Qhull examples

David C. Sterratt

3rd February 2023

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    2
[2,]   11    5
[3,]   14    5
[4,]   14    1
[5,]   13    2
[6,]   13   11
```

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)
```

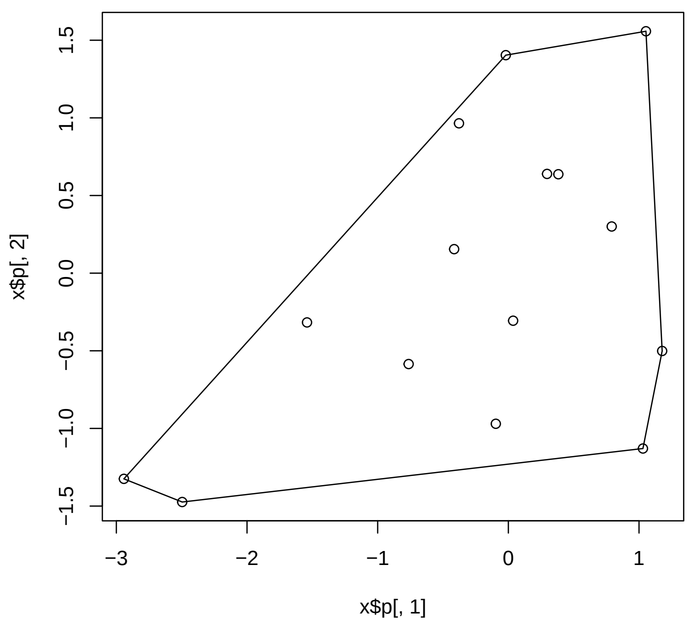
```
[1] 11.80321
```

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

```
[1] 7.103391
```

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)
```

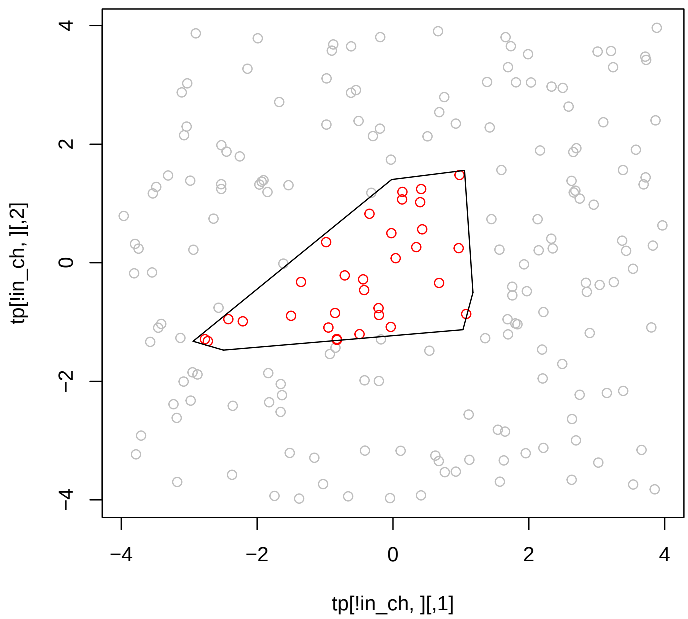
```
            [,1]        [,2]      [,3]
[1,]  0.99819266  0.06009509 -1.144762
[2,]  0.97368603 -0.22789366 -1.260279
[3,] -0.68240236  0.73097676 -1.039582
[4,] -0.14208083  0.98985506 -1.392326
[5,] -0.31701912 -0.94841915 -2.189101
[6,]  0.09725278 -0.99525971 -1.224117
```

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

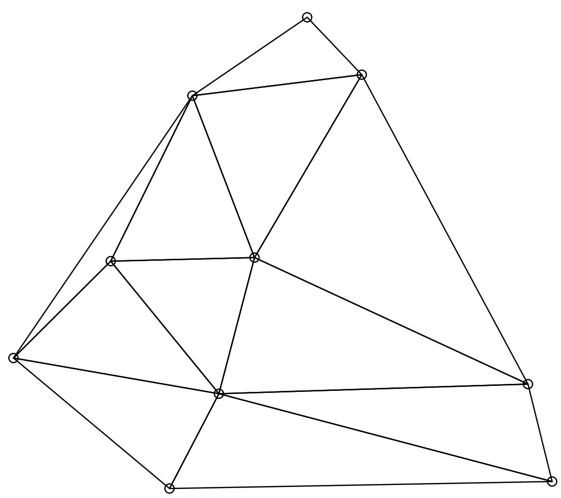
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

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

```
[2,]    2    4   10
[3,]    8    7   10
[4,]    8    4    6
[5,]    8    4   10
[6,]    1    7   10
```

```
> 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] 0.01022089 0.02935631 0.07943093 0.01357906 0.03587993 0.05213991
 [7] 0.02429860 0.04488249 0.03790367 0.02653477 0.02919624
```

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

```
[[1]]
[1] -8 11  2

[[2]]
[1] 5 7 1

[[3]]
[1]   6   5 -23

[[4]]
[1]  -8 -22   5

[[5]]
[1] 2 3 4

[[6]]
[1] 3 7 9

[[7]]
[1]  2  6 11

[[8]]
[1] -5  9 10

[[9]]
[1] -23   8   6

[[10]]
[1] -5 11  8

[[11]]
[1]  1 10  7
```