Qhull examples

David C. Sterratt

18th February 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,]    6    2
[2,]    4    2
[3,]   14    8
[4,]   14    6
[5,]    9    8
[6,]    9    4
```

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)
```

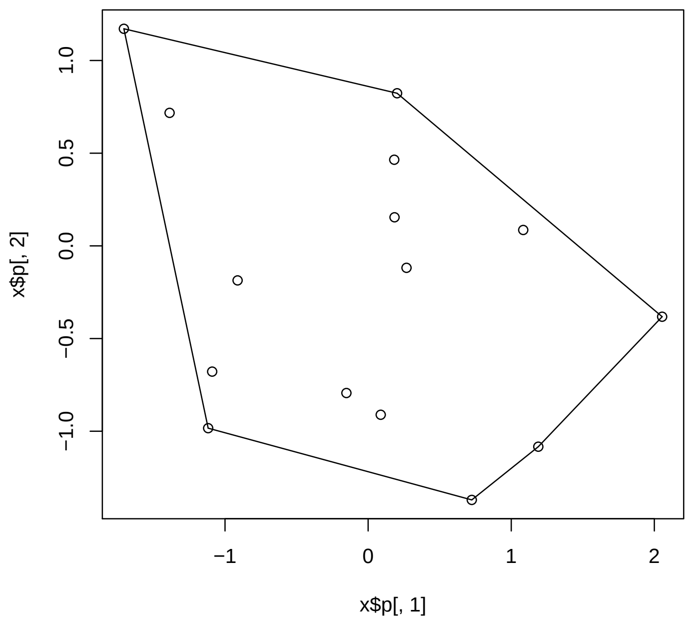
```
[1] 9.926696
```

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

```
[1] 5.6298
```

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)
```

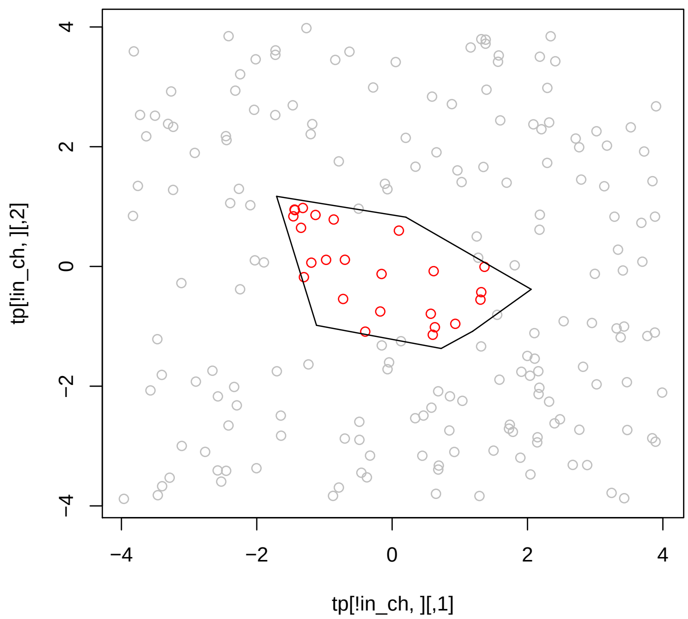
```
           [,1]       [,2]       [,3]
[1,] -0.9646147 -0.2636636 -1.3375580
[2,] -0.2056030 -0.9786355 -1.1922798
[3,]  0.1792508  0.9838034 -0.8462883
[4,]  0.5454867  0.8381195 -0.8004543
[5,]  0.6295677 -0.7769456 -1.5903763
[6,]  0.5254235 -0.8508408 -1.5465868
```

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

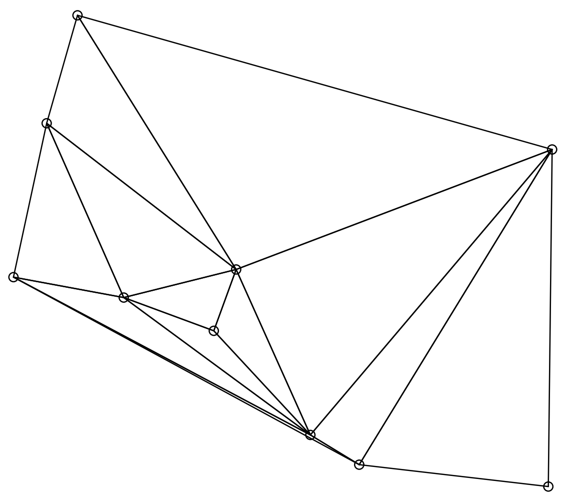
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]    6    8    9
```

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

```
> 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] 0.162675426 0.104496237 0.040860826 0.100195676 0.034601944 0.013507354
 [7] 0.001831417 0.028891631 0.035621975 0.010260715 0.018561721 0.010097123
```

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

```
[[1]]
[1] -4  4  3

[[2]]
[1]  -4   5 -15

[[3]]
[1]   1 -17   9

[[4]]
[1] 1 5 6

[[5]]
[1] 2 4 7

[[6]]
[1] 10  4 12

[[7]]
[1] -15  11   5

[[8]]
[1] -17  11   9

[[9]]
[1]  3 10  8

[[10]]
[1]  6  9 12

[[11]]
[1]  7  8 12

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