Qhull examples

David C. Sterratt

8th February 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,]    1   13
[2,]    9   13
[3,]    9   11
[4,]    3   11
[5,]    3    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)
```

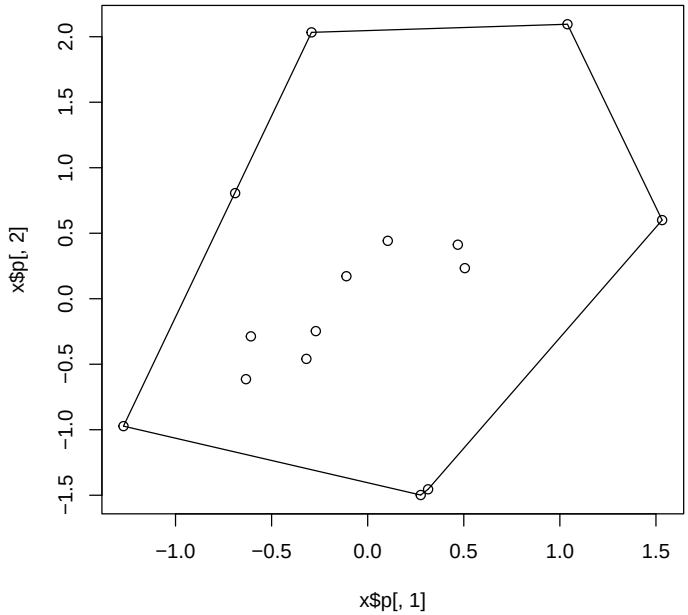
```
[1] 10.14886
```

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

```
[1] 6.420074
```

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)
```

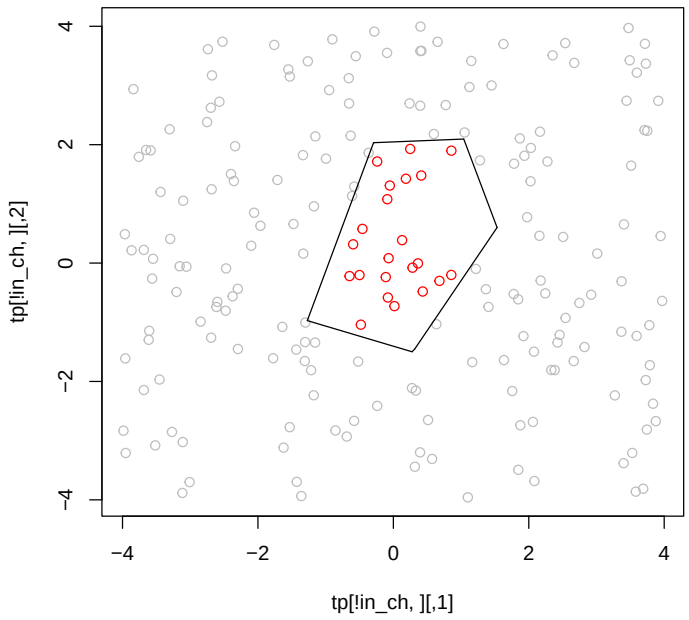
```
            [,1]       [,2]       [,3]
[1,]  0.94977482  0.3129342 -1.6430294
[2,] -0.32183281 -0.9467965 -1.3297490
[3,] -0.95089011  0.3095287 -0.9074194
[4,] -0.04639363  0.9989232 -2.0447076
[5,]  0.86026294 -0.5098506 -1.0119888
[6,]  0.75037612 -0.6610111 -1.1973350
```

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

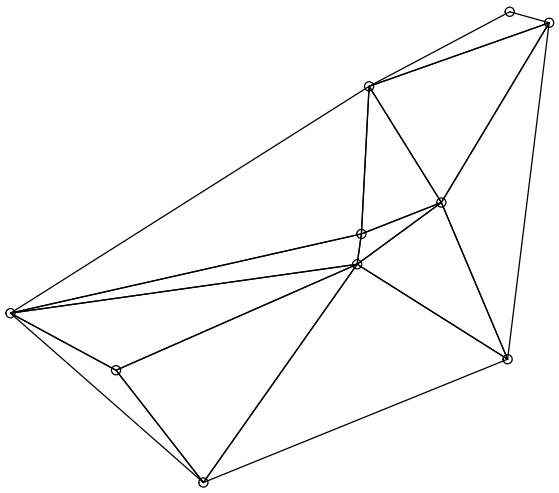
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]   10    4    2
```

```
[2,]    6    2    1
[3,]    6    4    1
[4,]    6    4    2
[5,]    9    7    5
[6,]    3    7    5
```

```
> 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] 0.073651761 0.010656171 0.038743285 0.056404057 0.006975020 0.039601036
 [7] 0.044695056 0.026844220 0.003559608 0.017953958 0.016004616 0.079475290
```

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

```
[[1]]
[1]  4 -5  8

[[2]]
[1] -1  3  4

[[3]]
[1] 11  2  4

[[4]]
[1] 1 2 3

[[5]]
[1]   6 -14 -15

[[6]]
[1]  5  7 10

[[7]]
[1] -5  6  8

[[8]]
[1] 1 9 7

[[9]]
[1]  8 11 10

[[10]]
[1]  6 12  9

[[11]]
[1]  3 12  9

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