Qhull examples

David C. Sterratt

18th April 2022

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,]    9   14
[2,]    3    9
[3,]    6   10
[4,]    8   10
[5,]    8    3
[6,]    4   14
```

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)
```

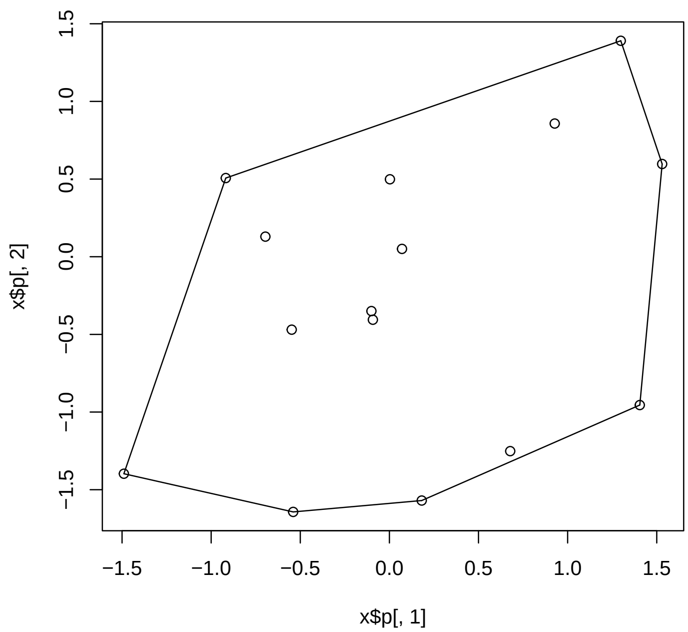
```
[1] 9.832655
```

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

```
[1] 6.247398
```

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)
```

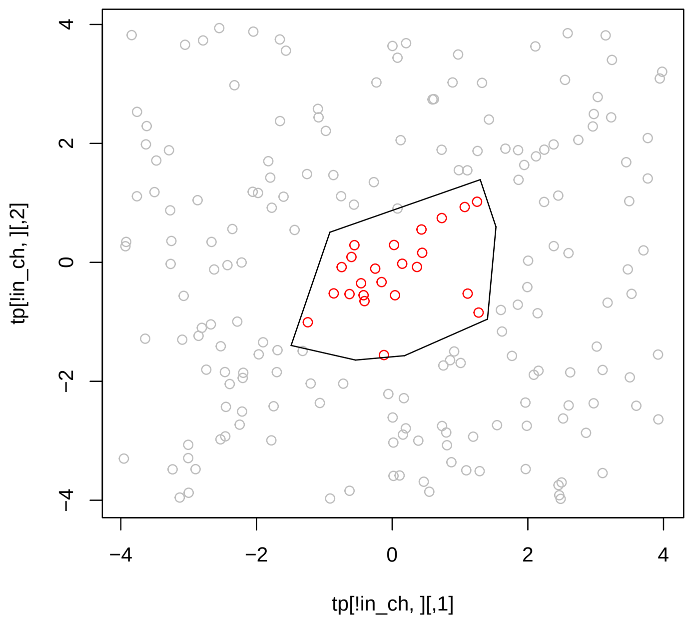
```
           [,1]       [,2]       [,3]
[1,]  0.9597518  0.2808495 -1.6364365
[2,]  0.9967519 -0.0805339 -1.4772996
[3,] -0.9577287  0.2876728 -1.0248869
[4,] -0.3704233  0.9288631 -0.8105545
[5,]  0.4489465 -0.8935586 -1.4840538
[6,] -0.2505589 -0.9681014 -1.7257887
```

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

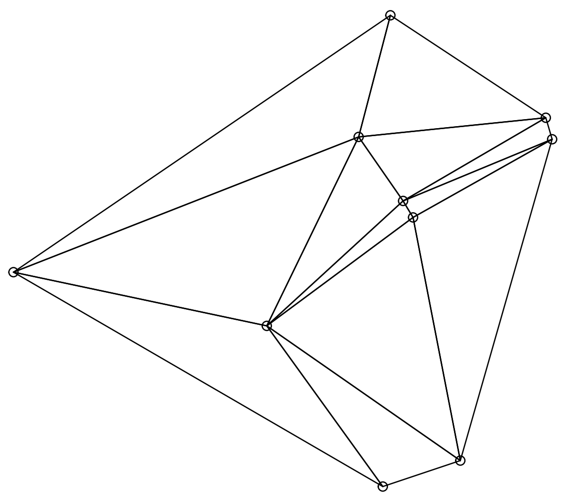
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]   10    4    7
```

```
[2,]   10    2    7
[3,]    8    2    7
[4,]    3    9    6
[5,]    3   10    4
[6,]    3    9   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] 0.047376344 0.072557640 0.021256899 0.051788429 0.030449139 0.055917155
 [7] 0.051559413 0.019621224 0.004796023 0.017638336 0.004938398 0.004219893
```

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

```
[[1]]
[1] -10   2   3

[[2]]
[1] 4 1 8

[[3]]
[1]   1 -10   6

[[4]]
[1]   2 -19   5

[[5]]
[1]   4  10 -19

[[6]]
[1] 3 9 7

[[7]]
[1] -18  12   6

[[8]]
[1]  2 10  9

[[9]]
[1]  6  8 12

[[10]]
[1]  5  8 11

[[11]]
[1] -18  12  10

[[12]]
[1]  7 11  9
```