Qhull examples

David C. Sterratt

12th July 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,]   12    5
[2,]    8   12
[3,]    2    5
[4,]    2   14
[5,]   10   15
[6,]   10   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)
```

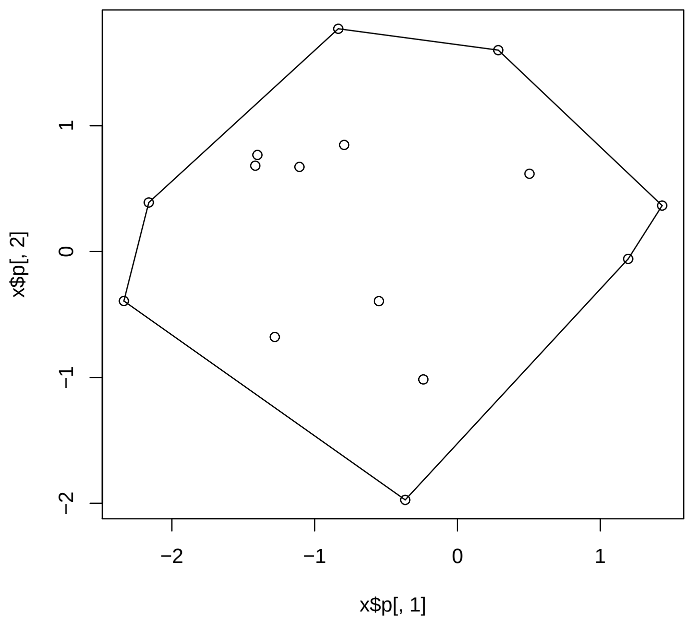
```
[1] 11.01695
```

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

```
[1] 8.327517
```

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)
```

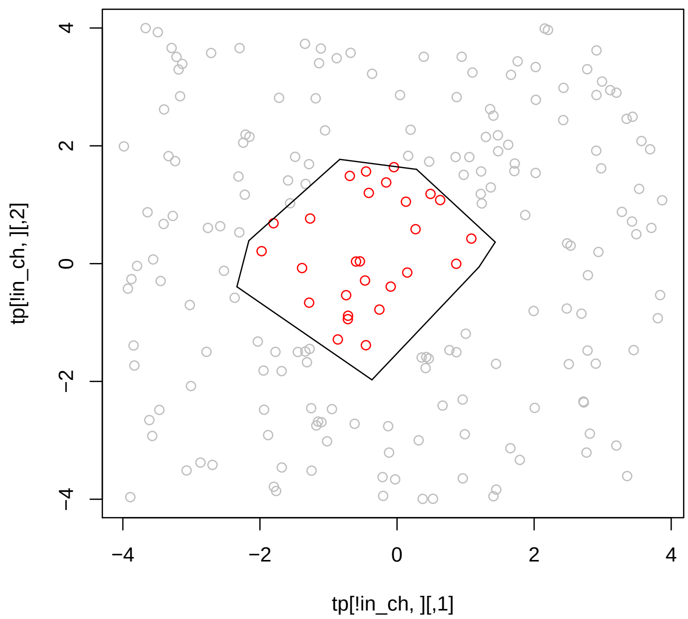
```
           [,1]       [,2]       [,3]
[1,] -0.6257266 -0.7800424 -1.7681379
[2,]  0.8720029 -0.4895007 -1.0705327
[3,]  0.7750403 -0.6319118 -0.9629537
[4,] -0.9760148  0.2177044 -2.1947452
[5,] -0.7208602  0.6930804 -1.8286042
[6,]  0.7324836  0.6807847 -1.2983860
```

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

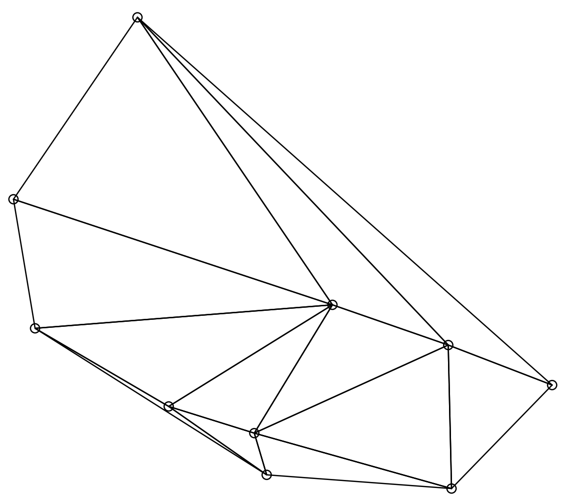
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

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

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

```
> 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] 0.067185309 0.012344612 0.020411384 0.024045606 0.017002317 0.013959085
 [7] 0.026566948 0.036736190 0.024904719 0.007122402 0.003058828 0.001409405
```

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

```
[[1]]
[1] -24   8   4

[[2]]
[1]  9 11  5

[[3]]
[1] -4  6  4

[[4]]
[1] 1 3 5

[[5]]
[1] 2 4 7

[[6]]
[1] -4  3  7

[[7]]
[1] 10  6  5

[[8]]
[1]   1 -24   9

[[9]]
[1]  2 12  8

[[10]]
[1]   7 -30  11

[[11]]
[1]  2 12 10

[[12]]
[1]   9  11 -31
```