Qhull examples

David C. Sterratt

10th January 2025

This document presents examples of the `geometry` package functions which implement functions using the Qhull library.

1 Convex hulls in 2D

1.1 Calling `convhulln` with one argument

With one argument, convhulln returns the indices of the points of the convex hull.

```
> library(geometry)
> ps <-matrix(rnorm(30), , 2)
> ch <- convhulln(ps)
> head(ch)
```

```
     [,1] [,2]
[1,]    4    8
[2,]    4    2
[3,]    1    8
[4,]   11    2
[5,]   11    1
```

1.2 Calling `convhulln` with `options`

We can supply Qhull options to `convhulln`; in this case it returns an object of class `convhulln` which is also a list. For example `FA` returns the generalised `area` and

`volume`. Confusingly in 2D the generalised area is the length of the perimeter, and the generalised volume is the area.

```
> ps <-matrix(rnorm(30), , 2)
> ch <- convhulln(ps, options="FA")
> print(ch$area)
```

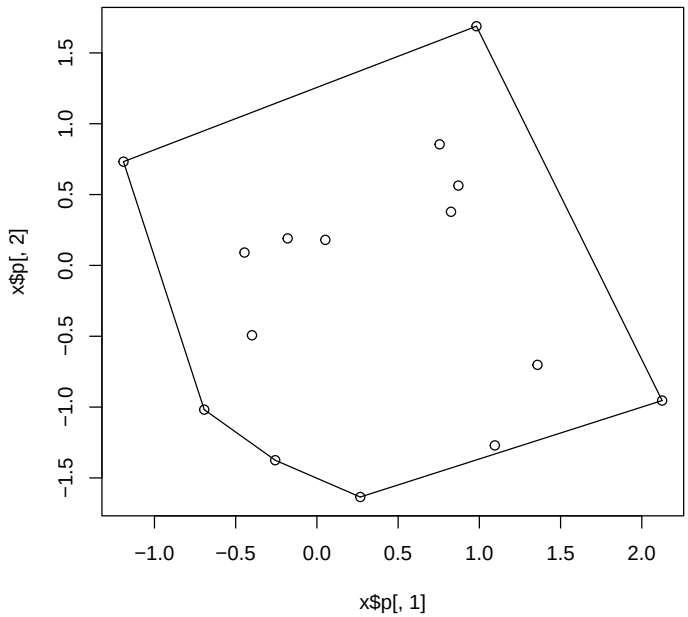
```
[1] 10.19971
```

```
> print(ch$vol)
```

```
[1] 6.836075
```

A `convhulln` object can also be plotted.

```
> plot(ch)
```



We can also find the normals to the "facets" of the convex hull:

```
> ch <- convhulln(ps, options="n")
> head(ch$normals)
```

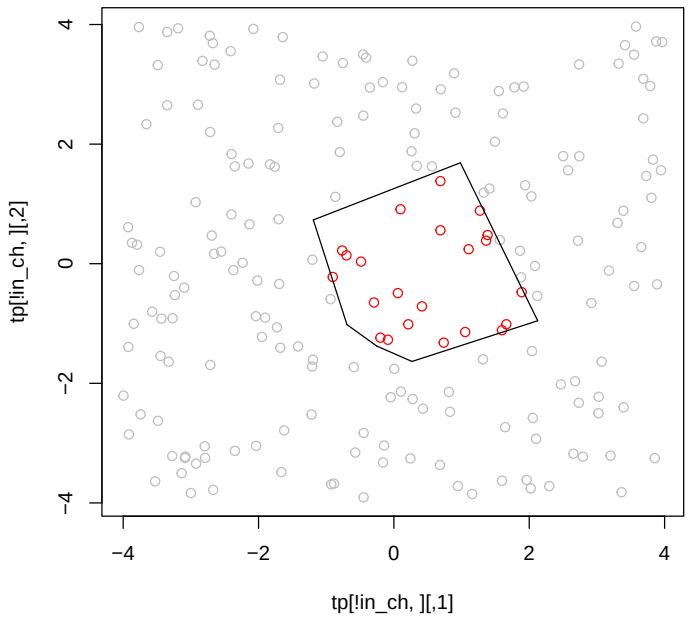
```
           [,1]       [,2]       [,3]
[1,] -0.4026540  0.9153522 -1.1496977
[2,]  0.9177743  0.3971024 -1.5716350
[3,] -0.9619445 -0.2732449 -0.9453899
[4,] -0.6328092 -0.7743078 -1.2271540
[5,]  0.3439975 -0.9389706 -1.6268302
[6,] -0.4442238 -0.8959158 -1.3459486
```

Here the first two columns and the x and y direction of the normal, and the third column defines the position at which the face intersects that normal.

1.3 Testing if points are inside a convex hull with `inhulln`

The function `inhulln` can be used to test if points are inside a convex hull. Here the function `rbox` is a handy way to create points at random locations.

```
> tp <- rbox(n=200, D=2, B=4)
> in_ch <- inhulln(ch, tp)
> plot(tp[!in_ch,], col="gray")
> points(tp[in_ch,], col="red")
> plot(ch, add=TRUE)
```



2 Delaunay triangulation in 2D

2.1 Calling `delaunayn` with one argument

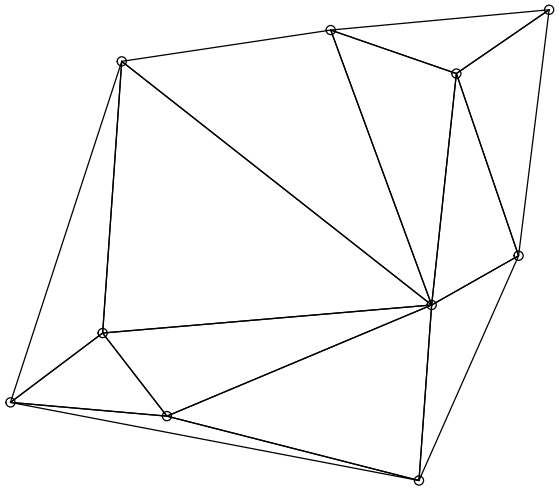
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

```
> ps <- rbox(n=10, D=2)
> dt <- delaunayn(ps)
> head(dt)
```

```
     [,1] [,2] [,3]
[1,]    7    3    5
```

```
[2,]    7    8    3
[3,]    6    8    3
[4,]   10    2    8
[5,]    9    2    8
[6,]    9    7    8
```

```
> trimesh(dt, ps)
> points(ps)
```



2.2 Calling `delaunayn` with `options`

We can supply Qhull options to `delaunayn`; in this case it returns an object of class `delaunayn` which is also a list. For example `Fa` returns the generalised `area` of each triangle. In 2D the generalised area is the actual area; in 3D it would be the volume.

```
> dt2 <- delaunayn(ps, options="Fa")
> print(dt2$areas)
```

```
 [1] 0.01011534 0.06898766 0.02239805 0.09277838 0.13602945 0.04461859
 [7] 0.03630432 0.01854963 0.02892956 0.04617745 0.03193573 0.01844089
```

```
> dt2 <- delaunayn(ps, options="Fn")
> print(dt2$neighbours)
```

```
[[1]]
[1] -1  8  2

[[2]]
[1] 3 1 6

[[3]]
[1]   2 -12   9

[[4]]
[1]   5  10 -15

[[5]]
[1] 4 6 7

[[6]]
[1] 2 5 8

[[7]]
[1] -5  8  5

[[8]]
[1] 1 7 6

[[9]]
[1]  3 10 11

[[10]]
[1]  4  9 12

[[11]]
[1] -12  12   9

[[12]]
[1] -15  11  10
```