Qhull examples

David C. Sterratt

31st August 2024

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   13
[2,]    2    8
[3,]   12    8
[4,]   12    4
[5,]   10   13
[6,]   10    2
```

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)
```

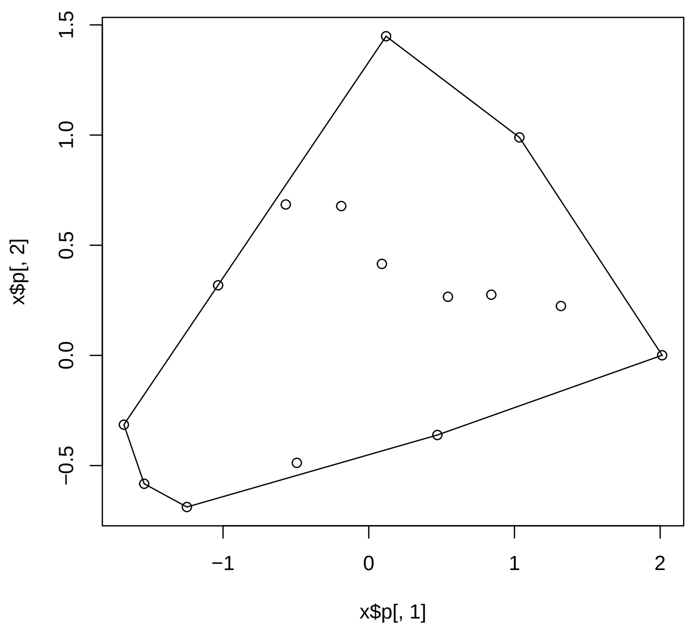
```
[1] 8.88303
```

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

```
[1] 4.049505
```

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)
```

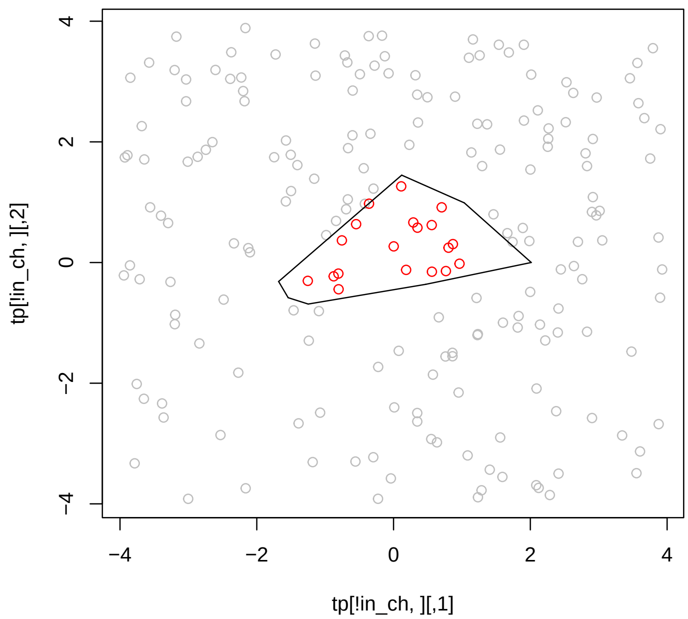
```
           [,1]       [,2]       [,3]
[1,] -0.6997701  0.7143681 -0.9514759
[2,]  0.7103306  0.7038682 -1.4307640
[3,]  0.4485676  0.8937489 -1.3480067
[4,] -0.8872996 -0.4611935 -1.6362805
[5,] -0.3385438 -0.9409506 -1.0699777
[6,]  0.2283874 -0.9735703 -0.4592712
```

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

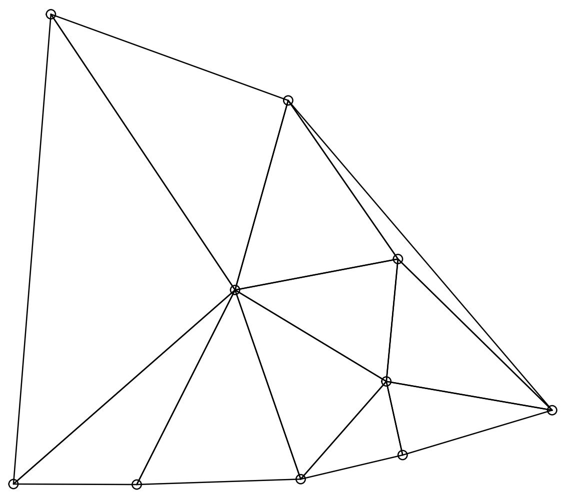
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]    3    9    4
```

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

```
> 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] 0.12375960 0.06332250 0.01002852 0.03735478 0.03071743 0.04008693
 [7] 0.02890286 0.02502634 0.02636622 0.01500031 0.01009191
```

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

```
[[1]]
[1] -1  5  2

[[2]]
[1]  1 -5  4

[[3]]
[1] -5  9  4

[[4]]
[1] 2 8 3

[[5]]
[1]   1 -16   6

[[6]]
[1]   7   5 -16

[[7]]
[1]  6  8 11

[[8]]
[1] 4 7 9

[[9]]
[1]  3 10  8

[[10]]
[1]   9 -22  11

[[11]]
[1]   7 -22  10
```