Qhull examples

David C. Sterratt

21st August 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   13
[2,]    6    5
[3,]   12   13
[4,]    7    5
[5,]    7   12
```

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)
```

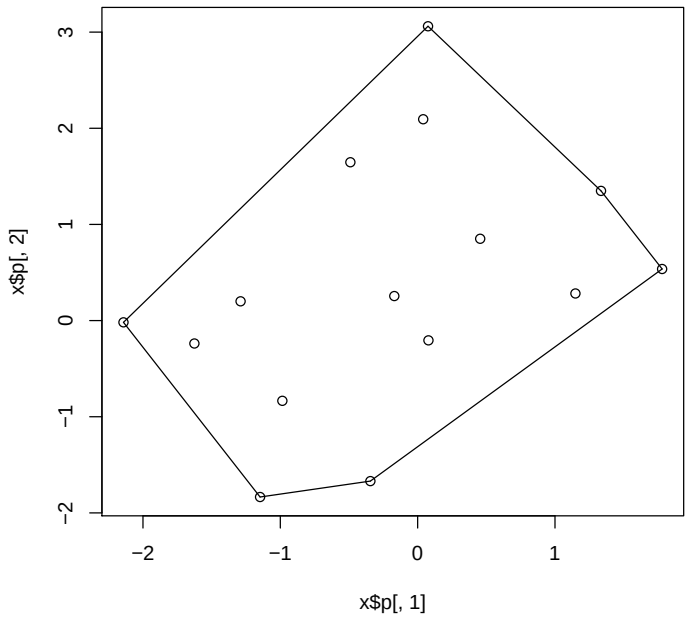
```
[1] 12.79967
```

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

```
[1] 10.10213
```

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)
```

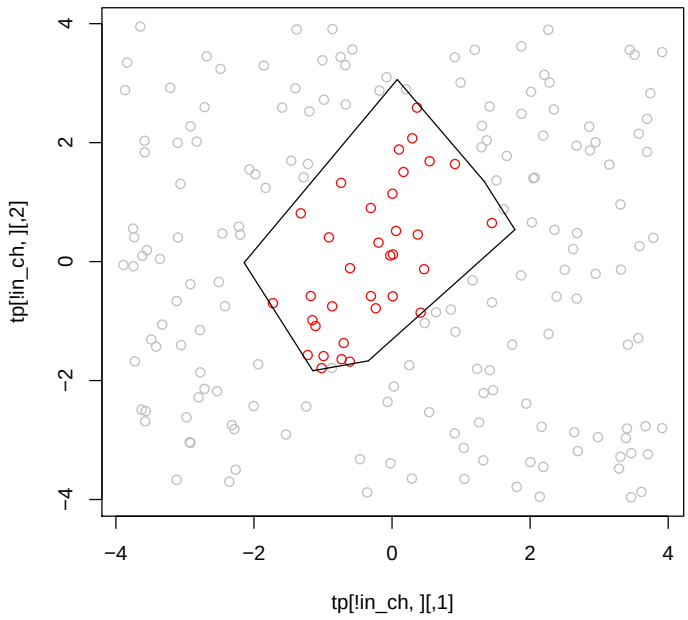
```
           [,1]       [,2]       [,3]
[1,] -0.8114663  0.5843992 -1.7275682
[2,] -0.8771429 -0.4802295 -1.8874797
[3,]  0.8769988  0.4804926 -1.8189454
[4,]  0.8055050  0.5925889 -1.8746992
[5,]  0.7203090 -0.6936533 -0.9099997
[6,]  0.2017606 -0.9794349 -1.5655190
```

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

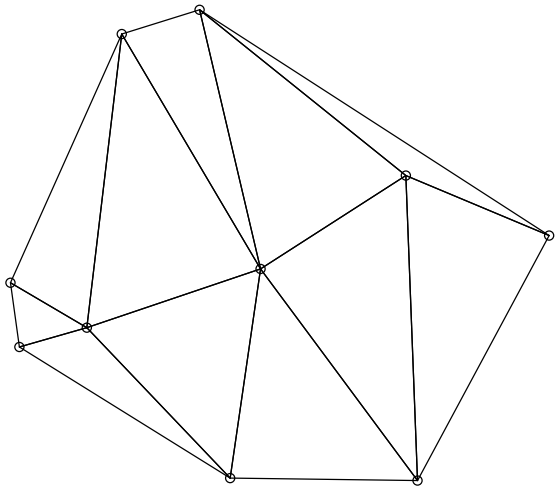
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]    1    5    8
```

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

```
> 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] 0.027952244 0.014686765 0.055981036 0.055535554 0.058641308 0.050707906
 [7] 0.044630166 0.016678190 0.005817549 0.030981940 0.063236248
```

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

```
[[1]]
[1]  3 -9 11

[[2]]
[1] -9  4  3

[[3]]
[1] 1 2 5

[[4]]
[1] -21   2   5

[[5]]
[1] 6 3 4

[[6]]
[1]   5   7 -22

[[7]]
[1]  6 11  8

[[8]]
[1] -22   7   9

[[9]]
[1] -19  10   8

[[10]]
[1] -19   9  11

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