Qhull examples

David C. Sterratt

31st August 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,]    1   14
[2,]    3   14
[3,]   11   13
[4,]   11    6
[5,]    7   13
[6,]    7    1
```

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] 10.13076
```
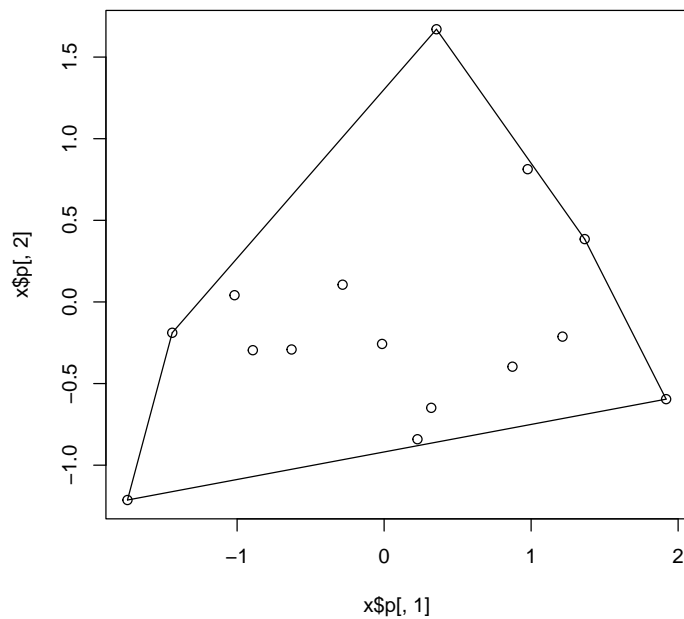
```
> print(ch$vol)
```

```
[1] 5.409989
```

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,]   0.1662698 -0.9860803 -0.9061268
[2,]  -0.9585738  0.2848443 -1.3279163
[3,]  -0.7189404  0.6950718 -0.9050389
[4,]   0.8706029  0.4919864 -1.3772447
[5,]   0.7868573  0.6171350 -1.3111625
```

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)
```
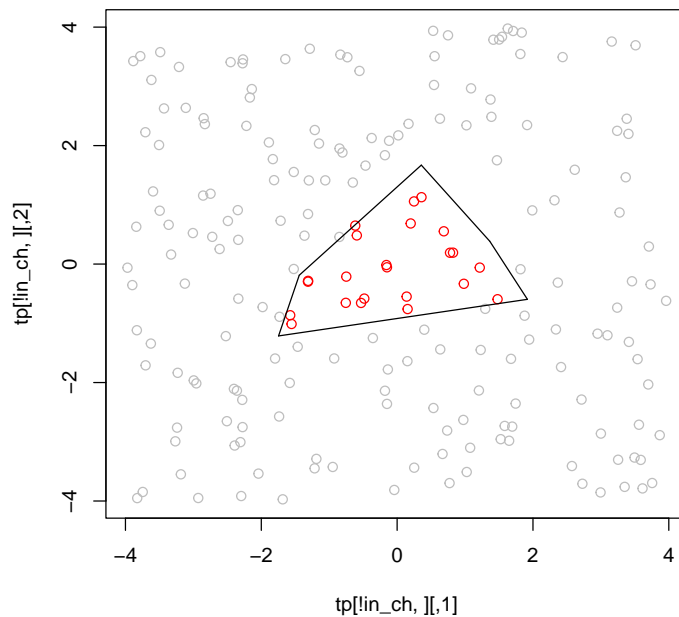
2 Delaunay triangulation in 2D

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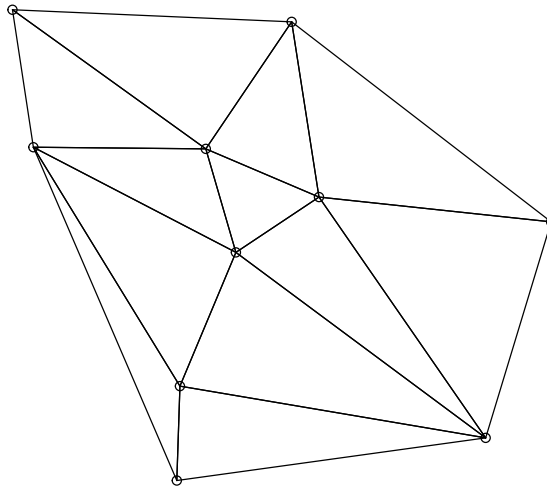
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

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

```
[2,]    10    1    4
[3,]    10    1    9
[4,]     7    3    6
[5,]     7    3    9
[6,]     7   10    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] 0.06130087 0.07933516 0.04479724 0.02234617 0.05052413 0.04447604
 [7] 0.06709637 0.02728631 0.01573636 0.02837476 0.05593327 0.03627724

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

```
[[1]]
[1] -5  2 10

[[2]]
[1] -8  1  3

[[3]]
[1] 7 9 2

[[4]]
[1] -12   6   5

[[5]]
[1] 8 7 4

[[6]]
[1] -9  4  7

[[7]]
[1] 3 5 6

[[8]]
[1]  5  9 12

[[9]]
[1]  3  8 10

[[10]]
[1]  1 11  9

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

[[12]]
[1] -12  11   8
```