applied Computational Geometry: Towards Geometric Engineering" (Ming C. Lin and Dinesh Manocha, editors), volume 1148 of Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996. (From the First ACM Workshop on Applied Computational Geometry.)
- Jonathan Richard Shewchuk, *Delaunay Refinement Algorithms for Triangular Mesh Generation*, Computational Geometry: Theory and Applications 22(1-3):21-74, May 2002.

See Also

`triangulate`

Examples

```
## Create an object with a concavity
p <- pslg(P=rbind(c(0, 0), c(0, 1), c(0.5, 0.5), c(1, 1), c(1, 0)),
          S=rbind(c(1, 2), c(2, 3), c(3, 4), c(4, 5), c(5, 1)))
## Plot it
plot(p)
## Triangulate it
tp <- triangulate(p)
plot(tp)
## Triangulate it subject to minimum area constraint
tp <- triangulate(p, a=0.01)
plot(tp)
## Load a data set containing a hole
A <- read.pslg(file.path(system.file(package = "RTriangle"), "extdata", "A.poly"))
plot(A)
## Triangulate the PSLG
tA <- triangulate(A)
plot(tA)
## Triangulate the PSLG with triangles in which no angle
## is smaller than 20 degrees
tA <- triangulate(A, q=20)
plot(tA)
## Triangulate the PSLG with triangles in which no triangle has
## area greater than 0.001
tA <- triangulate(A, a=0.001)
plot(tA)
```

`plot.pslg` *Plot pslg object*

Description

Plot `pslg` object

Usage

```
## S3 method for class 'pslg'
plot(x, ...)
```

Arguments

`x`	`pslg` object
`...`	Arguments to be passed to methods.

Author(s)

David Sterratt

plot.triangulation *Plot a triangulation object produced with triangulate*

Description

Plots a triangulation object produced with `triangulate`

Usage

```
## S3 method for class 'triangulation'
plot(x, ...)
```

Arguments

`x`	Triangulation object produced with `triangulate`.
`...`	Arguments to be passed to methods.

Author(s)

David Sterratt

pslg *Create a Planar Straight Line Graph object*

Description

A Planar Straight Line Graph (PSLG) is a collection of vertices and segments. Segments are edges whose endpoints are vertices in the PSLG, and whose presence in any mesh generated from the PSLG is enforced.

Usage

```
pslg(P, PB = NA, PA = NA, S = NA, SB = NA, H = NA)
```

Arguments

`P`	A 2-column matrix of x-y co-ordinates of vertices. There is one row per vertex.
`PB`	Vector of *boundary markers* of vertices. For each vertex this is 1 if the point should be on a boundary of any mesh generated from the PSLG and 0 otherwise. There should be as many elements in `VB` as there are vertices in `V`.
`PA`	Matrix of *attributes* which are typically floating-point values of physical quantities (such as mass or conductivity) associated with the nodes of a finite element mesh. When triangulating using `triangulate` these are copied unchanged to existing points in the output mesh and each new Steiner point added to the mesh will have quantities assigned to it by linear interpolation.

`S` A 2-column matrix of *segments* in which each row is a *segment*. Segments are edges whose endpoints are vertices in the PSLG, and whose presence in any mesh generated from the PSLG is enforced. Each segment refers to the indices in `V` of the endpoints of the segment. By default the segments are not specified (`NA`), in which case the convex hull of the vertices are taken to be the segments. Any vertices outside the region enclosed by the segments are eaten away by the triangulation algorithm. If the segments do not enclose a region the whole triangulation may be eaten away.

`SB` Vector of boundary markers of segments. For each segment this is 1 if the segment should be on a boundary of any mesh generated from the PSLG and 0 otherwise. There should be as many elements in `SB` as there are segments in `S`.

`H` 2-column matrix of *holes*, with one hole per row.Holes are specified by identifying a point inside each hole. After the triangulation is formed, Triangle creates holes by eating triangles, spreading out from each hole point until its progress is blocked by PSLG segments; you must be careful to enclose each hole in segments, or your whole triangulation might be eaten away. If the two triangles abutting a segment are eaten, the segment itself is also eaten. Do not place a hole directly on a segment; if you do, Triangle will choose one side of the segment arbitrarily.

Value

An object containing the input of type `pslg` that contains the information supplied in the inputs. This function does some sanity checking of its inputs.

Author(s)

David Sterratt

`read.pslg` *Read a Planar Straight Line Graph from file*

Description

Read a Planar Straight Line Graph from a `.poly` file, as used in Shewchuk's Triangle library (http://www.cs.cmu.edu/~quake/triangle.poly.html).

Usage

```
read.pslg(file)
```

Arguments

`file` File name of `.poly` file to read.

Value

`pslg` object. See `pslg`.

Author(s)

David Sterratt

`triangulate` *Triangulate a Planar Straight Line Graph*

Description

Triangulate a planar straight line graph using the Triangle library (http://www.cs.cmu.edu/~quake/triangle.html). The triangulation is a constrained conforming Delaunay triangulation in which additional vertices, called Steiner points, can be inserted into segments to improved the quality of the triangulation. To prevent the insertion of Steiner points on boundary segments, specify `Y=TRUE`. If the maximum triangle area `a` is specified, the area of each triangle is not allowed to exceed this value. If the the minimum angle `q` is specified, no triangle angle is allowed to be below this value.

Usage

```
triangulate(
  p,
  a = NULL,
  q = NULL,
  Y = FALSE,
  j = FALSE,
  D = FALSE,
  S = 10000,
  V = 0,
  Q = TRUE
)
```

Arguments

`p`	Planar straight line graph object; see `pslg`.
`a`	Maximum triangle area. If specified, triangles cannot be larger than this area.
`q`	Minimum triangle angle in degrees.
`Y`	If `TRUE` prohibits the insertion of Steiner points on the mesh boundary.
`j`	If `TRUE` jettisons vertices that are not part of the final triangulation from the output.
`D`	If `TRUE` produce a conforming Delaunay triangulation. This ensures that all the triangles in the mesh are truly Delaunay, and not merely constrained Delaunay. This option invokes Ruppert's original algorithm, which splits every subsegment whose diametral circle is encroached. It usually increases the number of vertices and triangles.
`S`	Specifies the maximum number of added Steiner points. If set to `Inf`, there is no limit on the number of Steine points added - but this can lead to huge amounts of memory being allocated.

V	Verbosity level. Specify higher values for more detailed information about what the Triangle library is doing.
Q	If TRUE suppresses all explanation of what the Triangle library is doing, unless an error occurs.

Value

A object with class triangulation. This contains the information in the same format as the PSLG, p, with an updated list of points P and point attributes PA, along with the following variables:

T	Triangulation specified as 3 column matrix in which each row contains indices in P of vertices.
E	Set of edges in the triangulation.
EB	Boundary markers of edges. For each edge this is 1 if the point is on a boundary of the triangulation and 0 otherwise.
VP	The points of the Voronoi tessalation as a 2-column matrix
VE	Set of edges of the Voronoi tessalation. An index of -1 indicates an infinite ray.
VN	Directions of infinite rays of Voroni tessalation as a 2-column matrix with the same number of rows as VP.
VA	Matrix of *attributes* associated with the polygons of the Voronoi tessalation.

Author(s)

David Sterratt

Examples

```
## Create an object with a concavity
p <- pslg(P=rbind(c(0, 0), c(0, 1), c(0.5, 0.5), c(1, 1), c(1, 0)),
          S=rbind(c(1, 2), c(2, 3), c(3, 4), c(4, 5), c(5, 1)))
## Plot it
plot(p)
## Triangulate it
tp <- triangulate(p)
plot(tp)
## Triangulate it subject to minimum area constraint
tp <- triangulate(p, a=0.01)
plot(tp)
## Load a data set containing a hole
A <- read.pslg(file.path(system.file(package = "RTriangle"), "extdata", "A.poly"))
plot(A)
## Produce a constrained Delaunay triangulation of the PSLG
tA <- triangulate(A, Y=TRUE)
plot(tA)
## Produce a conforming Delaunay triangulation of the PSLG
tA <- triangulate(A, D=TRUE)
plot(tA)
## Triangulate the PSLG with triangles in which no angle
## is smaller than 20 degrees
```

```
tA <- triangulate(A, q=20)
plot(tA)
## Triangulate the PSLG with triangles in which no triangle has
## area greater than 0.001
tA <- triangulate(A, a=0.001)
plot(tA)
```

Index

∗ **package**
    RTriangle-package, 2

plot.pslg, 3
plot.triangulation, 4
pslg, *2, 3*, 4, *5, 6*

read.pslg, 5
RTriangle *(*RTriangle-package*)*, 2
RTriangle-package, 2

triangulate, *2*, *4*, 6