# Package ‘alphashape’

April 24, 2020

**Type** Package

**Title** Create Delaunay triangulations, Voronoi vertices and alpha shape for n number of dimension using the QHULL library

**Version** 1.2

**Date** 2019-03-15

**Maintainer** Pascal Omondiagbe <omondiagbep@landcareresearch.co.nz>

**Description** Makes an Alpha shape for any number of N dimension using the Delaunay triangulations generated using via the Qhull library (www.qhull.org)
This package has been tested to work up to 5 number of dimension.

**License** MIT + file LICENSE

**Depends** R (>= 3.5.1)

**Imports** devtools

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.0.2

**Suggests** testthat

**NeedsCompilation** no

## R topics documented:

`alphashape` *Computation of n dimension $\alpha$-shape*

## Description

Implementation in n dimension of the alpha shape using the Q-hull library

|  |  |
|---|---|
| Package: | alphashape |
| Date: | 2019-03-14 |
| License: | GPL-2 |
| LazyLoad: | yes |

### Author(s)

Pascal Omondiagbe
Tom Etherington
Maintainers: pascal Omondiagbe <omondiagbep@landcareresearch.co.nz>

### References

http://www.qhull.org/html/qh-code.htm

---

## `alpha_shape` *alpha_shape*

---

### Description

Compute an alpha Shape Grid using the Q-hull library.

### Usage

```
alpha_shape(point, alphaRange, maxs, mins, n)
```

### Arguments

|  |  |
|---|---|
| `point` | observation as dataframe or matrix |
| `alphaRange,` | range of alpha value |
| `mins` | Vector of length `n` listing the point space minimum for each dimension. @param maxs Vector of length `n` listing the point space maximum for each dimension. |
| `n` | `n` dimension point co-ordinate |

### Details

The calculation is done by assigning the trigulation index when the grid cell center lies within the trigulation or -1 if it lies outside

### Value

grid stack as vector, gridSimplex, and the inputted grid point.

### Examples

```
x = c(30,70,20,50,40,70,20)
y = c(35,80,70,50,60,20,30)
p = data.frame(x,y)
alpha_shape(point = p,maxs = c(70,80),mins = c(20,20),n = 5,alphaRange = c(1:20))
```

convex_hull *Convex hull in $d$-dimensions.*

## Description

This function calculates the convex hull around a set of $n$ points in $d$-dimensional space using the Qhull library.

## Usage

```
convex_hull(points = NULL)
```

## Arguments

| | |
|---|---|
| points | points is an $n$-by-$d$ of dataframe or matrix. The rows of points represent $n$ points in $d$-dimensional space. |
| options | String containing extra options for the underlying Qhull command.(See the Qhull documentation (../doc/html/qdelaun.html) for the available options.) The Qbb option is always passed to Qhull. The default options are Qt. The degenerate (zero area) regions are returned. For silent operation, specify the option Pp. |

## Value

Returns a list consisting of...

## References

Barber CB, Dobkin DP, Huhdanpaa H (1996) The Quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software, 22(4):469-83 https://doi.org/10.1145/235815.235821.

## See Also

Used internally by convex_layer

## Examples

```
# Define points
x <- c(30, 70, 20, 50, 40, 70)
y <- c(35, 80, 70, 50, 60, 20)
p <- data.frame(x, y)
# Create convex hull and plot
ch <- convex_hull(points = p)
plot(p, pch = as.character(seq(nrow(p))))
polygon(ch$hull_points, border = "red")
```

## delaunay *Delaunay triangulation in N-dimensions.*

### Description

The Delaunay triangulation is a tessellation of the convex hull of the points such that no N-sphere defined by the N-triangles contains any other points from the set. This function calculates the Delaunay triangulation in N-dimensions using the qhull library

### Usage

```
delaunay(points = NULL)
```

### Arguments

`point` `point` is an `n`-by-`dim` dataframevor matrix. The rows of `point` represent `n` points in `dim`-dimensional space.

`options` String containing extra options for the underlying Qhull command.(See the Qhull documentation (`../doc/html/qdelaun.html`) for the available options.) The `Qbb` option is always passed to Qhull. The default options are `Qcc Qc Qt Qz` for `dim` <4 and `Qcc Qc Qt Qx` for `dim`>=4. If neither of the `QJ` or `Qt` options are supplied, the `Qt` option is passed to Qhull. The `Qt` option ensures all Delaunay regions are simplical (e.g., triangles in 2-d). See `../doc/html/qdelaun.html` for more details. The degenerate (zero area) regions are returned For silent operation, specify the option `Pp`.

`full` Return all information associated with the triangulation as a list and these are triangulation (`tri`), a vector of facet areas (`areas`) and a list of neighbours of each facet (`neighbours`) OR return the convexhull and the input point

### Examples

```
# Define points
x <- c(30, 70, 20, 50, 40, 70)
y <- c(35, 80, 70, 50, 60, 20)
p <- data.frame(x, y)
# Create Delaunay triangulation and plot
dt <- delaunay(points = p)
plot(p, pch = as.character(seq(nrow(p))))
```

## find_simplex *Find Simplex*

### Description

Returns the simplicies of the delaunay trigulation which contains a given point.

### Usage

```
find_simplex(tri, inputPoint, testPoint)
```

### Arguments

`tri` delaunay trigulation simplex using `delaunay`

`inputPoint` `n-by-dim` dataframe or matrix of original inputPoint.

`testPoint` `n-by-dim` dataframe of points or matrix to check.

### Details

Given a grid point and a test point point, the find Simplex will identify the simplicies contianing the test point. It works by first checking for all point inside a convex hull, and then check if the center of the grid cell is inside the trigulation

### Value

A `n*m` vector containing the result. -1 if a given point is outside the trigulation, or the trigulation index if the center of the cell is inside the alpha shape

### Examples

```
x = c(30,70,20,50,40,70)
y = c(35,80,70,50,60,20)
p = data.frame(x,y)
v=voronoi(point =p)
meshGrdiSpace = grid_coordinates(mins=c(15,0), maxs=c(35,200), nCoords=5)
simplex <- find_simplex(v$tri, v$inputPoints,meshGrdiSpace)
```

---

## `grid_coordinates` *Grid Coordinates*

---

### Description

Create an `n`-dimensional grid of coordinates across space.

### Usage

```
grid_coordinates(mins, maxs, nCoords)
```

### Arguments

`mins` Vector of length `n` listing the point space minimum for each dimension.

`maxs` Vector of length `n` listing the pointspace maximum for each dimension.

`nCoords` Number of coordinates across the point space in all dimensions.

### Details

This function creates a grid of coordinates systematically located throughout the specified point space to enable visualisation of alpha shape . The extent of the grid is given by the `mins` and `maxs`, and the number of coordinates for each dimension is given by `nCoords`.

### Value

A matrix with `n` columns.

### Examples

```
# Point space grid coordinates usage
xy = grid_coordinates(mins=c(15,0), maxs=c(35,200), nCoords=5)
```

## `in_convex_hull` *in convex hull*

### Description

To test if a given point is inside the convex hull. `TRUE` if the point lies within the hull and `FALSE` if it lies outwith the hull

### Usage

```
in_convex_hull(hull = NULL, test_points = NULL)
```

### Arguments

| | |
|---|---|
| `points` | points to make convex hull |
| `test_points:` | dataframe or matrix `n-by-dim` of points to check. |

### Value

A `n*m` vector containing the result. True if a given point was inside the convexhull , otherwise false

### Examples

```
# Define points to create the convex hull
x <- c(30, 70, 20, 50, 40, 70)
y <- c(35, 80, 70, 50, 60, 20)
p <- data.frame(x, y)
ch <- convex_hull(points = p)
# Check if some new points are in the convex hull
new = data.frame(c(20, 50), c(20, 50))
checks <- in_convex_hull(hull=ch, test_points = new)
```

## `voronoi` *Get Voronoi triangle and Delaunay triangulation*

### Description

Get Voronoi vertices(circumcenters of Delaunay triangle) and Delaunay triangulation in N-dimensions using the qhull library. The Voronoi diagram is the nearest-neighbor map for a set of points. Each region contains those points that are nearer one input site than any other input site. They can also be describe as the circumcenters of Delaunay triangle. The Delaunay triangulation is a tessellation of the convex hull ofthe points such that no N-sphere defined by the N-triangles contains any other points from the set.

### Usage

```
voronoi(points = NULL)
```

### Arguments

`point` : `point` is an `n`-by-`dim` dataframe or matrix. The rows of `point` represent `n` points in `dim`-dimensional space.

`options` : String containing extra options for the underlying Qhull command.(See the Qhull documentation (`../doc/html/qvoronoi.html`) for the available options.) The `Qbb` option is always passed to Qhull. The default options are `Qcc Qc Qt Qz` for `dim` <4 and `Qcc Qc Qt Qx` for `dim`>=4. If neither of the `QJ` or `Qt` options are supplied, the `Qt` option is passed to Qhull. The `Qt` option ensures all Delaunay regions are simplical (e.g., triangles in 2-d). See `../doc/html/qdelaun.html` for more details.

`full` : Return all information associated with triangulation as a list. This will return the triangulation (`tri`), list ofvoronoi Vertices (`voronoiVertices`) , vpoints (`points`) ,matrix list of the circum radii (`circumRadii`), matrix list of (`SimplicesPoints`) and a list of neighbours of each facet (`neighbours`).

### Value

The voronoi vertics, circumRadii,trigulation Points and Delaunay triangulation

### Examples

```
# Define points
x <- c(30, 70, 20, 50, 40, 70)
y <- c(35, 80, 70, 50, 60, 20)
p <- data.frame(x, y)
# Create Voronoi diagram and plot
vd <- voronoi(points = p)
plot(p, pch = as.character(seq(nrow(p))))
```

# Index