Package 'alphahull'

June 16, 2022

Version 2.5

Date 2022-06-16

Title Generalization of the Convex Hull of a Sample of Points in the Plane

Author Beatriz Pateiro-Lopez [aut, cre], Alberto Rodriguez-Casal, [aut].

Maintainer Beatriz Pateiro-Lopez <beatriz.pateiro@usc.es>

Imports ggplot2, interp, R.utils, sgeostat, spatstat.geom, spatstat.random, splancs

Description Computation of the alpha-shape and alpha-convex hull of a given sample of points in the plane. The concepts of alpha-shape and alpha-convex hull generalize the definition of the convex hull of a finite set of points. The programming is based on the duality between the Voronoi diagram and Delaunay triangulation. The package also includes a function that returns the Delaunay mesh of a given sample of points and its dual Voronoi diagram in one single object.

License GPL-2 | GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-16 17:00:02 UTC

R topics documented:

---

alphahull-package   *Generalization of the Convex Hull of a Sample of Points in the Plane*

Description

Computation of the α-shape and α-convex hull of a given sample of points in the plane. The concepts of α-shape and α-convex hull generalize the definition of the convex hull of a finite set of points. The programming is based on the duality between the Voronoi diagram and Delaunay triangulation. The package also includes a function that returns the Delaunay mesh of a given sample of points and its dual Voronoi diagram in one single object.

Details

Package:	alphahull
Type:	Package
Version:	2.5
Date:	2022-06-16
License:	R functions: GPL-2	GPL-3

Author(s)

Beatriz Pateiro-Lopez, Alberto Rodriguez-Casal.

Maintainer: Beatriz Pateiro-Lopez <beatriz.pateiro@usc.es>

---

ahull   *alpha-convex hull calculation*

Description

This function calculates the α-convex hull of a given sample of points in the plane for $\alpha > 0$.

Usage

```
ahull(x, y = NULL, alpha)
```

Arguments

x, y	The x and y arguments provide the x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.
alpha	Value of α.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the α-convex hull. Any reasonable way of defining the coordinates is acceptable, see xy.coords.

The α-convex hull is defined for any finite number of points. However, since the algorithm is based on the Delaunay triangulation, at least three non-collinear points are required.

If y is NULL and x is an object of class "delvor", then the α-convex hull is computed with no need to invoke again the function delvor (it reduces the computational cost).

The complement of the α-convex hull can be written as the union of $O(n)$ open balls and halfplanes, see complement. The boundary of the α-convex hull is formed by arcs of open balls of radius α (besides possible isolated sample points). The arcs are determined by the intersections of some of the balls that define the complement of the α-convex hull. The extremes of an arc are given by $c + rA_\theta v$ and $c + rA_{-\theta} v$ where c and r represent the center and radius of the arc, repectively, and $A_\theta v$ represents the clockwise rotation of angle θ of the unitary vector v. Joining the end points of adjacent arcs we can define polygons that help us to determine the area of the estimator , see areaahull.

Value

A list with the following components:

arcs	For each arc in the boundary of the α-convex hull, the columns of the matrix arcs store the center c and radius r of the arc, the unitary vector v, the angle θ that define the arc and the indices of the end points, see Details. The coordinates of the end points of the arcs are stored in xahull. For isolated points in the boundary of the α-convex hull, columns 3 to 6 of the matrix arcs are equal to zero.
xahull	A 2-column matrix with the coordinates of the original set of points besides possible new end points of the arcs in the boundary of the α-convex hull.
length	Length of the boundary of the α-convex hull, see lengthahull.
complement	Output matrix from complement.
alpha	Value of α.
ashape.obj	Object of class "ashape" returned by the function ashape.

References

Edelsbrunner, H., Kirkpatrick, D.G. and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4), pp.551-559.

Rodriguez-Casal, R. (2007). Set estimation under convexity type assumptions. *Annales de l'I.H.P.-Probabilites & Statistiques*, 43, pp.763-774.

Pateiro-Lopez, B. (2008). *Set estimation under convexity type restrictions*. Phd. Thesis. Universidad de Santiago de Compostela. ISBN 978-84-9887-084-8.

See Also

`plot.ahull`.

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(100), nc = 2)
# Value of alpha
alpha <- 0.2
# Alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
plot(ahull.obj)

# Uniform sample of size n=300 in the annulus B(c,0.5)\B(c,0.25),
# with c=(0.5,0.5).
n <- 300
theta<-runif(n,0,2*pi)
r<-sqrt(runif(n,0.25^2,0.5^2))
x<-cbind(0.5+r*cos(theta),0.5+r*sin(theta))
# Value of alpha
alpha <- 0.1
# Alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
# The arcs defining the boundary of the alpha-convex hull are ordered
plot(x)
for (i in 1:dim(ahull.obj$arcs)[1]){
arc(ahull.obj$arcs[i,1:2],ahull.obj$arcs[i,3],ahull.obj$arcs[i,4:5],
ahull.obj$arcs[i,6],col=2)
Sys.sleep(0.5)
}

# Random sample  from a uniform distribution on a Koch snowflake
# with initial side length 1 and 3 iterations
x <- rkoch(2000, side = 1, niter = 3)
# Value of alpha
alpha <- 0.05
# Alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
plot(ahull.obj)

## End(Not run)
```

---

ahull_track	*alpha-convex hull calculation of tracking data*

Description

This function approximates the α-convex hull of tracking data and returns a list of geom_path objects of the boundary.

Usage

```
ahull_track(x, y = NULL, alpha, nps = 10000, sc = 100)
```

Arguments

x, y	The x and y arguments provide the x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.
alpha	Value of α.
nps	Number of points to generate in each segment connecting two locations, see Details
sc	Scale factor.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the α-convex hull. Any reasonable way of defining the coordinates is acceptable, see `xy.coords`.

Increase nps if the trajectory is not contained in the computed estimator.

Value

A list of geom_path objects defining the boundary of the α-convex

References

Cholaquidis, A., Fraiman, R., Lugosi, G. and Pateiro-Lopez, B. (2014) Set estimation from reflected Brownian motion. *arXiv:1411.0433*.

Wikelski, M., and Kays, R. (2014). Movebank: archive, analysis and sharing of animal movement data. World Wide Web electronic publication.

Examples

```
## Not run:
library(move)
library(ggmap)
# Data from Movebank
# Study Name: Dunn Ranch Bison Tracking Project
# Principal Investigator: Stephen Blake, Randy Arndt, Doug Ladd
# Max Planck Institute for Ornithology Radolfzell Germany
```

```
study <- "Dunn Ranch Bison Tracking Project"
cainfo <- system.file("CurlSSL", "cacert.pem", package = "RCurl")
options(RCurlOptions = list(verbose = FALSE, capath = cainfo, ssl.verifypeer = FALSE))
# Login to movebank (first create the login object)
curl <- movebankLogin(username = "xxx", password = "zzz")
# Downloads study stored in Movebank
track <- getMovebankData(study = study, login = curl)
dat <- track@data[track@data[, "deployment_id"] == 13848432,]
# Map of animal locations
bbox <- ggmap::make_bbox(dat[,"location_long"], dat[,"location_lat"], f = 0.3)
map_loc <- get_map(location = bbox, source = "google", maptype = 'satellite')
map <- ggmap(map_loc, extent = 'panel', maprange=FALSE)
p <- map + geom_path(data = dat, aes(x = location_long, y = location_lat), col=2, size=0.3)
p
ah_gp <- ahull_track(x = dat[, c("location_long", "location_lat")], alpha = 0.005)
p + ah_gp

## End(Not run)
```

---

anglesArc   *Angles of the extremes of an arc*

Description

Given a vector v and an angle θ, anglesArc returns the angles that $A_\theta v$ and $A_{-\theta} v$ form with the axis *OX*, where $A_\theta v$ represents the clockwise rotation of angle θ of the vector v.

Usage

```
anglesArc(v, theta)
```

Arguments

v	Vector v in the plane.
theta	Angle θ (in radians).

Details

The angle that forms the vector v with the axis *OX* takes its value in $[0, 2\pi)$.

Value

angs	Numeric vector with two components.

Examples

```
## Not run:
# Let v=c(0,1) and theta=pi/4
# Consider the arc such that v is the internal angle bisector that
# divides the angle 2*theta into two equal angles
# The angles that the arc forms with the OX axis are pi/4 and 3*pi/4
v <- c(0,1)
theta <- pi/4
anglesArc(v,theta)

## End(Not run)
```

---

arc   *Add an arc to a plot*

Description

This function adds the arc of $B(c, r)$ between the angles that $A_\theta v$ and $A_{-\theta} v$ form with the axis *OX*, where $A_\theta v$ represents the clockwise rotation of angle θ of the vector v.

Usage

```
arc(c, r, v, theta, ...)
```

Arguments

c	Center c of the arc.
r	Radius r of the arc.
v	Vector v in the plane.
theta	Angle θ (in radians).
...	Arguments to be passed to methods, such as graphical parameters (see par).

See Also

plot.ahull.

Examples

```
## Not run:
# Plot of the circumference of radius 1
theta <- seq(0, 2*pi, length = 100)
r <- 1
plot(r*cos(theta), r*sin(theta), type = "l")
# Add in red the arc between pi/4 and 3*pi/4
arc(c(0,0), 1, c(0,1), pi/4, col = 2, lwd = 2)

## End(Not run)
```

---

areaahull   *Area of the alpha-convex hull*

Description

This function calculates the area of the α-convex hull of a sample of points.

Usage

```
areaahull(x, timeout = 5)
```

Arguments

x	Object of class "ahull".
timeout	A numeric specifying the maximum number of seconds the expression is allowed to run before being interrupted by the timeout.

Value

area	Area of the α-convex hull. If the area cannot be computed, the output will be NA with a warning.

See Also

ahull.

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(500), nc = 2)
# Value of alpha
alpha <- 1
# alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
# Area of the alpha-convex hull
areaahull(ahull.obj)

## End(Not run)
```

| ashape | *alpha-shape calculation* |

Description

This function calculates the α-shape of a given sample for $\alpha > 0$.

Usage

```
ashape(x, y = NULL, alpha)
```

Arguments

x, y   The x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.

alpha   Value of α.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the α-shape, see `xy.coords`.

The α-shape is defined for any finite number of points. However, since the algorithm is based on the Delaunay triangulation, at least three non-collinear points are required.

If y is NULL and x is an object of class "delvor", then the α-shape is computed without invoking again the function `delvor` (it reduces the computational cost).

The function `ashape` returns (among other values) the matrix edges. The structure of edges is that of matrix mesh returned by the function `delvor`. Note that the α-shape is a subgraph of the Delaunay triangulation and, therefore, edges is a submatrix of mesh.

Value

A list with the following components:

edges   A *n.seg*-row matrix with the coordinates and indexes of the edges of the Delaunay triangulation that form the α-shape. The number of rows *n.seg* coincides with the number of segments of the α-shape. The matrix also includes information of the Voronoi extremes corresponding to each segment.

length   Length of the α-shape.

alpha   Value of α.

alpha.extremes   Vector with the indexes of the sample points that are α-extremes. See Edelsbrunner *et al.* (1983).

delvor.obj   Object of class "delvor" returned by the `delvor` function.

x   A 2-column matrix with the coordinates of the set of points.

References

Edelsbrunner, H., Kirkpatrick, D.G. and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4), pp.551-559.

See Also

plot.ashape, delvor.

Examples

```
## Not run:
# Uniform sample of size n=300 in the annulus B(c,0.5)\B(c,0.25),
# with c=(0.5,0.5).
n <- 300
theta<-runif(n,0,2*pi)
r<-sqrt(runif(n,0.25^2,0.5^2))
x<-cbind(0.5+r*cos(theta),0.5+r*sin(theta))
# Value of alpha
alpha <- 0.1
# alpha-shape
ashape.obj <- ashape(x, alpha = alpha)
# If we change the value of alpha there is no need to compute
# again the Delaunay triangulation and Voronoi Diagram
alpha <- 0.4
ashape.obj.new <- ashape(ashape.obj$delvor.obj, alpha = alpha)

# Random sample  from a uniform distribution on a Koch snowflake
# with initial side length 1 and 3 iterations
x <- rkoch(2000, side = 1, niter = 3)
# Value of alpha
alpha <- 0.05
# alpha-shape
ashape.obj <- ashape(x, alpha = alpha)

## End(Not run)
```

---

complement	*Complement of the alpha-convex hull*

Description

This function calculates the complement of the α-convex hull of a given sample for $\alpha > 0$.

Usage

```
complement(x, y = NULL, alpha)
```

Arguments

`x, y`	The x and y arguments provide the x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.
`alpha`	Value of α.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the α-shape. Any reasonable way of defining the coordinates is acceptable, see `xy.coords`.

If y is NULL and x is an object of class `"delvor"`, then the complement of the α-convex hull is computed with no need to invoke again the function `delvor` (it reduces the computational cost).

The complement of the α-convex hull is calculated as a union of open balls and halfplanes that do not contain any point of the sample. See Edelsbrunnner *et al.* (1983) for a basic description of the algorithm. The construction of the complement is based on the Delaunay triangulation and Voronoi diagram of the sample, provided by the function `delvor`. The function `complement` returns a matrix compl. For each row i, compl[i,] contains the information relative to an open ball or halfplane of the complement. The first three columns are assigned to the characterization of the ball or halfplane i. The information relative to the edge of the Delaunay triangulation that generates the ball or halfplane i is contained in compl[i,4:16]. Thus, if the row i refers to an open ball, compl[i,1:3] contains the center and radius of the ball. Furthermore, compl[i,17:18] and compl[i,19] refer to the unitary vector v and the angle θ that characterize the arc that joins the two sample points that define the ball i. If the row i refers to a halfplane, compl[i,1:3] determines its equation. For the halfplane $y > a + bx$, compl[i,1:3]=(a,b,-1). In the same way, for the halfplane $y < a + bx$, compl[i,1:3]=(a,b,-2), for the halfplane $x > a$, compl[i,1:3]=(a,0,-3) and for the halfplane $x < a$, compl[i,1:3]=(a,0,-4).

Value

`compl`	Output matrix. For each row i, compl[i,] contains the information relative to an open ball or halfplane of the complement of the α-convex hull, see Details.

References

Edelsbrunner, H., Kirkpatrick, D.G. and Seidel, R. (1983) On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4), pp.551-559.

See Also

`delvor`, `ahull`.

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(100), nc = 2)
# Value of alpha
alpha <- 0.2
# Complement of the alpha-convex hull
compl <- complement(x, alpha = alpha)
```

```
## End(Not run)
```

---

delvor   *Delaunay triangulation and Voronoi diagram*

Description

This function returns a matrix with information about the Delaunay triangulation and Voronoi diagram of a given sample.

Usage

```
delvor(x, y = NULL)
```

Arguments

x, y   The x and y arguments provide the x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the Delaunay triangulation and Voronoi diagram . Any reasonable way of defining the coordinates is acceptable, see xy.coords.

The function tri.mesh from package **interp** calculates the Delaunay triangulation of at least three non-collinear points. Using the Delaunay triangulation, the function delvor calculates the correspondig Voronoi diagram. For each edge of the Delaunay triangulation there is a segment in the Voronoi diagram, given by the union of the circumcenters of the two neighbour triangles that share the edge. For those triangles with edges on the convex hull, the corresponding line in the Voronoi diagram is a semi-infinite segment, whose boundless extreme is calculated by the function dummycoor. The function delvor returns the sample, the output object of class "triSht" from the function tri.mesh and a matrix mesh with all the necessary information of the Delaunay triangulation and Voronoi diagram. Thus, for each edge of the Delaunay triangulation the output matrix contains the indexes and coordinates of the sample points that form the edge, the indexes and coordinates of the extremes of the corresponding segment in the Voronoi diagram, and an indicator that takes the value 1 for those extremes of the Voronoi diagram that represent a boundless extreme.

Value

A list with the following components:

mesh   A $n.edges$-row matrix, where $n.edges$ is the total number of different edges of the Delaunay triangulation.

x   A 2-column matrix with the coordinates of the sample points.

tri.obj   Object of class "tri". See tri.mesh in package **interp**.

References

Renka, R. J. (1996). Algorithm 751: TRIPACK: a constrained two-dimensional Delaunay triangulation package, *ACM Trans. Math. Softw.*, 22(1), pp.1-8.

See Also

[plot.delvor](#).

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(20), nc = 2)
# Delaunay triangulation and Voronoi diagram calculation
delvor.obj <- delvor(x)

## End(Not run)
```

---

dummycoor   *Semi-infinite edge of the Voronoi diagram*

Description

This function determines fictitious coordinates for the boundless extreme of a semi-infinite edge of the Voronoi diagram.

Usage

```
dummycoor(tri.obj, l1, l2, m, away)
```

Arguments

tri.obj	Object of class "triSht". See [tri.mesh](#) in package **interp**.
l1	Index of the sample point correponding to one vertex of a triangle of Delaunay that lies on the convex hull, see Details.
l2	Index of the sample point correponding to other vertex of a triangle of Delaunay that lies on the convex hull, see Details.
m	Index of the circumcenter of the triangle of Delaunay with one edge on the convex hull.
away	Constant that determines how far away the fictitious boundless extreme is located.

Details

When a triangle of the Delaunay triangulation has one of its edges (given by the segment that joins the sample points with indexes l1 and l2) on the convex hull, the corresponding segment of the Voronoi diagram is semi-infinite. The finite extreme coincides with the circumcenter of the triangle and the direction of the line is given by the perpendicular bisector of the edge that lies on the convex hull.

Value

dum   Fictitious coordinates of the boundless extreme.

See Also

[delvor](#).

---

dw   *Devroye-Wise estimator*

Description

This function calculates the Devroye-Wise estimator of a given sample of points in the plane for $\epsilon > 0$.

Usage

```
dw(x, y = NULL, eps)
```

Arguments

x, y   The x and y arguments provide the x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.

eps   Value of ϵ.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the Devroye-Wise estimator. Any reasonable way of defining the coordinates is acceptable, see [xy.coords](#).

Value

Given a sample of points in the plane, the estimator is defined as union of balls of radius ϵ with centers in the sample points. For each arc in the boundary of the Devroye-Wise estimator, the columns of the output matrix store the center c and radius r of the arc, the unitary vector v, the angle θ that define the arc and the indices of the end points.

References

Devroye, L. and Wise, G. (1980) Detection of abnormal behaviour via nonparametric estimation of the support. *SIAM J. Appl. Math.* 3, pp. 480-488.

Examples

```
## Not run:
# Uniform sample of size n = 200 in the annulus B(c, 0.5)\B(c, 0.25),
# with c = (0.5, 0.5).
n <- 200
theta <- runif(n, 0, 2*pi)
r <- sqrt(runif(n, 0.25^2, 0.5^2))
x <- cbind(0.5 + r*cos(theta), 0.5 + r*sin(theta))
eps <- 0.05
dw.obj <- dw(x, eps = eps)
plot(x)
for(i in 1:dim(dw.obj)[1]){arc(dw.obj[i, 1:2], eps, dw.obj[i, 4:5], dw.obj[i, 6])}

## End(Not run)
```

---

dw_track   *RBM-sausage calculation of tracking data*

Description

This function approximates the RBM-sausage of tracking data and returns a list of geom_path objects of the boundary.

Usage

```
dw_track(x, y = NULL, eps, nps = 20000, sc = 100)
```

Arguments

x, y	The x and y arguments provide the x and y coordinates of a set of points. Alternatively, a single argument x can be provided, see Details.
eps	Value of ϵ.
nps	Number of points to generate in each segment connecting two locations, see Details.
sc	Scale factor.

Details

An attempt is made to interpret the arguments x and y in a way suitable for computing the RBM-sausage. Any reasonable way of defining the coordinates is acceptable, see `xy.coords`. Given a trajectory in the plane, the estimator is defined as the set of points whose distance to the trajectory is less than or equal to ϵ (this estimator is analogous to the one of Devroye and Wise (1980) for i.i.d. data). Increase nps if the trajectory is not contained in the computed estimator.

Value

A list of geom_path objects defining the boundary of the estimator

References

Cholaquidis, A., Fraiman, R., Lugosi, G. and Pateiro-Lopez, B. (2014) Set estimation from reflected Brownian motion. *arXiv:1411.0433*.

Devroye, L. and Wise, G. (1980) Detection of abnormal behaviour via nonparametric estimation of the support. *SIAM J. Appl. Math.* 3, pp. 480-488.

Wikelski, M., and Kays, R. (2014). Movebank: archive, analysis and sharing of animal movement data. World Wide Web electronic publication.

Examples

```
## Not run:
library(move)
library(ggmap)
# Data from Movebank
# Study Name: Dunn Ranch Bison Tracking Project
# Principal Investigator: Stephen Blake, Randy Arndt, Doug Ladd
# Max Planck Institute for Ornithology Radolfzell Germany
study <- "Dunn Ranch Bison Tracking Project"
cainfo <- system.file("CurlSSL", "cacert.pem", package = "RCurl")
options(RCurlOptions = list(verbose = FALSE, capath = cainfo, ssl.verifypeer = FALSE))
# Login to movebank (first create the login object)
curl <- movebankLogin(username = "xxx", password = "zzz")
# Downloads study stored in Movebank
track <- getMovebankData(study = study, login = curl)
dat <- track@data[track@data[, "deployment_id"] == 13848432,]
# Map of animal locations
bbox <- ggmap::make_bbox(dat[,"location_long"], dat[,"location_lat"], f = 0.3)
map_loc <- get_map(location = bbox, source = "google", maptype = 'satellite')
map <- ggmap(map_loc, extent = 'panel', maprange=FALSE)
p <- map + geom_path(data = dat, aes(x = location_long, y = location_lat), col=2, size=0.3)
p
ah_dw <- dw_track(x = dat[, c("location_long", "location_lat")], eps = 0.001)
p + ah_dw

## End(Not run)
```

---

inahull	*Determines for one or more points whether they belong to the alpha-convex hull*

Description

This function determines for one or more points p whether they belong to the α-convex hull of a sample.

Usage

```
inahull(ahull.obj, p)
```

Arguments

ahull.obj	Object of class "ahull" returned by the funcion ahull.
p	Numeric vector with two components describing a point in the plane or two-column matrix of points.

Details

The complement of the α-convex hull of a sample is calculated by complement. The function inahull checks whether each point in p belongs to any of the open balls or halfplanes that define the complement.

Value

in.ahull	A logical vector specifying whether each point in p belongs to the α-convex hull.

See Also

ahull, complement.

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(100), nc = 2)
# Value of alpha
alpha <- 0.2
# alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
# Check if the point (0.5, 0.5) belongs to the alpha-convex hull
inahull(ahull.obj, p = c(0.5, 0.5))
# Check if the points (0.5, 0.5) and (2, 2) belong to the alpha-convex hull
inahull(ahull.obj, p = rbind(c(0.5, 0.5), c(2, 2)))

## End(Not run)
```

---

inter	*Intersection of two circumferences*

Description

This function calculates the intersection of two circumferences, given their centers and radius $c1, r1$ and $c2, r2$, respectively.

Usage

```
inter(c11, c12, r1, c21, c22, r2)
```

Arguments

c11	*X*-coordinate of the center $c1$.
c12	*Y*-coordinate of the center $c1$.
r1	Radius $r1$.
c21	*X*-coordinate of the center $c2$.
c22	*Y*-coordinate of the center $c2$.
r2	Radius $r2$.

Details

The function inter is internally called by the function [ahull](#).

Value

A list with the following components:

n.cut	Number of intersection points (0,1,2, or Inf).
v1	If there are two intersection points, v1 is the numeric vector whose components are the coordinates of the unitary vector that has its origin in $c1$ and it's perpendicular to the chord that joins the intersection points of the two circumferences. Otherwise, v1=(0,0)
theta1	Angle that forms v1 with the radius that joins the center $c1$ with an intersection point.
v2	If there are two intersection points, v2 is the numeric vector whose components are the coordinates of the unitary vector that has its origin in $c2$ and it's perpendicular to the chord that joins the intersection points of the two circumferences. Otherwise, v2=(0,0)
theta2	Angle that forms v2 with the radius that joins the center $c2$ with an intersection point.

---

koch   *Construct a Kock snowflake curve*

Description

This function uses recursion to construct a Kock snowflake curve.

Usage

```
koch(side = 3, niter = 5)
```

Arguments

side	Side length of the initial equilateral triangle.
niter	Number of iterations in the development of the snowflake curve.

Details

The Koch snowflake is a fractal curve described by the Swedish mathematician Helge von Koch in 1904. It is built by starting with an equilateral triangle, removing the inner third of each side, building another equilateral triangle at the location where the side was removed, and then repeating the process.

Value

vertices   A 2-column matrix with the coordinates of the snowflake vertices.

References

von Koch, H. (1904). Sur une courbe continue sans tangente, obtenue par une construction geometrique elementaire. *Arkiv for Matematik*, 1, pp.681-704.

See Also

rkoch.

Examples

```
## Not run:
# The first four iterations of a Koch snowflake
# with side length of the initial equilateral triangle equal to 3.
vertices <- koch(side = 2, niter = 4)
plot(vertices[, 1], vertices[, 2], type = "l", asp = TRUE,
main = "Koch snowflake", xlab = "", ylab = "", col = 4)
polygon(vertices[, 1], vertices[, 2] , col = 4)

## End(Not run)
```

---

lengthahull   *Length of the boundary of the alpha-convex hull*

Description

This function calculates the length of the boundary of the α-convex hull of a given sample.

Usage

```
lengthahull(ahull.arcs)
```

Arguments

ahull.arcs   Output matrix of arcs returned by ahull.

Details

The function lengthahull is internally called by the function ahull.

Value

length   Length of the boundary of the α-convex hull.

See Also

[ahull](#).

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(100), nc = 2)
# Value of alpha
alpha <- 0.2
# alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
# Length of the alpha-convex hull
ahull.obj$length

## End(Not run)
```

---

plot.ahull   *Plot the alpha-convex hull*

Description

This function returns a plot of the α-convex hull. If desired, it also adds the Delaunay triangulation, Voronoi diagram and α-shape of the sample.

Usage

```
## S3 method for class 'ahull'
plot(x, add = FALSE, do.shape = FALSE,
wlines = c("none", "both", "del", "vor"), wpoints = TRUE,
number = FALSE, col = NULL, xlim = NULL,
ylim = NULL, lwd = NULL, ...)
```

Arguments

x   Object of class "ahull".

add   Logical, if TRUE add to a current plot.

do.shape   Logical, indicates if the α-shape should also be plotted.

wlines   "Which lines?". I.e. should the Delaunay triangulation be plotted (wlines='del'), should the Voronoi diagram be plotted (wlines='vor'), should both be plotted (wlines='both'), or none (wlines='none', the default)?

wpoints   Logical, indicates if sample points should be plotted.

number	Logical, defaulting to FALSE; if TRUE then the points plotted will be labelled with their index numbers.
col	The colour numbers for plotting the α-convex hull, α-shape, data points, Delaunay triangulation, Voronoi diagram, and the point numbers, in that order; defaults to c(1,1,1,1,1,1). If fewer than six numbers are given, they are recycled. (If more than six numbers are given, the redundant ones are ignored.)
xlim	The limits on the x-axis.
ylim	The limits on the y-axis.
lwd	The line widths for plotting the tesselations, the α-shape, and the α-convex hull, in that order; defaults to c(1,1,2).
...	Arguments to be passed to methods, such as graphical parameters (see [par](#)).

See Also

[ahull](#), [ashape](#).

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(100), nc = 2)
# Value of alpha
alpha <- 0.2
# alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
# Plot including the alpha-convex hull in pink, alpha-shape in blue,
# sample points in black, voronoi diagram in green
# and Delaunay triangulation in red
plot(ahull.obj, do.shape = TRUE, wlines = "both", col = c(6, 4, 1, 2, 3))

# Random sample  from a uniform distribution on a Koch snowflake
# with initial side length 1 and 3 iterations
x <- rkoch(2000, side = 1, niter = 3)
# Value of alpha
alpha <- 0.05
# Alpha-convex hull
ahull.obj <- ahull(x, alpha = alpha)
plot(ahull.obj)

## End(Not run)
```

plot.ashape	*Plot the alpha-shape*

Description

This function returns a plot of the α-shape. If desired, it also adds the Delaunay triangulation and Voronoi diagram of the sample.

Usage

```
## S3 method for class 'ashape'
plot(x, add = FALSE, wlines = c("none", "both", "del", "vor"),
wpoints = TRUE, number = FALSE, col = NULL,
xlim = NULL, ylim = NULL, lwd = NULL, ...)
```

Arguments

x	Object of class "ashape".
add	Logical, if TRUE add to a current plot.
wlines	"Which lines?". I.e. should the Delaunay triangulation be plotted (wlines='del'), should the Voronoi diagram be plotted (wlines='vor'), should both be plotted (wlines='both'), or none (wlines='none', the default)?
wpoints	Logical, indicates if sample points should be plotted.
number	Logical, defaulting to FALSE; if TRUE then the points plotted will be labelled with their index numbers.
col	The colour numbers for plotting the α-shape, data points, Delaunay triangulation, Voronoi diagram, and the point numbers, in that order; defaults to c(1,1,1,1,1). If fewer than five numbers are given, they are recycled. (If more than five numbers are given, the redundant ones are ignored.)
xlim	The limits on the x-axis.
ylim	The limits on the y-axis.
lwd	The line widths for plotting the tesselations and the α-shape; defaults to c(1,2).
...	Arguments to be passed to methods, such as graphical parameters (see [par](par)).

See Also

ashape.

Examples

```
## Not run:
# Uniform sample of size n=300 in the annulus B(c, 0.5)\B(c, 0.25)
# with c=(0.5, 0.5).
n <- 300
theta<-runif(n,0,2*pi)
r<-sqrt(runif(n,0.25^2,0.5^2))
x<-cbind(0.5+r*cos(theta),0.5+r*sin(theta))
# Value of alpha
alpha <- 0.1
# alpha-shape
ashape.obj <- ashape(x, alpha = alpha)
# Plot alpha-shape in blue, sample points in black,
# and Delaunay triangulation in red
plot(ashape.obj, wlines= "del", col = c(4, 1, 2))

# Random sample  from a uniform distribution on a Koch snowflake
```

```
# with initial side length 1 and 3 iterations
x <- rkoch(2000, side = 1, niter = 3)
# Value of alpha
alpha <- 0.05
# alpha-shape
ashape.obj <- ashape(x, alpha = alpha)
# Plot alpha-shape in blue
plot(ashape.obj, col = c(4, 1))

## End(Not run)
```

plot.delvor   *Plot the Voronoi diagram and Delaunay traingulation*

Description

This function returns a plot of the Voronoi diagram and Delaunay traingulation.

Usage

```
## S3 method for class 'delvor'
plot(x, add = FALSE, wlines = c("both", "del", "vor"),
wpoints = TRUE, number = FALSE, col = NULL,
xlim = NULL, ylim = NULL, ...)
```

Arguments

x	An object of class "delvor" as constructed by the function delvor.
add	Logical, if TRUE add to a current plot.
wlines	"Which lines?". I.e. should the Delaunay triangulation be plotted (wlines='del'), should the Voronoi diagram be plotted (wlines='vor'), or should both be plotted (wlines='both', the default)?
wpoints	Logical, indicates if sample points should be plotted.
number	Logical, defaulting to FALSE; if TRUE then the points plotted will be labelled with their index numbers.
col	The colour numbers for plotting the data points, Delaunay triangulation, Voronoi diagram, and the point numbers, in that order; defaults to c(1,1,1,1). If fewer than four numbers are given, they are recycled. (If more than four numbers are given, the redundant ones are ignored.)
xlim	The limits on the x-axis.
ylim	The limits on the y-axis.
...	Arguments to be passed to methods, such as graphical parameters (see par).

See Also

delvor.

Examples

```
## Not run:
# Random sample in the unit square
x <- matrix(runif(100), nc = 2)
# Delaunay triangulation and Voronoi diagram
delvor.obj <- delvor(x)
# Plot Voronoi diagram and Delaunay triangulation
plot(delvor.obj)

## End(Not run)
```

---

rkoch   *Random generation on a Koch snowflake curve*

Description

This function generates ramdom points from a uniform distribution on a Koch snowflake.

Usage

```
rkoch(n, side = 3, niter = 5)
```

Arguments

n	Number of observations.
side	Side length of the initial equilateral triangle of the Koch snowflake curve.
niter	Number of iterations in the development of the snowflake curve.

Value

A 2-column matrix with the coordinates of generated points.

See Also

[koch](#).

Examples

```
## Not run:
unifkoch <- rkoch(2000, side = 1, niter = 3)
plot(unifkoch, asp = TRUE)

## End(Not run)
```

---

rotation   *Clockwise rotation*

Description

This function calculates the clockwise rotation of angle θ of a given vector v in the plane.

Usage

```
rotation(v, theta)
```

Arguments

v	Vector v in the plane.
theta	Angle θ (in radians).

Value

v.rot	Vector after rotation.

Examples

```
## Not run:
# Rotation of angle pi/4 of the vector (0,1)
rotation(v = c(0, 1), theta = pi/4)

## End(Not run)
```

Index