Qhull examples

David C. Sterratt

27th July 2024

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,]   15    5
[2,]   10    5
[3,]   14   11
[4,]   14   15
[5,]    1   11
[6,]    1   10
```

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)
```

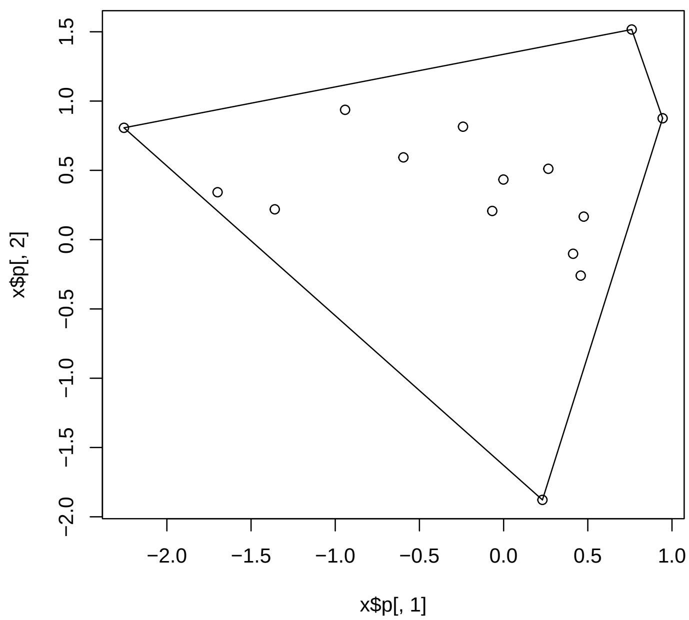
```
[1] 10.2679
```

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

```
[1] 5.41154
```

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)
```

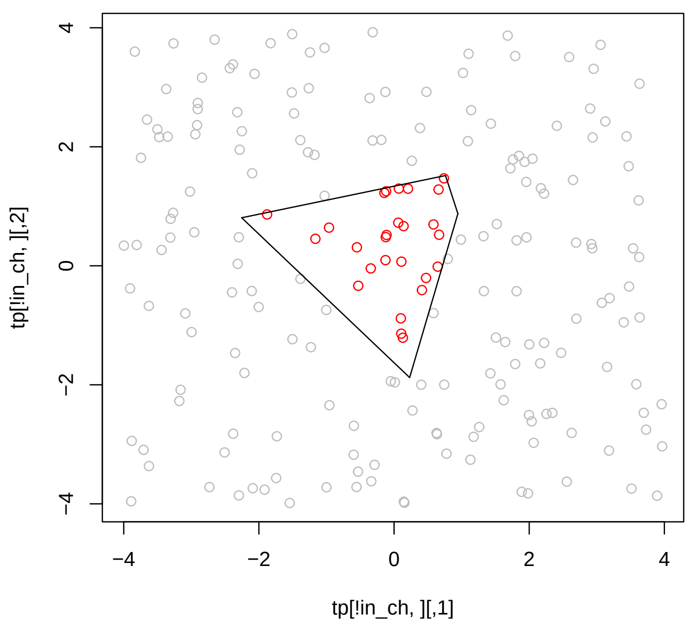
```
           [,1]       [,2]       [,3]
[1,] -0.7338407 -0.6793216 -1.1064722
[2,]  0.9680036 -0.2509365 -0.6944692
[3,] -0.2290014  0.9734261 -1.3019190
[4,]  0.9612910  0.2755353 -1.1492274
```

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

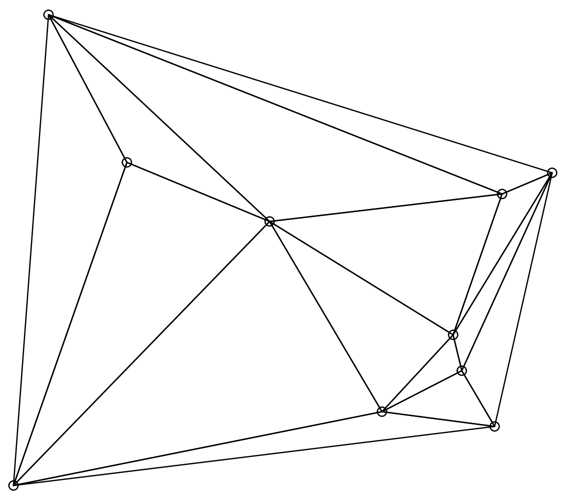
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

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

```
[5,]    1    7    2
[6,]    1    5    9
```

```
> 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] 0.067265127 0.084429276 0.026279911 0.009613462 0.050198272 0.029701454
 [7] 0.086540502 0.018433968 0.007921595 0.021976439 0.009261722 0.125325581
[13] 0.035441269 0.005144094
```

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

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

[[2]]
[1] 12  1  3

[[3]]
[1] 7 1 2

[[4]]
[1] 9 6 5

[[5]]
[1] 13  7  4

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

[[7]]
[1] 3 6 5

[[8]]
[1] -5  9 11

[[9]]
[1]  4  8 14

[[10]]
[1] -5 12 11

[[11]]
[1]  8 10 14

[[12]]
[1]  2 10 13

[[13]]
[1]  5 12 14

[[14]]
[1]  9 13 11
```