Qhull examples

David C. Sterratt

3rd December 2019

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,]   10   13
[2,]   15    6
[3,]   15   10
[4,]   11    6
[5,]   11    4
[6,]    1   13
```

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)
```

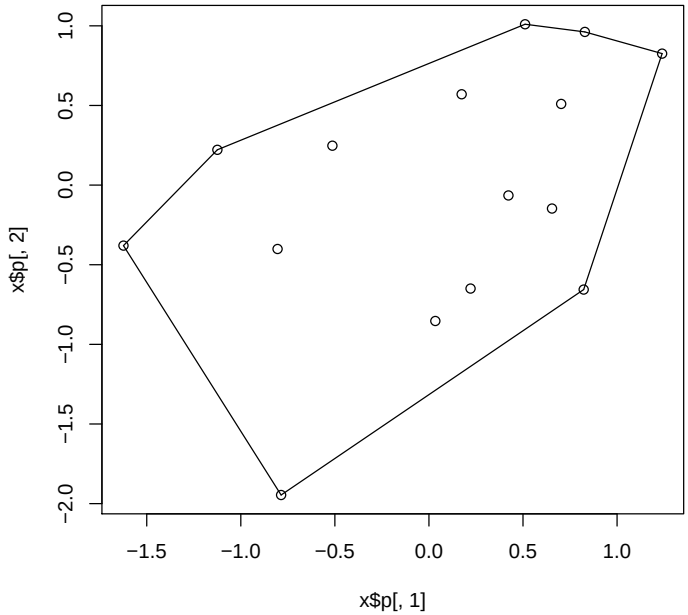
```
[1] 8.728395
```

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

```
[1] 4.679497
```

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)
```

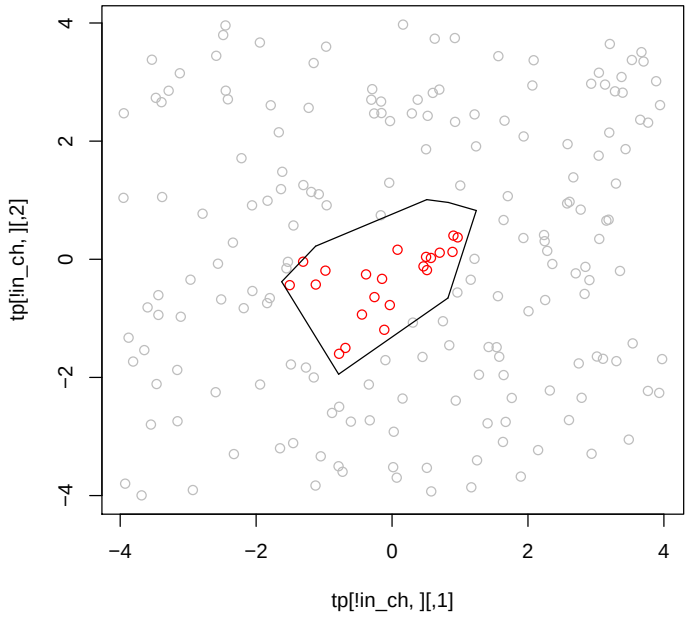
```
           [,1]       [,2]       [,3]
[1,] -0.8817106 -0.4717907 -1.6108794
[2,]  0.9626036 -0.2709140 -0.9695298
[3,]  0.6256402 -0.7801118 -1.0263535
[4,] -0.7696230  0.6384986 -1.0070657
[5,] -0.4341306  0.9008499 -0.6879394
[6,]  0.3146140  0.9492197 -1.1737152
```

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

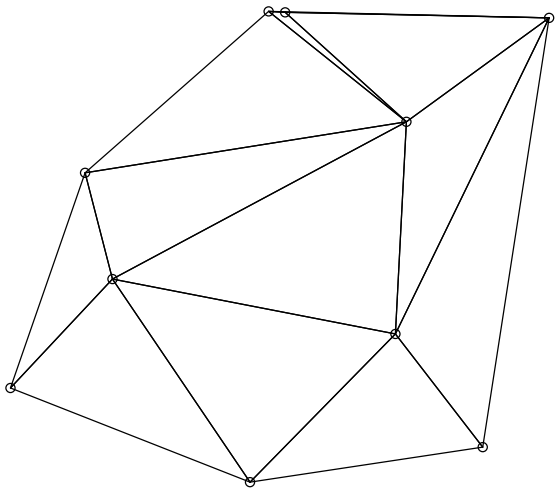
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

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

```
[2,]   10    1    7
[3,]    5    8    9
[4,]    5    7    9
[5,]    5    1    7
[6,]    6    1    2
```

```
> 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] 0.0444646876 0.0621703536 0.0560487301 0.0366148609 0.0172143239
 [6] 0.0529475921 0.0362908235 0.0755227712 0.0442629053 0.0001979219
[11] 0.0351576549 0.0021084158
```

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

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

[[2]]
[1] 1 4 8

[[3]]
[1]   7 -12   4

[[4]]
[1]   2 -12   3

[[5]]
[1]   1 -16   9

[[6]]
[1] -16  12   9

[[7]]
[1]  3 11  8

[[8]]
[1] 2 9 7

[[9]]
[1] 5 8 6

[[10]]
[1] -8 11 12

[[11]]
[1]  7 10 12

[[12]]
[1]  6 10 11
```