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.

```r
> library(geometry)
> ps <- matrix(rnorm(30), , 2)
> ch <- convhulln(ps)
> head(ch)

[,1] [,2]
[1,] 7  15
[2,] 14 15
[3,] 11  6
[4,] 11  7
[5,] 13  6
[6,] 13 14
```

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.

```r
> ps <- matrix(rnorm(30), , 2)
> ch <- convhulln(ps, options="FA")
> print(ch$area)

[1] 13.49196
```
A `convhulln` object can also be plotted.

```r
> plot(ch)
```

We can also find the normals to the "facets" of the convex hull:

```r
> ch <- convhulln(ps, options="n")
> head(ch$normals)

```

<table>
<thead>
<tr>
<th>[,1]</th>
<th>[,2]</th>
<th>[,3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.7841531</td>
<td>-0.6205675</td>
<td>-1.407990</td>
</tr>
<tr>
<td>0.9841861</td>
<td>-0.1771377</td>
<td>-1.068412</td>
</tr>
<tr>
<td>0.2902411</td>
<td>0.9569535</td>
<td>-2.341298</td>
</tr>
<tr>
<td>-0.6348369</td>
<td>0.7726462</td>
<td>-1.569560</td>
</tr>
<tr>
<td>-0.5821727</td>
<td>0.8130652</td>
<td>-1.593810</td>
</tr>
</tbody>
</table>
```

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.

```r
> 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.

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

[,1] [,2] [,3]
[1,] 4 7 10
```
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.

```r
> dt2 <- delaunayn(ps, options="Fa")
> print(dt2$areas)
[1] 0.094983849 0.084280224 0.058328041 0.007140705 0.057067978 0.067100078
[7] 0.033513186 0.097113988 0.001922239 0.022078453 0.008318539 0.105728194
> dt2 <- delaunayn(ps, options="Fn")
> print(dt2$neighbours)
```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-13</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>-10</td>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>-10</td>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>-9</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>-9</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>