Qhull examples

David C. Sterratt

3rd September 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,]   14   12
[2,]   14    6
[3,]   15    6
[4,]   11   15
[5,]   10   12
[6,]   10   11
```

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)
```

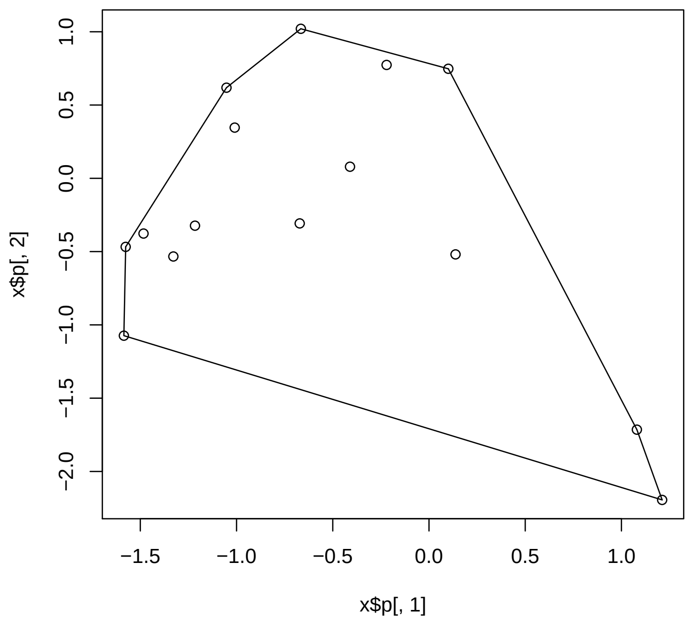
```
[1] 9.342614
```

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

```
[1] 4.865268
```

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)
```

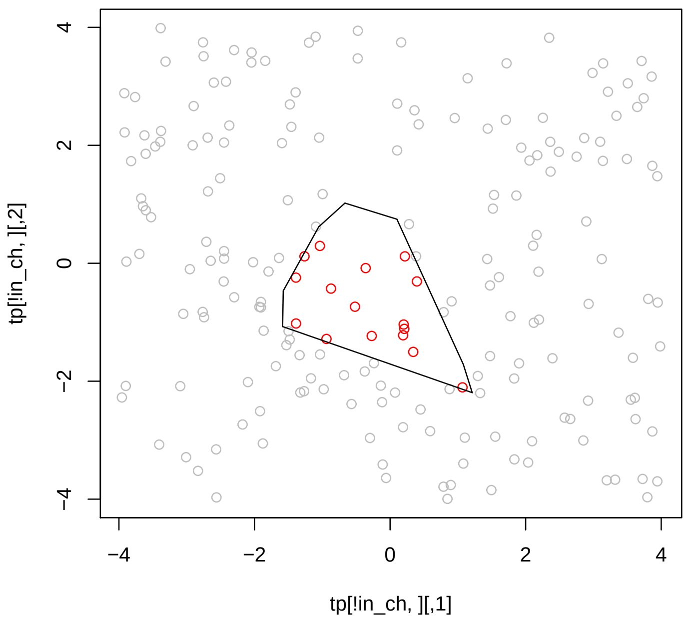
```
           [,1]        [,2]       [,3]
[1,] -0.3719326 -0.92825975 -1.5860981
[2,]  0.3355017  0.94203961 -0.7377917
[3,] -0.9998825  0.01533019 -1.5687061
[4,]  0.9644909  0.26411596 -0.5890068
[5,]  0.9291260  0.36976339 -0.3696523
[6,] -0.7207810  0.69316286 -1.1871492
```

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

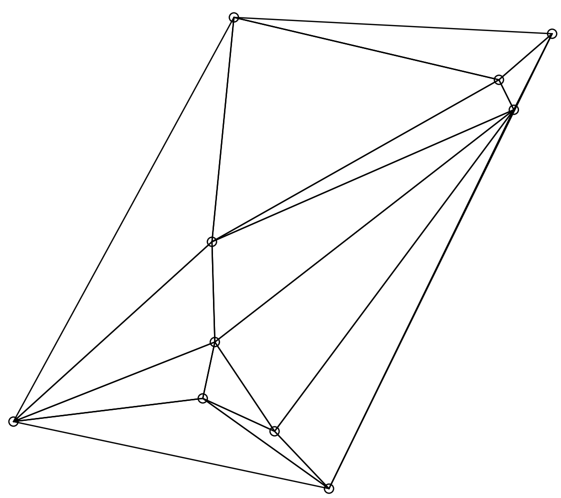
With one argument, a set of points, `delaunayn` returns the indices of the points at each vertex of each triangle in the triangulation.

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

```
     [,1] [,2] [,3]
[1,]    7    2    8
```

```
[2,]    1    5    8
[3,]    4    5   10
[4,]    9    1    5
[5,]    9    4    5
[6,]    6    7   10
```

```
> 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] 0.0477354797 0.0235319781 0.0005427886 0.0027393269 0.0367158707
 [6] 0.0182648794 0.0026812504 0.0715507219 0.0129480324 0.0052285253
[11] 0.0240145535 0.0122203334 0.0360452460 0.0477133185
```

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

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

[[2]]
[1] -1 12  4

[[3]]
[1] -1  7  5

[[4]]
[1]  2  5 10

[[5]]
[1]  3  4 14

[[6]]
[1] -5  7  8

[[7]]
[1] 3 6 9

[[8]]
[1] 1 9 6

[[9]]
[1] 13  8  7

[[10]]
[1]  4 12 14

[[11]]
[1]  1 12 13

[[12]]
[1]  2 11 10

[[13]]
[1]  9 11 14

[[14]]
[1]  5 13 10
```