# Qhull examples

David C. Sterratt

11th March 2025

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,]    2    5
[2,]    3    5
[3,]   13    4
[4,]   13    2
[5,]   15    4
[6,]   15    3
```

### 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)
```

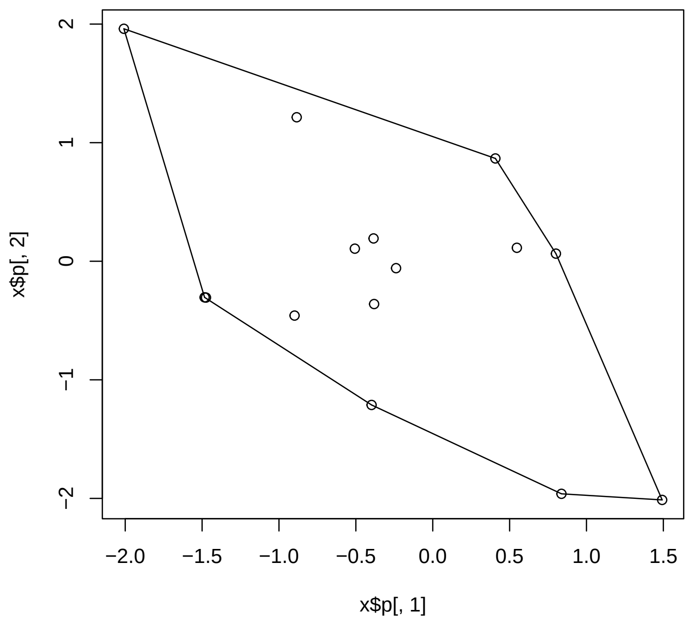
```
[1] 11.57665
```

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

```
[1] 6.575656
```

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)
```

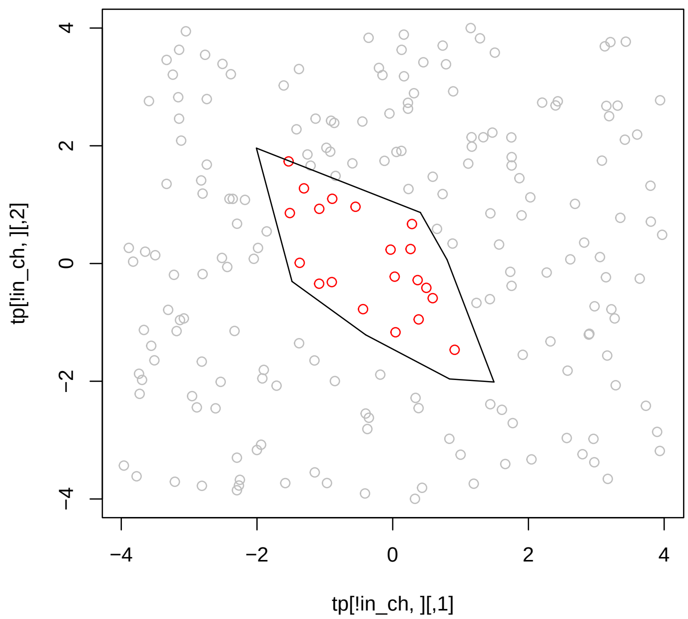
```
            [,1]       [,2]       [,3]
[1,] -0.07811689 -0.9969442 -1.8899581
[2,]  0.41227935  0.9110575 -0.9578206
[3,] -0.97418727 -0.2257414 -1.5143694
[4,]  0.94887476  0.3156528 -0.7804965
[5,]  0.89781426  0.4403743 -0.7476058
[6,] -0.51880629 -0.8548918 -1.2423934
```

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

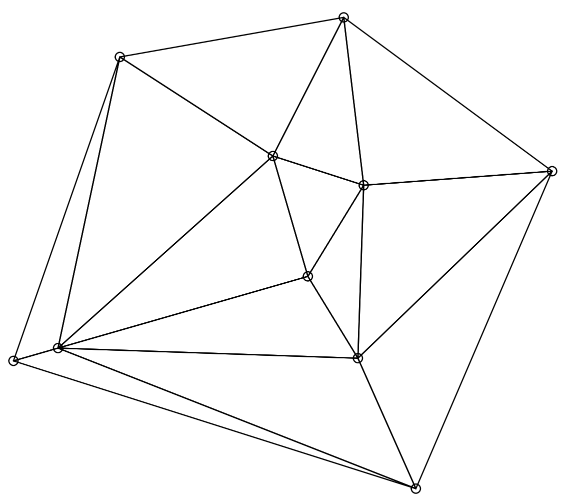
With one argument, a set of points, delaunayn returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]    6    5    3
```

```
[2,]    9    2    1
[3,]    9    6   10
[4,]    9    5    1
[5,]    9    6    5
[6,]    8    7    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] 0.04570371 0.01539041 0.03035940 0.01369283 0.04868785 0.03567756
 [7] 0.04116661 0.06403851 0.04031994 0.01851921 0.01251536 0.04107368
[13] 0.01152994
```

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

[[1]]
[1] -1 12  5

[[2]]
[1] -5  4  8

[[3]]
[1] 13  7  5

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

[[5]]
[1] 1 4 3

[[6]]
[1] -18   8  10

[[7]]
[1]  3 11  8

[[8]]
[1] 2 6 7

[[9]]
[1] -18  12  10

[[10]]
[1]  6  9 11

[[11]]
[1]  7 13 10

[[12]]
[1]  1  9 13

[[13]]
[1]  3 11 12
```