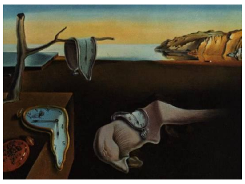

Journal of Statistical Software

April 2010, Volume 34, Issue 5. *http://www.jstatsoft.org/*

Generalizing the Convex Hull of a Sample: The R Package alphahull

Beatriz Pateiro-López
Universidad de Santiago de Compostela

Alberto Rodríguez-Casal
Universidad de Santiago de Compostela

Abstract

This paper presents the R package **alphahull** which implements the α-convex hull and the α-shape of a finite set of points in the plane. These geometric structures provide an informative overview of the shape and properties of the point set. Unlike the convex hull, the α-convex hull and the α-shape are able to reconstruct non-convex sets. This flexibility make them specially useful in set estimation. Since the implementation is based on the intimate relation of theses constructs with Delaunay triangulations, the R package **alphahull** also includes functions to compute Voronoi and Delaunay tesselations. The usefulness of the package is illustrated with two small simulation studies on boundary length estimation.

Keywords: set estimation, convexity, α-convexity, α-convex hull, α-shape, R.

1. Introduction

The problem of reconstructing a set S from a finite set of points taken into it has been addressed in different fields of research. In computational geometry, for instance, the efficient construction of convex hulls for finite sets of points has important applications in pattern recognition, cluster analysis and image processing, among others. We refer the reader to Preparata and Shamos (1985) for an introduction to computational geometry and its applications. From a probabilistic point of view, the set of points from which we try to reconstruct S is assumed to be non-deterministic. Thus, the term set estimation refers to the statistical problem of estimating an unknown set given a random sample of points whose distribution is closely related to it. Under this perspective, the target S might be, for example, a distribution support, its boundary or a level set. See Cuevas and Fraiman (2009) for a survey of set estimation theory.

The support estimation problem is formally established as the problem of estimating the support of an absolutely continuous probability measure from independent observations drawn from it. The papers by Geffroy (1964), Rényi and Sulanke (1963), and Rényi and Sulanke (1964) are the first works on support estimation. They deal with the convex case. When S is a convex support the natural estimator is the convex hull of the sample. See Schneider (1988) for classical results on the convex hull estimator. However, when S is not convex, the convex hull of the sample is not an appropriate choice. In a more flexible framework, Chevalier (1976) and Devroye and Wise (1980) proposed to estimate the support (without any shape restriction) of an unknown probability measure by means of a smoothed version of the sample. The problem of support estimation is introduced by Devroye and Wise (1980) in connection with a practical application, the detection of abnormal behaviour of a system, plant or machine. We refer to Korostelëv and Tsybakov (1993) for a compilation of the most relevant theoretical results on the performance of such estimator. Anyhow, there are also approaches in-between the two aforementioned. In Rodríguez-Casal (2007), the estimation of an α-convex support is considered. The α-convexity is a condition that affects the shape of the set of interest but which is much more flexible than convexity and therefore, it allows a wider range of applications. The α-convex hull of the sample is the natural estimator when S is α-convex. In this work we discuss the details on the implementation of this estimator.

Set estimation is also related to another interesting problem, the estimation of certain geometric characteristics of the set S such as the surface area (boundary length in $\mathbb{R}^2$). There are other statistical fields which also cope with problems regarding set measurements as, for example, the stereology. However, stereology focuses on the estimation of certain geometric characteristics of S without needing to reconstruct the set, see e.g., Baddeley and Jensen (2005), Cruz-Orive (2001/02), whereas the primary object of interest of set estimation is the set itself. So in our framework, given a random sample of points in S, the solution to the surface area estimation problem consists in defining an estimator that captures the shape of S and then estimating the surface area of S by means of the surface area of the estimator. Bräker and Hsing (1998) studied the asymptotic properties of the boundary length of the convex hull of a random sample of points in $\mathbb{R}^2$. They obtained the asymptotic normality of the boundary length of the convex hull estimator as well as its convergence rate in mean. In spite of the fact that the results are really significant, they are established on the assumption that the set of interest is convex, which may be too restrictive in practice. As mentioned before, more flexible estimators, such as the α-convex hull, can be considered. The α-shape, introduced by Edelsbrunner, Kirkpatrick, and Seidel (1983), is another geometrical structure that serves to characterize the shape of a set. Its definition is based on a generalization of the convex hull. This article presents the R implementation (R Development Core Team 2008) of the α-convex hull and α-shape of a random sample of points in the plane with the package **alphahull**, see Pateiro-López and Rodríguez-Casal (2009). We also illustrate the applicability of the library with a simulation study on boundary length estimation.

The paper is organized as follows. In Section 2 we introduce some notation and describe the primary estimators under study, the α-convex hull and the α-shape of a random sample of points taken in the set of interest. The details on the implementation in R of the estimators are given in Section 3, along with the comprehensive description of the library **alphahull**. In Section 4, we present a simulation study on boundary length estimation through two examples. Further computational details are given in Section 5. Section 6 concludes with a discussion of the contributions included in this paper. Some open problems are also pointed out.

2. The α-convex hull

Let S be a nonempty compact subset of the d-dimensional Euclidean space $\mathbb{R}^d$, equipped with the norm $\|\cdot\|$. Assume that we are given a random sample $\mathcal{X}_n = \{X_1, \ldots, X_n\}$ from X, where X denotes a random variable in $\mathbb{R}^d$ with distribution P_X and support S. The problem is to find a suitable estimator of S based on the sample. It seems natural that, in order to properly define such estimator, we should impose some geometric restriction on S. As we have already commented, assuming convexity considerably limits the family of sets to estimate. So we focus on a more flexible shape condition, named α-convexity.

We denote by $\mathring{B}(x, r)$ and $B(x, r)$ the open and closed ball with center x and radius r, respectively. Given $A \subset \mathbb{R}^d$, A^c and ∂A will denote the complement and boundary of A, respectively. A set $A \subset \mathbb{R}^d$ is said to be α-convex, for $\alpha > 0$, if $A = C_\alpha(A)$, where

$$C_\alpha(A) = \bigcap_{\{\mathring{B}(x,\alpha):\ \mathring{B}(x,\alpha)\cap A = \emptyset\}} \left(\mathring{B}(x, \alpha)\right)^c \tag{1}$$

is called the α-convex hull of A. Therefore, if A is α-convex any point of A^c can be separated from A by means of an open ball of radius α. Note that the definition of the α-convex hull given by (1) reminds us of the definition of the convex hull, but replacing the open balls of radius α with half-spaces. Regarding the relation between convexity and α-convexity, it can be proved that, if A is convex and closed, then it is also α-convex for all $\alpha > 0$. If the interior of the convex hull is not empty then the reciprocal is also true, see Walther (1999).

Using the same ideas as in the convex case, an estimator for an α-convex support can be defined. Assume that S is α-convex for some $\alpha > 0$. Then it seems reasonable to consider an estimator fulfilling this shape restriction. So, given a sample $\mathcal{X}_n \subset S$, the natural estimator of S is the smallest α-convex set which contains $\mathcal{X}_n$, that is, the α-convex hull of the sample, $C_\alpha(\mathcal{X}_n)$, see Figure 1.

What can we say about the performance of $C_\alpha(\mathcal{X}_n)$ as a support estimator? Like in other contexts, in order to evaluate a set estimator, we need certain measure of the distance between the estimator and the target S. Thus, the performance of a set estimator is usually

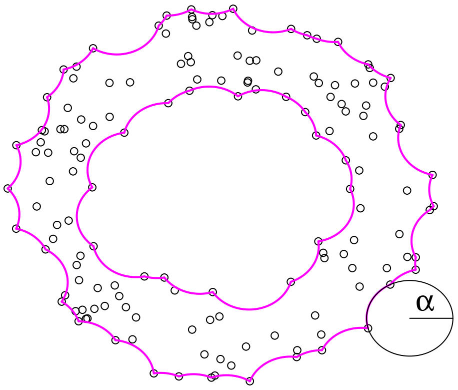

Figure 1: In pink, α-convex hull of a set of points in the plane for some $\alpha > 0$.

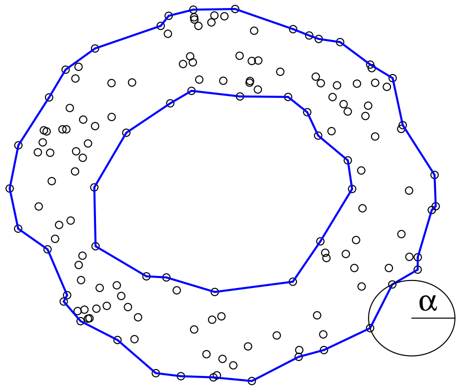

Figure 2: In blue, α-shape of a set of points in the plane for some $\alpha > 0$. Two points are α-neighbours if there exists an open ball of radius α with both points on its boundary and which contains no sample points.

evaluated through either the Hausdorff distance, d_H, or the distance in measure, d_μ, where μ stands for the Lebesgue measure. We refer to Edgar (1990) for a discussion on metrics between sets. Rodríguez-Casal (2007) proved that, if S is α-convex and standard with respect to P_X, then $d_H(S, C_\alpha(\mathcal{X}_n)) = O((\log n/n)^{1/d})$ almost surely. A Borel set A is said to be standard with respect to a Borel measure ν if there exists $\varepsilon_0 > 0$ and $\delta > 0$ such that $\nu(B(x,\varepsilon) \cap A) \geq \delta\mu(B(x,\varepsilon))$, for all $x \in A$, $0 < \varepsilon \leq \varepsilon_0$. The standardness condition prevents the set S from being too spiky, see Cuevas and Fraiman (1997) for more details. Although the family of α-convex sets is much wider than the family of convex sets, $C_\alpha(\mathcal{X}_n)$ achieves the same convergence rates as the convex hull of $\mathcal{X}_n$ in the convex case, see Dümbgen and Walther (1996). Moreover, if S belongs to Serra's regular model, that is, if S is morphologically open and closed with respect to a compact ball of radius α (see Serra 1984), then $d_H(S, C_\alpha(\mathcal{X}_n)) = O((\log n/n)^{2/(d+1)})$ almost surely. Again, the α-convex hull is able to achieve the same convergence rate as the convex hull when S belongs to the Serra's regular model.

Edelsbrunner *et al.* (1983) defined in $\mathbb{R}^2$ a similar construct, the λ-hull of a finite set of points in the plane, for an arbitrary $\lambda \in \mathbb{R}$. Following their terminology, $C_\alpha(\mathcal{X}_n)$ equals the λ-hull of $\mathcal{X}_n$ for $\lambda = -1/\alpha$ and it can be computed in time $O(n \log n)$ using $O(n)$ space. The algorithm, described in Section 3, is based on the closed relationship that exists between λ-hulls and Delaunay triangulations. The Delaunay triangulation of a finite set of points contains, as subgraphs, various structures that have important applications in pattern recognition, cluster analysis, etc. See Aurenhammer and Klein (2000) for a survey. The α-shape is one of those subgraphs, derived from a straightforward generalization of the convex hull. For $\alpha > 0$, the α-shape of $\mathcal{X}_n$ is defined as the straight line graph connecting α-neighbour points. Two points $X_i, X_j \in \mathcal{X}_n$ are α-neighbour if there exists an open ball of radius α with both points on its boundary and which contains no sample points, see Figure 2. The definition of α-shape can be extended to arbitrary real α and higher dimensions, see Edelsbrunner and Mücke (1994). The α-shape is an approach to formalize the intuitive notion of shape for spatial point sets.

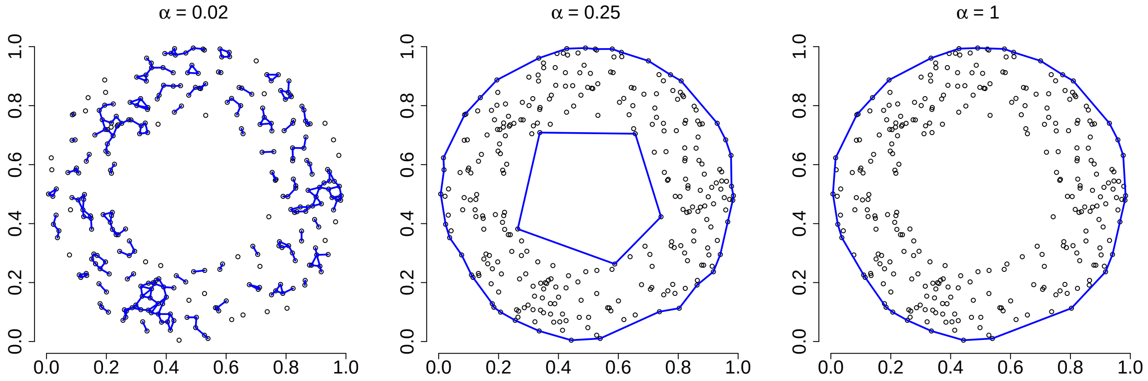

Figure 3: Influence of the parameter α on the α-shape. From left to right, α-shape of a random sample of size $n = 300$ on the annulus with outer radius 0.5 and inner radius 0.25 for $\alpha = 0.02, 0.25, 1$.

The value of the parameter α controls the shape of the estimator. For sufficiently large α, the α-shape is identical to the boundary of the convex hull of the sample. As α decreases, the shape shrinks until that, for sufficiently small α, the α-shape is the empty set, see Figure 3. As with the α-convex hull, the α-shape of n points in the plane can be determined in time $O(n \log n)$ and space $O(n)$, see Edelsbrunner *et al.* (1983). The description of the algorithm and the details of its implementation in R are given in Section 3.

3. Implementation

The R package **alphahull** consists of three main functions: `delvor`, `ashape`, and `ahull`. The implementation of the α-convex hull and of the α-shape is based on the intimate relation of theses constructs with Delaunay triangulations. The function `delvor` described in Section 3.1 computes the Delaunay triangulation and the Voronoi diagram of a given sample of points in the plane. Based on the information provided by the function `delvor`, the function `ashape` described in Section 3.2 constructs the α-shape for a given value of $\alpha > 0$. Finally, the function `ahull` described in Section 3.3 constructs the α-convex hull. The R package **alphahull** also includes plot functions for the different objects and some other auxiliary functions which are described throughout this section.

3.1. Voronoi diagram and Delaunay triangulation

The Voronoi diagram of a finite sample of points $\mathcal{X}_n = \{X_1, \ldots, X_n\}$ in $\mathbb{R}^2$ is a covering of the plane by n regions V_i where, for each $i \in \{1, \ldots, n\}$, the cell V_i consists of all points in $\mathbb{R}^2$ which have X_i as nearest sample point. That is, $V_i = \{x \in \mathbb{R}^2 : \ \|x - X_i\| \leq \|x - X_j\| \text{ for all } X_j \in \mathcal{X}_n\}$. We denote the Voronoi Diagram of $\mathcal{X}_n$ by $VD(\mathcal{X}_n)$. The Voronoi cells are closed and convex. Furthermore, V_i is unbounded if and only if X_i lies on the boundary of the convex hull of $\mathcal{X}_n$. Otherwise V_i is a nonempty convex polygon. Two sample points X_i and X_j are said to be Voronoi neighbours if the cells V_i and V_j share a common point.

A triangulation of $\mathcal{X}_n$ is a planar graph with vertex set $\mathcal{X}_n$ and straight line edges that partition

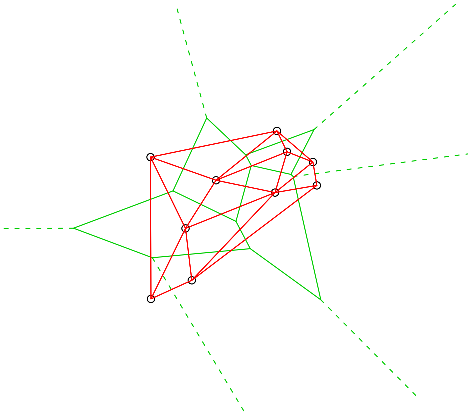

Figure 4: There is a one-to-one correspondence between the Voronoi diagram (in green) and the Delaunay triangulation (in red).

into triangles the convex hull of the nodes $\mathcal{X}_n$. The Delaunay triangulation of $\mathcal{X}_n$, denoted by $DT(\mathcal{X}_n)$, is defined as the straight line dual to $VD(\mathcal{X}_n)$. That is, there exists a Delaunay edge connecting the sample points X_i and X_j if and only if they are Voronoi neighbours, see Figure 4. In other words, each circumcentre of a Delaunay triangle coincides with a Voronoi cell vertex. The Delaunay triangulation was introduced by Voronoi (1908) and extended by Delaunay (1934), to whom it owes its name. A complete overview over the existing literature on these geometric constructions can be found in Okabe, Boots, Sugihara, and Chiu (2000). We also refer to the survey by Aurenhammer (1991) for more details on the properties of the Delaunay triangulations and the Voronoi diagrams and their multiple applications.

From the computational viewpoint, efficient methods for the computation of Delaunay triangulations and Voronoi diagrams have been developed. Aurenhammer and Klein (2000) presented a review of algorithms, from the earliest intuitive methods to more efficient representations of these geometric structures. For example, the incremental insertion process by Green and Sibson (1978), the Divide and Conquer method by Shamos and Hoey (1975) or the plane-sweep technique by Fortune (1987) are the basis of a large class of worst-case optimal algorithms for computing the whole Voronoi diagram in the plane. Of course, these techniques can also be applied to the computation of the Delaunay triangulation. See for example the efficient incremental algorithm by Lawson (1977). Both the Voronoi diagram and the Delaunay triangulation of n points can be computed in $O(n \log n)$ time and linear space. Furthermore, by the duality between the Voronoi diagram and the Delaunay triangulation, either tessellation is easily obtained from a representation of the other in $O(n)$ time.

Currently, there are several libraries available in R that compute the Delaunay triangulation or the Voronoi diagram of a given set of points. See the packages **deldir** by Turner (2009) or **geometry** by Grasman and Gramacy (2008), among others. These libraries differ on the implemented algorithms and the data structures that store the information. For example, the package **deldir** computes the Delaunay triangulation and the Voronoi diagram of a planar point set according to the second iterative algorithm of Lee and Schachter (1980). Unfortu-

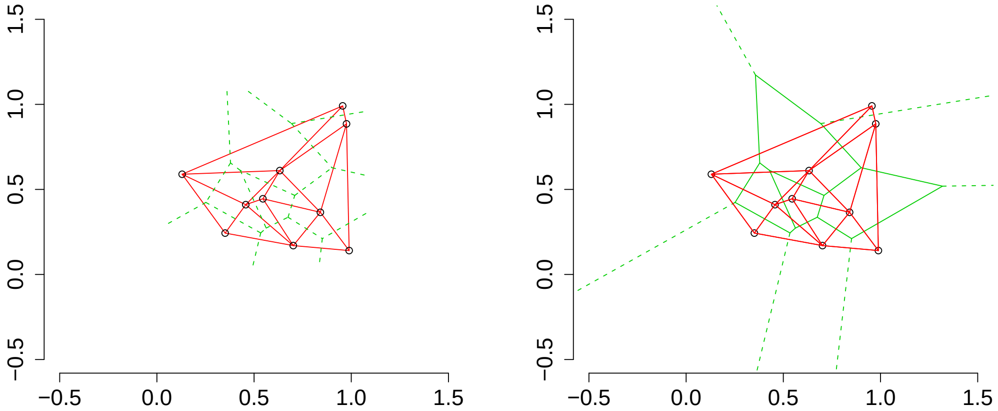

Figure 5: Voronoi diagram and Delaunay triangulation of a uniform random sample of size $n = 25$ on the unit square obtained from package **deldir** (left) and package **alphahull** (right).

nately, this package does not return the kind of data structure we need in order to compute the α-shape and the α-convex hull. The function `deldir` computes the triangulation of a set of points enclosed in a finite rectangular window. In consequence, the endpoints of the Voronoi edges outside that window are discarded. This fact does not appear to be a problem unless we need to know all the Voronoi edges, as in our case. Note that, in principle, the location of the furthest Voronoi vertex is unknown and enlarging the window size to ensure that the information is complete has a considerable computational cost. Our package **alphahull** computes the Delaunay triangulation and the Voronoi diagram with respect to the whole plane, see Figure 5. The function `delvor` included in the library internally calls the **TRIPACK** Fortran 77 software package by Renka (1996) that employs an incremental algorithm to construct a constrained Delaunay triangulation of a set of points in the plane. Then, the Voronoi diagram is derived via dualization. For each edge of the Delaunay triangulation the corresponding segment in the Voronoi diagram is obtained by connecting the circumcenters of the two neighbour triangles that share that edge. For those edges of the Delaunay triangulation that lie on the boundary of the convex hull, the corresponding segments in the Voronoi diagram are semi-infinite. We compute the triangulation by invoking the function `tri.mesh` from package **tripack**, see Renka and Gebhardt (2009). The code to compute the corresponding Voronoi diagram is a corrected version of the function `voronoi.mosaic` which is also included in the package **tripack**. We have observed that, for certain sets of points, `voronoi.mosaic` fails when computing the so called dummy nodes. These points represent the endpoints of the unbounded Voronoi edges, see Figure 6.

The output of the function `delvor` is a list with three components. The first component, `mesh`, stores the primal and dual information. For each edge of the Delaunay triangulation `mesh` contains the indexes `ind1`, `ind2` and the coordinates (`x1`, `y1`), (`x2`, `y2`) of the sample points that form the edge, the coordinates of the endpoints of the corresponding segment in the Voronoi diagram (`mx1`, `my1`), (`mx2`, `my2`), and an indicator that takes the value 1 for those endpoints of the Voronoi edges that represent a boundless extreme, that is, `bp1 = 1`

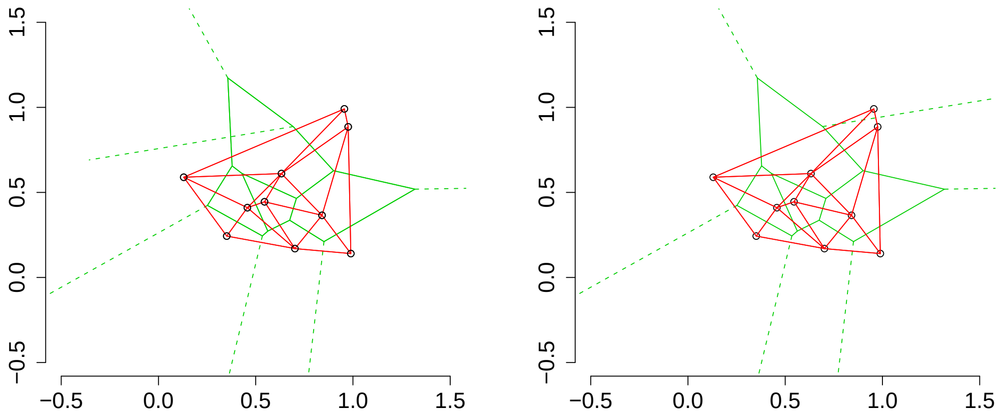

Figure 6: Voronoi diagram and Delaunay triangulation of a uniform random sample of size $n = 10$ on the unit square obtained from package **tripack** (left) and package **alphahull** (right). The Voronoi diagram returned by the function `voronoi.mosaic` from **tripack** is not well defined.

if (`mx1`, `my1`) is a dummy node and the same for `bp2`. The second component, `x`, stores the sample points and the third component, `tri.obj`, stores the information of the Delaunay triangulation obtained from the function `tri.mesh`. As an illustration, we have applied the function `delvor` to a uniform random sample of size $n = 5$ on the unit square. The result is assigned to the object `dv` and printed out.

```
R> x <- matrix(runif(10), ncol = 2)
R> dv <- delvor(x)
R> dv
```

```
$mesh
 ind1 ind2        x1         y1        x2         y2       mx1       my1
    2    5 0.6066043 0.93781202 0.7440690 0.73178007 0.7920124 0.9126422
    5    4 0.7440690 0.73178007 0.9885872 0.25985470 0.8741500 0.4998702
    3    5 0.4580710 0.09584452 0.7440690 0.73178007 0.2685215 0.5633686
    1    5 0.9055416 0.76391156 0.7440690 0.73178007 0.7920124 0.9126422
    1    4 0.9055416 0.76391156 0.9885872 0.25985470 0.8741500 0.4998702
    1    2 0.9055416 0.76391156 0.6066043 0.93781202 0.7920124 0.9126422
    2    3 0.6066043 0.93781202 0.4580710 0.09584452 0.2685215 0.5633686
    3    4 0.4580710 0.09584452 0.9885872 0.25985470 0.6583438 0.3880547
       mx2       my2 bp1 bp2
  0.2685215  0.5633686   0   0
  0.6583438  0.3880547   0   0
  0.6583438  0.3880547   0   0
  0.8741500  0.4998702   0   0
  2.5733037  0.7798133   0   1
  1.9802581  2.9552532   0   1
 -0.4374804  0.6879159   0   1
  1.1711731 -1.2707707   0   1

$x
          [,1]       [,2]
[1,] 0.9055416 0.76391156
[2,] 0.6066043 0.93781202
[3,] 0.4580710 0.09584452
[4,] 0.9885872 0.25985470
[5,] 0.7440690 0.73178007

$tri.obj
triangulation nodes with neigbours:
node: (x,y): neighbours
1: (0.9055416,0.7639116) [3]: 2 4 5
2: (0.6066043,0.937812) [3]: 1 3 5
3: (0.458071,0.09584452) [3]: 2 4 5
4: (0.9885872,0.2598547) [3]: 1 3 5
5: (0.744069,0.7317801) [4]: 1 2 3 4
number of nodes: 5
number of arcs: 8
number of boundary nodes: 4
boundary nodes:  1 2 3 4
number of triangles: 4
number of constraints: 0

attr(,"class")
[1] "delvor"
```

The plot of the previous Delaunay triangulation and Voronoi diagram is displayed in Figure 7. This graph is produced using the function `plot.delvor` (S3 method for class 'delvor'). The arguments are the same as those of the function `plot.deldir` from package **deldir**.

```
R> plot(dv, main = "Delaunay triangulation and Voronoi diagram",
+    col = 1:3, xlab = "x-coordinate", ylab = "y-coordinate",
+    xlim = c(-0.5, 1.5), ylim = c(-0.5, 1.5), number = TRUE)
```

3.2. The α-shape

For the construction of the α-shape of a finite sample of points $\mathcal{X}_n$, the package **alphahull** implements the algorithm by Edelsbrunner *et al.* (1983), see Table 1.

The algorithm relies on the notion of α-extreme point and α-neighbour. A sample point X_i is termed α-extreme if there exists an open ball of radius α with X_i on its boundary and which contains no sample points. Finding the α-extreme points is not a difficult task once the Delaunay triangulation and the Voronoi diagram are determined. Note that, if the sample

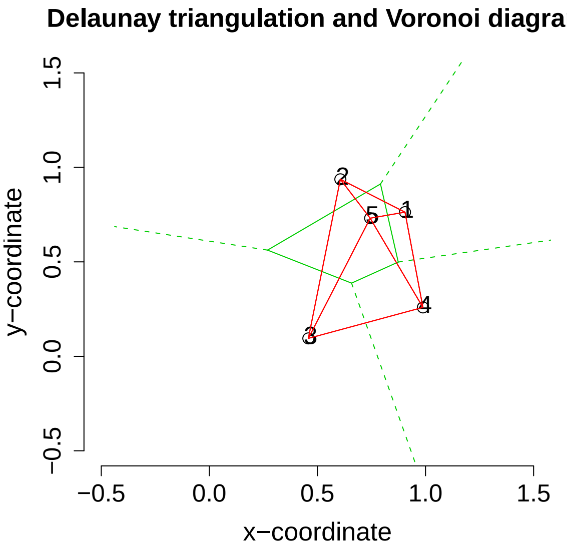

Figure 7: Plot of the Delaunay triangulation (red) and Voronoi diagram (green) of a uniform random sample of size $n = 5$ on the unit square. The dashed green lines correspond to the unbounded Voronoi edges.

points X_i lies on the convex hull of $\mathcal{X}_n$, then X_i is α-extreme for all $\alpha > 0$. If X_i does not lie on the convex hull we only need to compute the distances from X_i to the vertexes of the Voronoi cell V_i. Then, X_i is α-extreme for all α satisfying $0 < \alpha \leq \max\{\|X_i - v\|,\ v \text{ vertex of } V_i\}$, see the left-hand side of Figure 8. Finding the α-neighbour points is, however, trickier. Consider an edge of the Delaunay triangulation connecting the sample points X_i and X_j and its dual edge of the Voronoi diagram. The points X_i and X_j are α-neighbours for all α satisfying $\alpha_{\min} \leq \alpha \leq \alpha_{\max}$, where $\alpha_{\min}$ and $\alpha_{\max}$ are determined from the position of X_i and X_j with respect to the vertexes of the dual Voronoi edge, see the right-hand side of Figure 8.

The function `ashape`, included in the library **alphahull**, computes the α-shape of a given sample $\mathcal{X}_n$ for a given value of $\alpha > 0$. The output of the function `ashape` is a list with six components. The components `x` and `alpha` store the input information whereas the component `delvor.obj` stores the output object from the function `delvor`, see Section 3.1. The indexes of the α-extremes are given by the component `alpha.extremes`. The α-neighbours connections are given by the first two columns of 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`. The length of the

```
**Algorithm** $\alpha$-shape
1: Construct the Voronoi diagram and Delaunay triangulation of $\mathcal{X}_n$.
2: Determine the $\alpha$-extremes of $\mathcal{X}_n$.
3: Determine the $\alpha$-neighbours of $\mathcal{X}_n$.
4: Output the $\alpha$-shape.
```

Table 1: Algorithm for computing the α-shape.

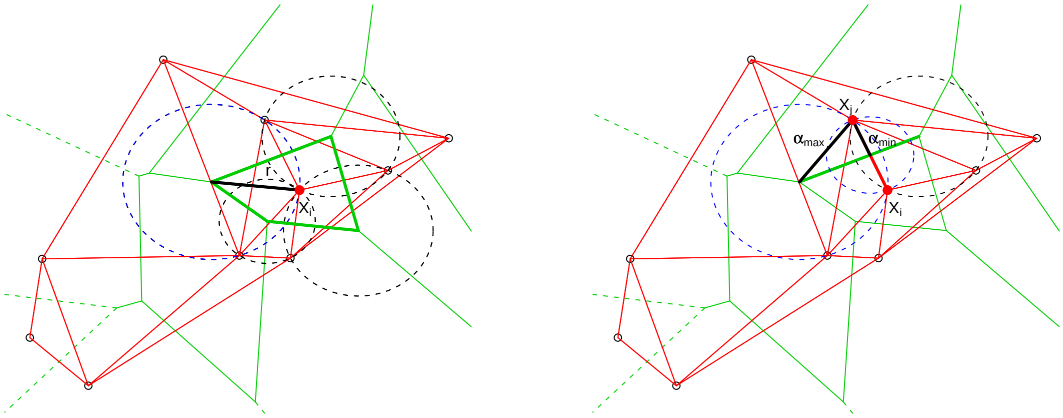

Figure 8: Delaunay triangulation (in red) and Voronoi diagram (in green) of a random sample of size $n = 10$. Left: The sample point X_i is α-extreme for all $\alpha \leq r$, where r is the maximum distance from X_i to the vertexes of the Voronoi cell V_i (in thick solid line). Right: The sample points X_i and X_j are α-neighbours for all α satisfying $\alpha_{\min} \leq \alpha \leq \alpha_{\max}$.

α-shape is stored in the component `length`. The function `plot.ashape` (S3 method for class 'ashape') produces a plot of the α-shape. Graphic parameters can control the plot appearance. Moreover, the Delaunay triangulation and the Voronoi diagram can be added to the plot by specifying the argument `wlines` (`wlines = "del"` for the Delaunay triangulation, `wlines = "vor"` for the Voronoi diagram or `wlines = "both"` for both plots). Next, we show an example on how these functions work. We have applied the function `ashape` to a uniform random sample of size $n = 20$ on the unit square with $\alpha = 0.2$. The result is assigned to the object `alphashape`.

```
R> x <- matrix(runif(40), ncol = 2)
R> alpha <- 0.2
R> alphashape <- ashape(x, alpha = alpha)
R> names(alphashape)

[1] "edges"          "length"         "alpha"          "alpha.extremes"
[5] "delvor.obj"     "x"

R> alphashape$alpha.extremes

 [1]  5 14  9  6 12 11 13  2  3  8 19 20

R> alphashape$edges[, 1:2]

      ind1 ind2
 [1,]   12    6
 [2,]   13   20
 [3,]    8   14
 [4,]   13   19
 [5,]    2    3
 [6,]    2   12
 [7,]    3   11
 [8,]    3   19
 [9,]    5   20
[10,]    5    8
[11,]    6    9
[12,]    9   14
```

```
R> alphashape$length
```

```
[1] 3.076418
```

A plot of the α-shape may be obtained as follows. The result is displayed in Figure 9.

```
R> plot(alphashape, col = c(4, 1), xlab = "x-coordinate", ylab = "y-coordinate",
+    number = TRUE, main = expression(paste(alpha, "-shape")))
```

Also the Delaunay triangulation can be plotted by specifying the argument `wlines` as in the following code. The result is displayed in Figure 10.

```
R> plot(alphashape, wlines = "del", col = c(4, 1, 2), xlab = "x-coordinate",
+    ylab = "y-coordinate",
+    main = expression(paste(alpha, "-shape and Delaunay triangulation")))
```

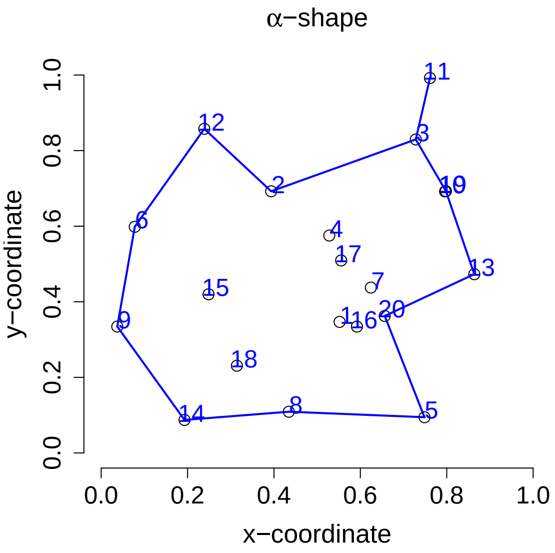

Figure 9: Plot of the α-shape of a uniform random sample of size $n = 20$ on the unit square for $\alpha = 0.2$.

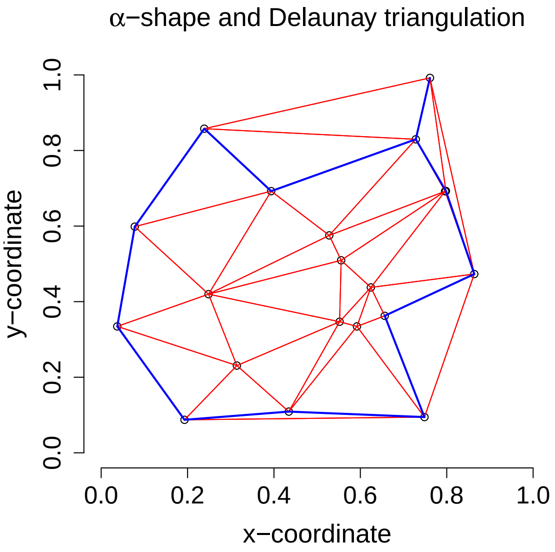

Figure 10: Plot of the α-shape (in blue) and Delaunay triangulation (in red) of a uniform random sample of size $n = 20$ on the unit square for $\alpha = 0.2$. The α-shape is a subgraph of the Delaunay triangulation.

3.3. The α-convex hull

Recall the definition of the α-convex hull given in Equation (1). By DeMorgan's law, the complement of $C_\alpha(\mathcal{X}_n)$ can be written as the union of all open balls of radius α which contain no point of $\mathcal{X}_n$, that is,

$$C_\alpha(\mathcal{X}_n)^c = \bigcup_{\{\mathring{B}(x,\alpha):\ \mathring{B}(x,\alpha)\cap\mathcal{X}_n=\emptyset\}} \mathring{B}(x,\alpha). \tag{2}$$

Therefore, in order to compute the complement of the α-convex hull of a sample of points we should identify all those balls of radius α that do contain no point of the sample. Fortunately, the problem simplifies thanks to the following lemma by Edelsbrunner *et al.* (1983).

Lemma 3.1. *Let $\mathring{B}(x, r)$ be an open ball which does not contain any point of a sample $\mathcal{X}_n$. Either $\mathring{B}(x, r)$ lies entirely outside the convex hull of $\mathcal{X}_n$ or there is an open ball which contains $\mathring{B}(x, r)$ but no points of $\mathcal{X}_n$ and which has its centre on an edge of $VD(\mathcal{X}_n)$.*

Therefore, we only have to consider appropriate balls centered on edges of $VD(\mathcal{X}_n)$. For the implementation we have considered the two following situations. First, if X_i and X_j are two sample points such that the corresponding Voronoi cells V_i and V_j share a common closed line segment $[a, b]$, then it follows from the duality between the $VD(\mathcal{X}_n)$ and $DT(\mathcal{X}_n)$ that the union of open balls with centres on the edge $[a, b]$ which do not contain any point of a sample is equal to $\mathring{B}(a, \|a - X_i\|) \cup \mathring{B}(b, \|b - X_i\|)$, see the left-hand side of Figure 11. Second, if X_i and X_j are two sample points such that the corresponding Voronoi cells V_i and V_j share a common semi-infinite line segment $[a, +\infty)$, then the union of open balls with centres on the edge $[a, +\infty)$ which do not contain any point of a sample can be written as $\mathring{B}(a, \|a - X_i\|) \cup H(X_i, X_j)$, where $H(X_i, X_j)$ denotes the open halfplane defined by the

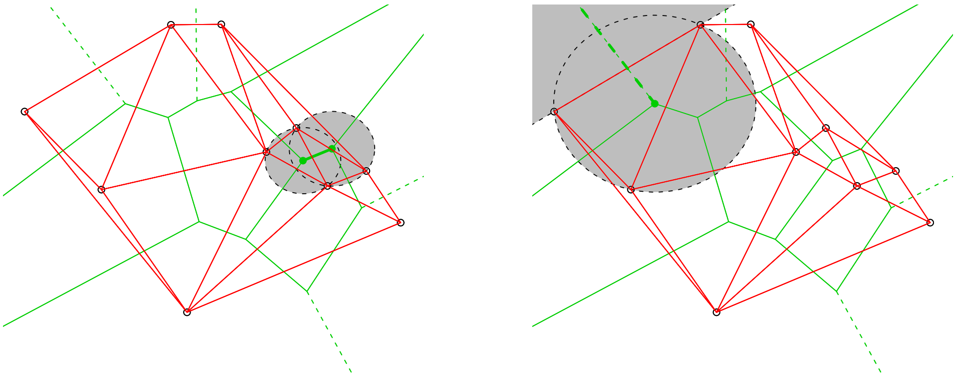

Figure 11: Delaunay triangulation (in red) and Voronoi diagram (in green) of a random sample of size $n = 10$. Left: Union of open balls centered on a closed Voronoi edge. Right: Union of open balls centered on a semi-infinite closed Voronoi edge.

straight line through X_i and X_j, see the right-hand side of Figure 11. It can be shown that, for each edge of the Voronoi diagram, the union of open balls with centres on it and radius α which do not contain any point of $\mathcal{X}_n$ can be written as the union of a finite number of open balls or halfplanes. As a consequence, the complement of the α-convex hull of n points can be expressed as the union of $O(n)$ open balls and halfplanes, see Lemma 6 by Edelsbrunner *et al.* (1983). The package **alphahull** implements the algorithm in Table 2.

The function `ahull`, included in the library **alphahull**, computes the α-convex hull of a given sample $\mathcal{X}_n$ for a given value of $\alpha > 0$. The output of the function `ahull` is a list with six components. The components `x` and `alpha` store the input information whereas the component `ashape.obj` stores the output object from the function `ashape` (see Section 3.2) which is invoked during the computation of the α-convex hull. Information about the open balls and halfplanes that define the complement of the α-convex hull is stored in the component `complement`. For each row i, `complement[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. Thus, if the row i refers to an open ball, `complement[i, 1:3]` contains the center and radius of the ball. If the row i refers to a halfplane, `complement[i, 1:3]` determines its equation. For the halfplane $y > a + bx$, `complement[i, 1:3]=(a, b, -1)`. In the same way, for the halfplane $y < a + bx$, `complement[i, 1:3]=(a, b, -2)`,

```
Algorithm α-convex hull
1: Construct the Voronoi diagram and Delaunay triangulation of $\mathcal{X}_n$.
2: Determine the union of open balls and halfplanes that form $C_\alpha(\mathcal{X}_n)^c$.
3: Output the $\alpha$-convex hull.
```

Table 2: Algorithm for computing the α-convex hull.

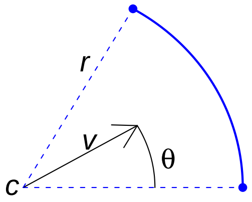

Figure 12: The extremes of an arc can be written as $c + rA_\theta(v)$ and $c + rA_{-\theta}(v)$, where $A_\theta(v)$ represents the clockwise rotation of angle θ of the unitary vector v.

for the halfplane $x > a$, `complement[i, 1:3]=(a, 0, -3)` and for the halfplane $x < a$, `complement[i, 1:3]=(a, 0, -4)`. On the other hand, the component `arcs` stores the boundary of the α-convex. Note that $\partial C_\alpha(\mathcal{X}_n)$ is formed by arcs of balls of radius α (besides possible isolated sample points). These 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 can be written as $c + rA_\theta(v)$ and $c + rA_{-\theta}(v)$ where c and r represent the center and radius of the arc, respectively, and $A_\theta(v)$ represents the clockwise rotation of angle θ of a unitary vector v, see Figure 12. Thus, the structure of the matrix `arcs` is as follows. For each arc of the boundary, the first two columns (`c1`, `c2`) correspond to the coordinates of the center c. The third column `r` corresponds to the radius α. The next two columns (`v.x`, `v.y`) correspond to the coordinates of the unitary vector v and the angle θ is stored in the last column `theta`. The matrix `arcs` also stores the sample points that lie on the boundary of the α-convex hull (rows where columns 3 to 6 are equal to zero). Finally, the boundary length of the α-convex hull is stored in the component `length`.

The function `plot.ahull` (S3 method for class 'ahull') produces a plot of the α-convex hull. As with `plot.ashape` and `plot.delvor` some graphic parameters can control the plot appearance. The α-shape can be added to the plot by specifying the argument `do.shape = TRUE`. Moreover, the Delaunay triangulation and the Voronoi diagram can be added to the plot by specifying the argument `wlines` (`wlines = "del"` for the Delaunay triangulation, `wlines = "vor"` for the Voronoi diagram or `wlines = "both"` for both plots). As an example, we have applied the function `ahull` to a uniform sample of size $n = 100$ on the annulus with outer radius 0.5 and inner radius 0.25. The result is assigned to the object `alphahull`.

```
R> alpha <- 0.1
R> alphahull <- ahull(x, alpha = alpha)
R> names(alphahull)
```

```
[1] "arcs"       "length"     "complement" "alpha"      "ashape.obj"
[6] "x"
```

```
R> alphahull$complement[, 1:3]
```

```
                  c1          c2         r
  [1,]     0.9859157   0.2624408 0.1470753
  [2,]      0.9854346     0.2629982  0.1464956
  [3,]      0.2911565     0.9650050  0.1000354
  [4,]          .....         .....      .....

[188,]     2.08475937 -37.350594570 -2.0000000
[189,]     0.87392545   0.206752552 -1.0000000
[190,]    -0.05950829   0.179047491 -2.0000000
```

```
R> alphahull$arcs
```

```
                 c1          c2   r        v.x        v.y       theta
  [1,]    0.9727443  0.30951251 0.1  -0.2632053  0.9647398   0.3255635
  [2,]    0.9415490  0.29557316 0.1  -0.8029687  0.5960211   0.6837166
  [3,]    0.9337136  0.25201205 0.1  -0.9806226 -0.1959064   0.5982983
  [4,]    0.8762460  0.08137977 0.1  -0.7360387  0.6769395   0.7016039
  [5,]    0.7854558 -0.01400530 0.1  -0.3084052  0.9512551   0.2221727
  [6,]    0.6398895  0.03934587 0.1   0.7451447  0.6669028   0.3975462
  [7,]    0.6288842  0.04544647 0.1  -0.2913489  0.9566168   0.8647566
  [8,]    0.5345367 -0.01462047 0.1  -0.5636399  0.8260206   0.6253032
  [9,]        .....       ..... ...       .....      .....       .....
```

```
R> alphahull$length
```

```
[1] 5.827047
```

A plot of the α-convex hull may be obtained as follows. The result is displayed in Figure 13.

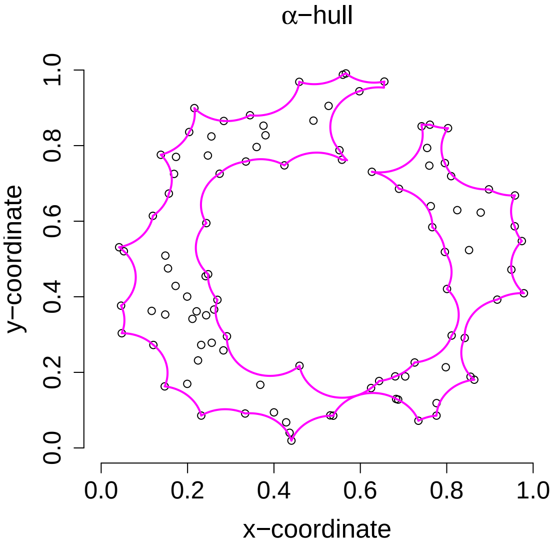

Figure 13: Plot of the α-convex hull of a uniform random sample of size $n = 100$ on the annulus with outer radius 0.5 and inner radius 0.25. The value of α is 0.1.

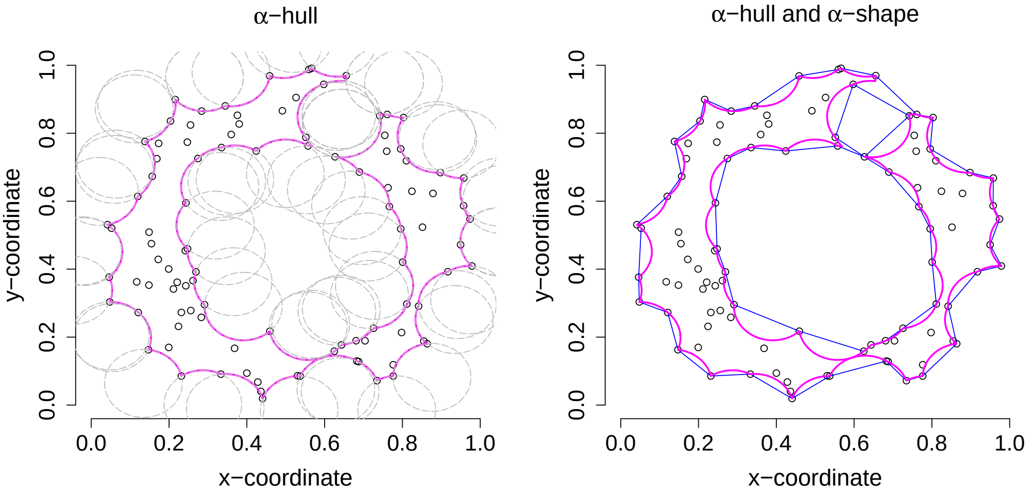

Figure 14: Left: plot of the α-convex hull and balls defining the complement. Right: plot of the α-convex hull (in pink) and α-shape (in blue).

```
R> plot(alphahull, col = c(6, rep(1, 5)), xlab = "x-coordinate",
+    ylab = "y-coordinate", main = expression(paste(alpha, "-hull")))
```

The plot of the α-convex hull together with some of the balls of radius α that define its complement, see Equation (2), is displayed in the left-hand side of Figure 14. This plot is obtained by using the information of the component `arcs` as shown in the following code. The function `arc` included in the library plots an arc given its center, radius, the unitary vector v and the angle θ, see Figure 12.

```
R> plot(alphahull, col = c(6, rep(1, 5)), xlab = "x-coordinate",
+    ylab = "y-coordinate", main = expression(paste(alpha, "-hull")))
R> warcs <- which(alphahull$arcs[, 3] > 0)
R> for(i in warcs) {
+    arc(alphahull$arcs[i, 1:2], alphahull$arcs[i, 3], c(0, 1),
+      pi, col = "gray", lty = 2)
+  }
```

The relation between the α-convex hull and the α-shape is shown in the right-hand side of Figure 14. In order to plot both the α-convex hull and the α-shape, specify the parameter `do.shape = TRUE` as in the following code.

```
R> plot(alphahull, do.shape = TRUE, col = c(6, 4, rep(1, 4)),
+    xlab = "x-coordinate", ylab = "y-coordinate",
+    main = expression(paste(alpha, "-hull and ",alpha, "-shape")))
```

Once $C_\alpha(\mathcal{X}_n)^c$ is constructed we can decide whether a given point $p \in \mathbb{R}^2$ belongs to the α-convex hull or not, by checking if it belongs to any of the open balls or halfplanes that form the complement of the α-convex hull. The function `inahull` returns a logical value specifying whether p belongs to $C_\alpha(\mathcal{X}_n)^c$. For the previous example:

```
R> inahull(alphahull, c(0.5, 0.5))
```

```
[1] FALSE
```

```
R> inahull(alphahull, c(0.8, 0.2))
```

```
[1] TRUE
```

4. Boundary length estimation

Boundary length estimation is an interesting geometrical problem that has been studied in several fields like stereology or set estimation. It also has some relevant applications. For example, the ratio between the boundary length and the squared root of the area is a scale invariant measurement of boundary roughness. Small values of this ratio correspond to round and non fragmented sets whereas large values suggest irregular boundaries and/or fragmented sets. This boundary roughness measurement has been used as an auxiliary diagnosis criterion to distinguish from a good or bad prognosis based on medical images in fields like oncology or cardiology. See, for instance, Cuevas, Fraiman, and Rodríguez-Casal (2007).

4.1. Simulation study

Here we illustrate, with a small simulation study, the ability of the α-convex hull and the α-shape to estimate the boundary length of an unknown set based on a random sample of points taken on it. We have considered the set S in Figure 15. The exact value of the boundary length of the set S is 3.1612. We shall use this information to evaluate the performance of the estimators.

For the study we have generated 500 uniform samples of size $n = 1000$ on S. For each sample $\mathcal{X}_n$ we construct $C_\alpha(\mathcal{X}_n)$ and the corresponding α-shape, denoted by $\alpha^{\text{shape}}(\mathcal{X}_n)$. The

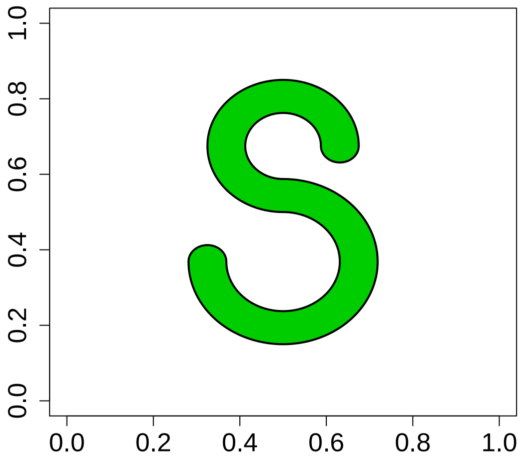

Figure 15: Set S with boundary length 3.1612.

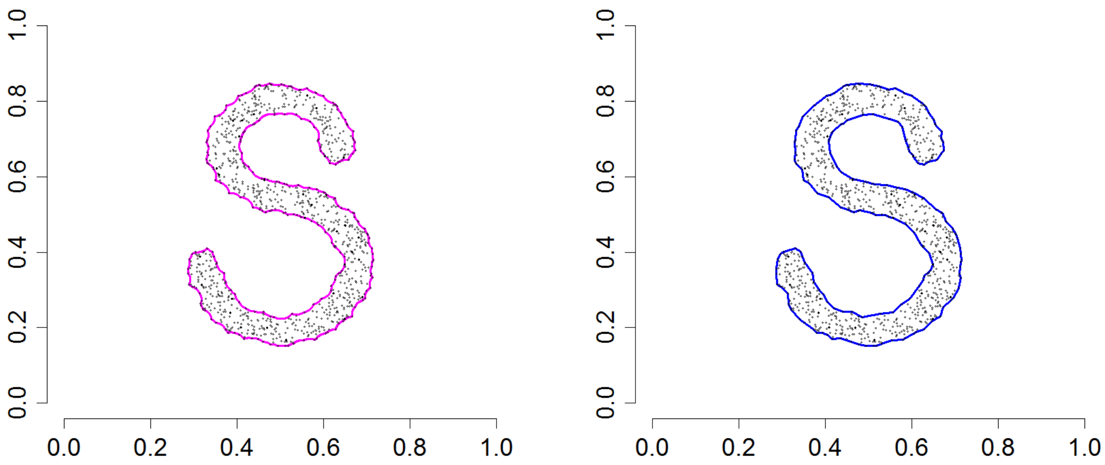

Figure 16: Uniform sample $\mathcal{X}_n$ of size $n = 1000$ on S. Left: α-convex hull of $\mathcal{X}_n$ for $\alpha = 0.035$. Right: α-shape of $\mathcal{X}_n$ for $\alpha = 0.035$.

boundary length of S is estimated through the boundary length of both estimators. Note that the α-convex hull and the α-shape have the drawback of depending on the parameter α, which in general may be unknown. From the theoretical viewpoint, nothing is known about how to select the parameter α. However, some preliminary results for the α-convex hull of a sample, see Rodríguez-Casal (2007), suggest that it is advisable to choose the largest α for which the set of interest is α-convex, at least if you want the set to be properly recovered. In this example, this value is $\alpha = 0.035$, see Figure 16. Even so, we have evaluated the estimators for different values of α in order to study the influence of this parameter on the estimations. The results are summarized in Tables 3, 4, and 5.

For those values of α close to $\alpha = 0.035$, the results are quite reasonable. Small values of α provide, however, considerably biased estimations, especially in the case of the α-shape, see the first column of Table 4. Recall that the α-shape was defined as the straight line graph whose edges connect α-neighbours. When α is small, a considerable number of interior points of the set turn out to be α-extremes and the α-shape looks like a mesh connecting many of them, see Figure 17. As a consequence, the length of the α-shape is large, as it is the result of the addition of many small segments. The estimations are also biased for large values of α, observe the last column of Tables 3 and 4. When α is too large the estimators are not

α	0.01	0.03	0.05	0.07	0.09	0.11
Average	7.5838	3.3180	3.2770	3.2214	2.6603	2.7800
Std. deviation	0.2293	0.0188	0.0342	0.0112	0.0593	0.0152
Median	7.5776	3.3172	3.2853	3.2207	2.6562	2.7816

Table 3: Average, standard deviation and median of the boundary length of $C_\alpha(\mathcal{X}_n)$, based on 500 uniform samples $\mathcal{X}_n$ on S with sample size $n = 1000$. The exact value of the boundary length of S is 3.1612.

α	0.01	0.03	0.05	0.07	0.09	0.11
Average	10.5307	3.1949	3.2151	3.2107	2.7416	2.6949
Std. deviation	0.3310	0.0151	0.0225	0.0990	0.0502	0.0125
Median	10.5308	3.1940	3.2127	3.1514	2.7366	2.6934

Table 4: Average, standard deviation and median of the boundary length of $\alpha^{\text{shape}}(\mathcal{X}_n)$, based on 500 uniform samples $\mathcal{X}_n$ on S with sample size $n = 1000$.

α	0.01	0.03	0.05	0.07	0.09	0.11
$C_\alpha(\mathcal{X}_n)$	19.6117	0.0249	0.0146	0.0037	0.2545	0.1456
$\alpha^{\text{shape}}(\mathcal{X}_n)$	54.4186	0.0014	0.0034	0.0123	0.1786	0.2177

Table 5: Mean square error of the boundary length of $C_\alpha(\mathcal{X}_n)$ and $\alpha^{\text{shape}}(\mathcal{X}_n)$, based on 500 uniform samples $\mathcal{X}_n$ on S with sample size $n = 1000$.

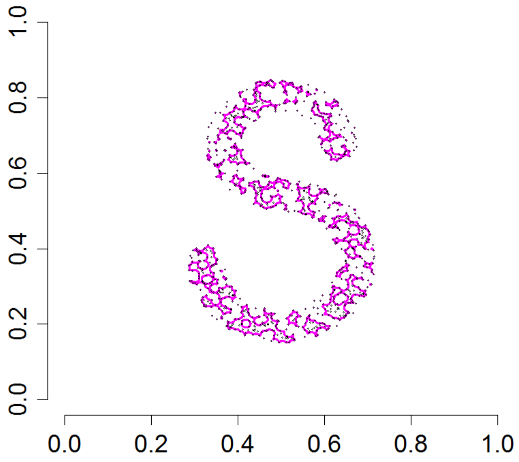

Figure 17: Uniform sample $\mathcal{X}_n$ of size $n = 1000$ on S and α-convex hull of $\mathcal{X}_n$ for $\alpha = 0.01$.

longer able to identify the cavities of the set, see Figure 18. Note that the α-shape tends to the convex hull as α is larger. As a consequence, the boundary length is underestimated.

Finally, we also include some descriptive graphs. Figure 19 shows boxplots of the estimates for different values of α. Due to the bias problems explained before, the scale for the case $\alpha = 0.01$ is much higher than for the rest of values of α. For the sake of clarity, we have omitted this case in the plots.

4.2. The Koch snowflake curve

As a second illustration, we have considered an example from fractal geometry. Fractal shapes can be generated in many ways. The simplest way is to take a generating rule and iterate it

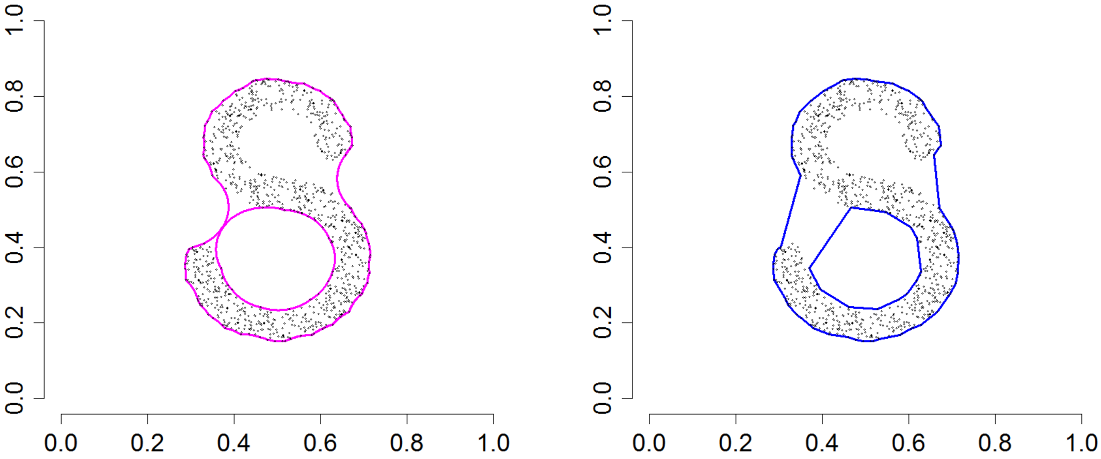

Figure 18: Uniform sample $\mathcal{X}_n$ of size $n = 1000$ on S. Left: α-convex hull of $\mathcal{X}_n$ for $\alpha = 0.11$. Right: α-shape of $\mathcal{X}_n$ for $\alpha = 0.11$.

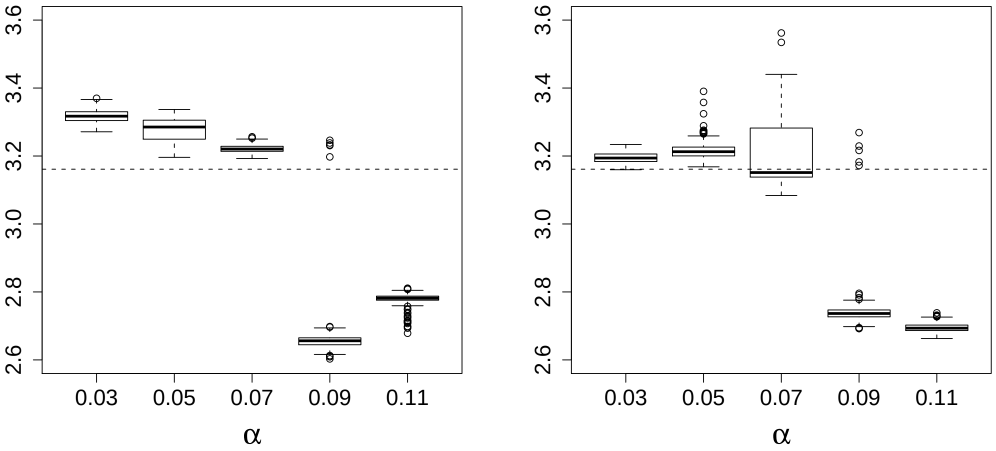

Figure 19: Summary graphs for the set S. Left: boxplots for the boundary length of $C_\alpha(\mathcal{X}_n)$ for different values of α. Right: boxplots for the boundary length of $\alpha^{\text{shape}}(\mathcal{X}_n)$ for different values of α.

over and over again. The Koch snowflake, described by von Koch (1904), is one of the earliest fractal curves. 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, see Figure 20. The package **alphahull** includes the function `koch`, that computes the twist points of a Koch snowflake given the side length of the initial equilateral triangle and the number of iterations. As an example, the code that generates Figure 20 is given below.

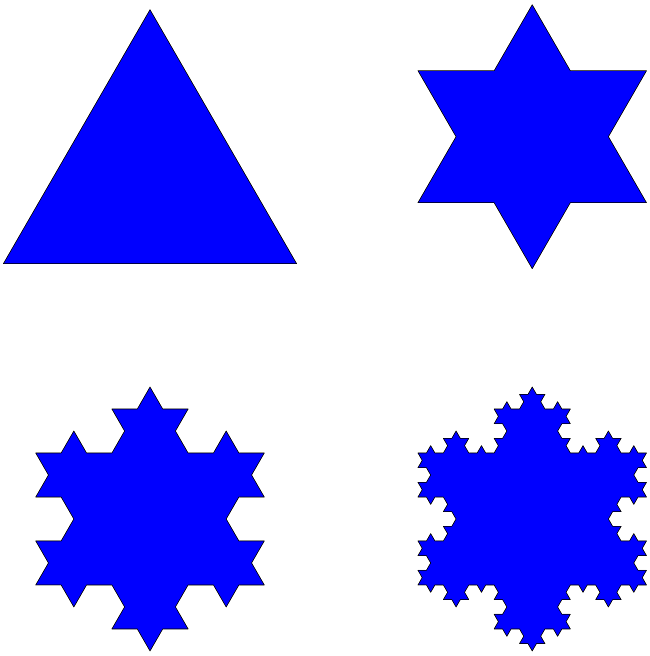

Figure 20: The first four iterations of the Koch snowflake.

```
R> par(mfrow = c(2, 2))
R> for(k in 1:4) {
+    snow <- koch(side = 1, niter = k)
+    plot(snow[, 1], snow[, 2], type = "l", xlab = "", ylab = "",
+      axes = FALSE, asp = TRUE)
+    polygon(snow[, 1], snow[, 2], col = 4)
+  }
```

The snowflake, conceptually, has an infinite length. On iteration k ($k = 1, 2, \ldots$), the curve consists of $3 \cdot 4^{k-1}$ line segments, each of length $l(1/3)^{k-1}$, being l the side length of the initial equilateral triangle. Therefore, the total length of the Koch snowflake on iteration k is $3l(4/3)^{k-1}$. Assume now that we are given a random sample of points into the set enclosed by a Koch snowflake curve. Then, we can approximate the length of the Koch snowflake by means of the perimeter of the α-convex hull or the length of the α-shape of the sample. The function `rkoch` generates random points from a uniform distribution on the set enclosed by a Koch snowflake. In the following example, we use the function `rkoch` to generate $n = 2000$ points on a Koch snowflake with initial side length 1 and 3 iterations. Then, we construct the α-shape and the α-convex hull of the sample by invoking the function `ahull` (note that the function `ahull` implicitly calculates the α-shape), see Figure 21. Of course, the estimation of the boundary length will depend on the selected parameters, in particular, in the value of α, see Figure 22.

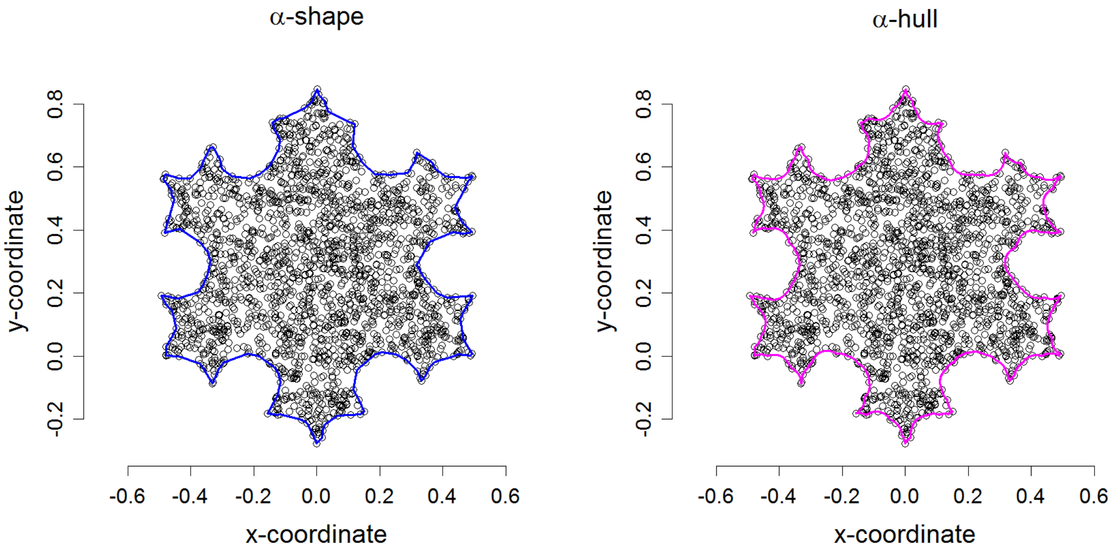

Figure 21: Random sample of size $n = 2000$ from a uniform distribution on a Koch snowflake with initial side length 1 and 3 iterations. Left: α-shape for $\alpha = 0.05$. Right: α-convex hull for $\alpha = 0.05$.

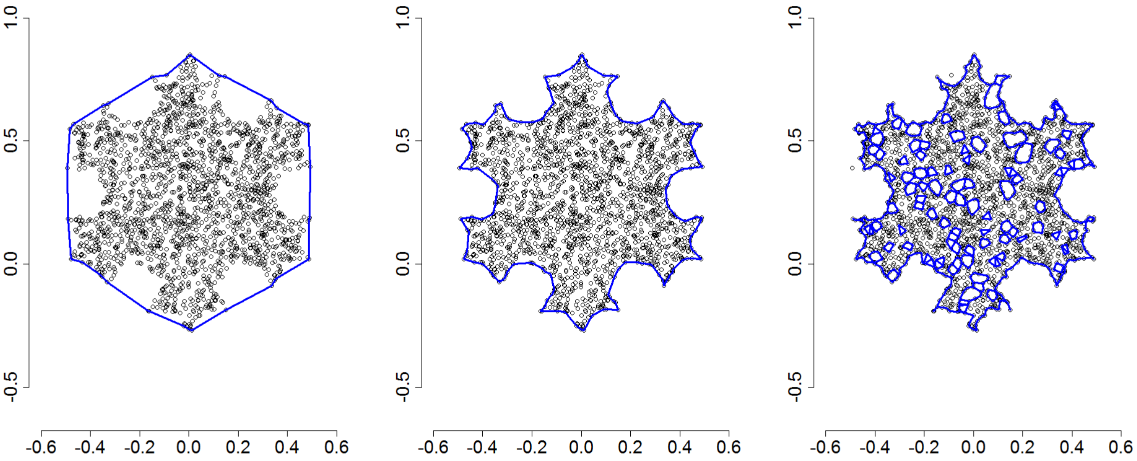

Figure 22: Random sample of size $n = 2000$ from a uniform distribution on a Koch snowflake with initial side length 1 and 3 iterations. From left to right, α-shape for $\alpha = 0.15, 0.05, 0.02$.

```
R> unifkoch <- rkoch(2000, side = 1, niter = 3)
R> alpha = 0.05
R> alphahull <- ahull(unifkoch, alpha = alpha)
```

In order to study this, we have generated 500 uniform samples of size $n = 2000$ on the first

	α	0.01	0.03	0.05	0.07	0.09	0.11
Iteration $k = 1$ ($L = 3$)							
	Average	2.9322	3.0296	2.9610	2.9439	2.9364	2.9321
	Std. deviation	0.0221	0.0454	0.0240	0.0226	0.0223	0.0221
	Median	2.9338	3.0251	2.9621	2.9449	2.9377	2.9337
Iteration $k = 2$ ($L = 4$)							
	Average	3.7582	4.1563	3.8678	3.8173	3.7842	3.7582
	Std. deviation	0.0412	0.1606	0.0442	0.0420	0.0412	0.0412
	Median	3.7604	4.1332	3.8677	3.8164	3.7855	3.7604
Iteration $k = 3$ ($L = 5.333$)							
	Average	3.4962	5.4180	4.7710	4.6239	4.4936	3.4962
	Std. deviation	0.0930	0.2688	0.0738	0.0719	0.0678	0.0930
	Median	3.4791	5.3963	4.7755	4.6272	4.4991	3.4791
Iteration $k = 4$ ($L = 7.111$)							
	Average	3.4752	6.0907	4.3594	4.1746	3.7671	3.4752
	Std. deviation	0.0367	0.3167	0.0805	0.0814	0.1021	0.0367
	Median	3.4742	6.0629	4.3585	4.1721	3.7586	3.4742

Table 6: Average, standard deviation and median of the length of $\alpha^{\text{shape}}(\mathcal{X}_n)$, based on 500 uniform samples $\mathcal{X}_n$ with sample size $n = 2000$ on a Koch snowflake with initial side length 1 and k iterations. The lengths of the first four iterations of a Koch snowflake curve with initial side length 1 are $L = 3, 4, 5.333$ and 7.111, respectively.

four iterations of a Koch snowflake with initial side length 1. For each sample $\mathcal{X}_n$ we construct the α-shape. The length of the Koch snowflake curve is estimated through the length of the α-shape. Different values of α have been considered in order to study the influence of this parameter on the estimations. The results are summarized in Table 6.

As expected, the boundary length estimation becomes more biased as the complexity of the Koch snowflake increases. In the convex case, $k = 1$, the estimator works reasonably well for every value of α. However, for $k = 2$ and $k = 3$, the influence of α is more obvious. As in the previous simulation study, when α is too large the estimator is not longer able to identify the cavities of the set. On the other hand, small values of alpha can reduce the bias, even though the variance of the estimator increases, see for instance the case $\alpha = 0.03$ and $k = 3$. Lastly, as k increases ($k \geq 4$), the estimator has more difficulty in capturing the numerous twists of the curve even for small values of α until it finally breaks down, see Figure 23.

5. Computational details

The results in this paper were obtained using R 2.9.2 (R Development Core Team 2008). The **alphahull** package is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=alphahull`. This article describes version 0.1-1 of the package.

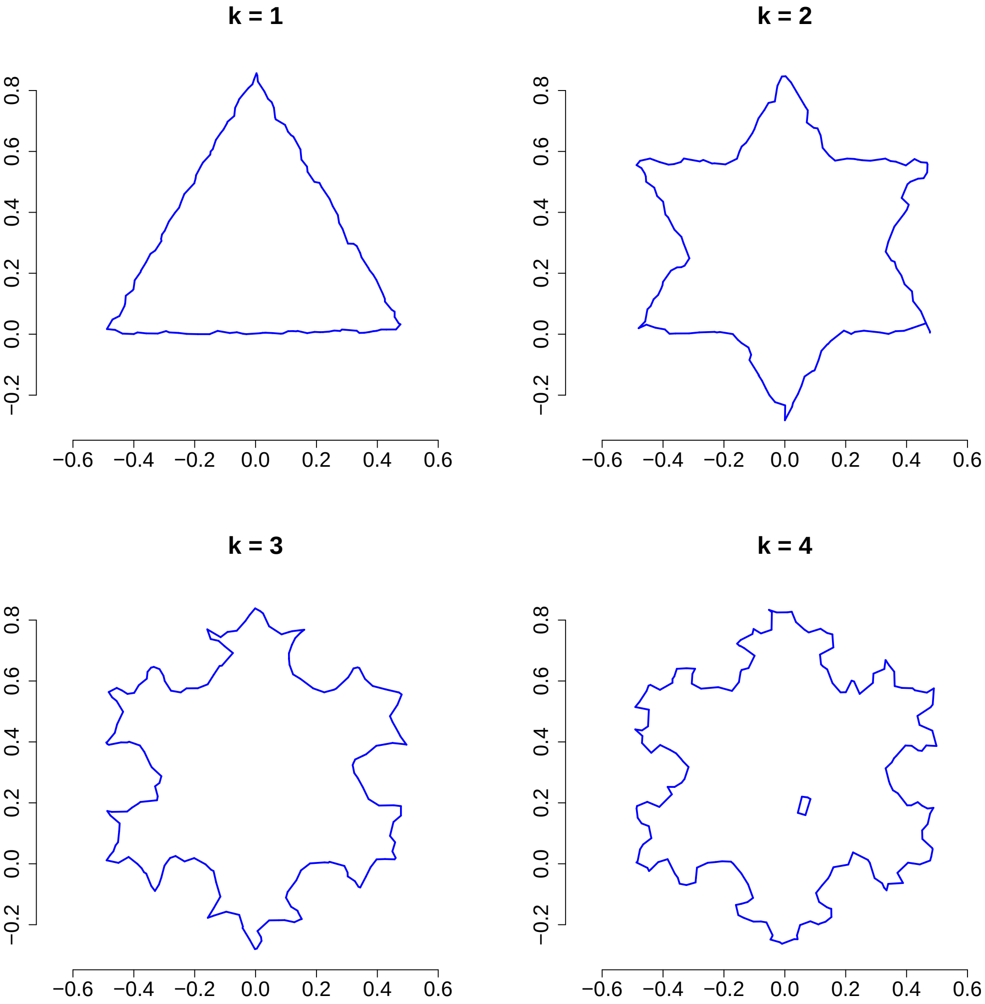

Figure 23: In blue, α-shape for $\alpha = 0.03$ of a random sample of size $n = 2000$ from a uniform distribution on a Koch snowflake with initial side length 1 and k iterations.

6. Conclusions

In this paper we have described the implementation in R of the α-convex hull and of the α-shape of a random sample of points in the plane. The package **alphahull** is the result of that implementation, which is based on efficient algorithms presented by Edelsbrunner *et al.* (1983). The α-convex hull and the α-shape generalize the notion of convex hull and serve to characterize the shape of an unknown set based on random sample of points taken into it. The results obtained by Rodríguez-Casal (2007) on the behaviour of the α-convex hull estimator give us the theoretical basis for using this geometrical construct as a support estimator. Apart from the support estimation problem, we think that these structures can play an important role in other interesting statistical problems which involve the notion of the shape of a point cloud such as cluster analysis and data depth. As an example of these potential applications of the proposed algorithms two small simulation studies on boundary length estimation has been carried out in this work. The results inspire us to address in the future this problem from the theoretical viewpoint.

Acknowledgments

This work was supported by grants MTM2008-03010 from the Ministerio de Ciencia e Innovación, PGIDIT06PXIB207009PR from the Xunta de Galicia, and the IAP research network grant no. P6/03 from the Belgian government.

References

Aurenhammer F (1991). "Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure." *ACM Computing Surveys*, **23**(3), 345–405.

Aurenhammer F, Klein R (2000). "Voronoi Diagrams." In *Handbook of Computational Geometry*, pp. 201–290. North-Holland, Amsterdam.

Baddeley A, Jensen EBV (2005). *Stereology for Statisticians*, volume 103 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, Boca Raton, FL.

Bräker H, Hsing T (1998). "On the Area and Perimeter of a Random Convex Hull in a Bounded Convex Set." *Probability Theory and Related Fields*, **111**(4), 517–550.

Chevalier J (1976). "Estimation du support et du contour du support d'une loi de probabilité." *Annales de l'Institut Henri Poincaré (B) Probabilités & Statistiques*, **12**(4), 339–364.

Cruz-Orive LM (2001/02). "Stereology: Meeting Point of Integral Geometry, Probability, and Statistics." *Mathematicae Notae*, **41**, 49–98 (2003). Homage to Luis Santaló. Vol. 1.

Cuevas A, Fraiman R (1997). "A Plug-in Approach to Support Estimation." *The Annals of Statistics*, **25**(6), 2300–2312.

Cuevas A, Fraiman R (2009). *New Perspectives on Stochastic Geometry*, chapter Set Estimation, pp. 366–385. Oxford University Press.

Cuevas A, Fraiman R, Rodríguez-Casal A (2007). "A Nonparametric Approach to the Estimation of Lengths and Surface Areas." *The Annals of Statistics*, **35**(3), 1031–1051.

Delaunay B (1934). "Sur la sphère vide." *Bulletin of the Academy of Sciences of the USSR*, **1934**(6), 793–800.

Devroye L, Wise GL (1980). "Detection of Abnormal Behavior via Nonparametric Estimation of the Support." *SIAM Journal on Applied Mathematics*, **38**(3), 480–488.

Dümbgen L, Walther G (1996). "Rates of Convergence for Random Approximations of Convex Sets." *Advances in Applied Probability*, **28**(2), 384–393.

Edelsbrunner H, Kirkpatrick DG, Seidel R (1983). "On the Shape of a Set of Points in the Plane." *IEEE Transactions on Information Theory*, **29**(4), 551–559.

Edelsbrunner H, Mücke EP (1994). "Three-Dimensional Alpha Shapes." *ACM Transactions on Graphics*, **13**(1), 43–72.

Edgar GA (1990). *Measure, Topology, and Fractal Geometry.* Springer-Verlag, New York.

Fortune S (1987). "A Sweepline Algorithm for Voronoi Diagrams." *Algorithmica*, **2**, 153–174.

Geffroy J (1964). "Sur un problème d'estimation géométrique." *Publications de l'Institut de Statistique de l'Université de Paris*, **13**, 191–210.

Grasman R, Gramacy RB (2008). ***geometry**: Mesh Generation and Surface Tesselation.* R package version 0.1-4, URL `http://CRAN.R-project.org/package=geometry`.

Green P, Sibson R (1978). "Computing Dirichlet Tessellations in the Plane." *The Computer Journal*, **21**, 168–173.

Korostelëv AP, Tsybakov AB (1993). *Minimax Theory of Image Reconstruction*, volume 82 of *Lecture Notes in Statistics*. Springer-Verlag, New York.

Lawson CL (1977). "Software for $\mathcal{C}^1$ Surface Interpolation." In *Mathematical Software III, Proceedings of a Symposium Conducted by the Mathematics Research Center. The University of Wisconsin-Madison.* Academic Press, New York.

Lee DT, Schachter BJ (1980). "Two Algorithms for Constructing a Delaunay Triangulation." *International Journal of Computing and Information Sciences*, **9**, 219–242.

Okabe A, Boots B, Sugihara K, Chiu SN (2000). *Spatial Tessellations. Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, New York.

Pateiro-López B, Rodríguez-Casal A (2009). ***alphahull**: Generalization of the Convex Hull of a Sample of Points in the Plane.* R package version 0.1-1, URL `http://CRAN.R-project.org/package=alphahull`.

Preparata FP, Shamos MI (1985). *Computational Geometry: An Introduction.* Texts and Monographs in Computer Science. Springer-Verlag, New York.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org/`.

Renka RJ (1996). "Algorithm 751: **TRIPACK**: A Constrained Two-Dimensional Delaunay Triangulation Package." *ACM Transactions on Mathematical Software*, **22**(1), 1–8.

Renka RJ, Gebhardt A (2009). ***tripack**: Triangulation of Irregularly Spaced Data.* R package version 1.3-3, URL `http://CRAN.R-project.org/package=tripack`.

Rényi A, Sulanke R (1963). "Über die konvexe Hülle von n zufällig gewählten Punkten." *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, **2**, 75–84.

Rényi A, Sulanke R (1964). "Über die konvexe Hülle von n zufällig gewählten Punkten II." *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, **3**, 138–147.

Rodríguez-Casal A (2007). "Set Estimation under Convexity Type Assumptions." *Annales de l'Institut Henri Poincaré B*, **43**, 763–774.

Schneider R (1988). "Random Approximation of Convex Sets." *Journal of Microscopy*, **151**, 211–227.

Serra J (1984). *Image Analysis and Mathematical Morphology*. Academic Press, London.

Shamos MI, Hoey D (1975). "Closest-Point Problems." In *Proceedings of the 16th IEEE Symposium on Foundations of Computer Science*, pp. 151–162.

Turner R (2009). ***deldir**: Delaunay Triangulation and Dirichlet (Voronoi) Tessellation.* R package version 0.0-8, URL `http://CRAN.R-project.org/package=deldir`.

von Koch H (1904). "Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire." *Arkiv för Matematik*, **1**, 681–704.

Voronoi G (1908). "Nouvelles applications des paramètres continus à la théorie des formes quadratiques." *Journal für die reine und angewandte Mathematik*, **134**, 198–287.

Walther G (1999). "On a Generalization of Blaschke's Rolling Theorem and the Smoothing of Surfaces." *Mathematical Methods in the Applied Sciences*, **22**(4), 301–316.

Affiliation:

Beatriz Pateiro-López. Alberto Rodríguez-Casal
Departamento de Estadística e Investigación Operativa
Facultad de Matemáticas
Rúa Lope Gómez de Marzoa s/n
15782 Santiago de Compostela, Spain
E-mail: `beatriz.pateiro@usc.es`, `alberto.rodriguez.casal@usc.es`

Journal of Statistical Software `http://www.jstatsoft.org/`
published by the American Statistical Association `http://www.amstat.org/`
Volume 34, Issue 5 *Submitted:* 2009-04-03
April 2010 *Accepted:* 2009-07-07