Package: cxhull (via r-universe)

March 26, 2025

Type Package

Title Convex Hull

Version 0.7.4

Date 2023-10-23

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Computes the convex hull in arbitrary dimension, based on the Qhull library (<http://www.qhull.org>). The package provides a complete description of the convex hull: edges, ridges, facets, adjacencies. Triangulation is optional.

License GPL-3

URL https://github.com/stla/cxhull

BugReports https://github.com/stla/cxhull/issues

Depends R (>= 2.10)

Imports data.table, grDevices, rgl, Rvcg

Suggests colorspace

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation yes

Author C. B. Barber [cph] (author of the Qhull library), The Geometry Center [cph], Stéphane Laurent [cph, aut, cre]

Date/Publication 2023-10-24 04:40:09 UTC

Additional_repositories https://cranhaven.r-universe.dev

Config/pak/sysreqs libfreetype6-dev libglu1-mesa-dev make libpng-dev libgl1-mesa-dev zlib1g-dev

Repository https://cranhaven.r-universe.dev

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/cxhull

RemoteSha bf0f4c5f16ad02ad862dd44235d13da9cba119a3

RemoteSubdir cxhull

Contents

`cxhull` *Convex hull*

Description

Computes the convex hull of a set of points.

Usage

```
cxhull(points, triangulate = FALSE)
```

Arguments

`points`	numeric matrix, one point per row
`triangulate`	logical, whether to triangulate the convex hull

Value

A list providing a lot of information about the convex hull. See the **README** file for details.

Examples

```
library(cxhull)
points <- rbind(
 c(0.5,0.5,0.5),
 c(0,0,0),
 c(0,0,1),
 c(0,1,0),
 c(0,1,1),
 c(1,0,0),
 c(1,0,1),
 c(1,1,0),
```

```
 c(1,1,1)
)
cxhull(points)
```

---

cxhullEdges *Vertices and edges of convex hull*

Description

Computes the vertices and the edges of the convex hull of a set of points.

Usage

```
cxhullEdges(points, adjacencies = FALSE, orderEdges = FALSE)
```

Arguments

points	numeric matrix, one point per row; it must contain at least three columns (the two-dimensional case is not implemented yet)
adjacencies	Boolean, whether to return the vertex adjacencies
orderEdges	Boolean, whether to order the edges in the output

Value

A list with two fields: `vertices` and `edges`. The `vertices` field is a list which provides an id for each vertex and its coordinates. If `adjacencies=TRUE`, it provides in addition the ids of the adjacent vertices for each vertex. The `edges` fields is an integer matrix with two columns. Each row provides the two ids of the vertices of the corresponding edge.

Examples

```
library(cxhull)
# let's try with the hexacosichoron (see `?hexacosichoron`)
#   it is convex so its convex hull is itself
VE <- cxhullEdges(hexacosichoron)
edges <- VE[["edges"]]
random_edge <- edges[sample.int(720L, 1L), ]
A <- hexacosichoron[random_edge[1L], ]
B <- hexacosichoron[random_edge[2L], ]
sqrt(c(crossprod(A - B))) # this is 2/phi
# Now let's project the polytope to the H4 Coxeter plane
phi <- (1 + sqrt(5)) / 2
u1 <- c(
  0,
  2*phi*sin(pi/30),
  0,
  1
)
u2 <- c(
```

```
  2*phi*sin(pi/15),
  0,
  2*sin(2*pi/15),
  0
)
u1 <- u1 / sqrt(c(crossprod(u1)))
u2 <- u2 / sqrt(c(crossprod(u2)))
# projections to the Coxeter plane
proj <- function(v){
  c(c(crossprod(v, u1)), c(crossprod(v, u2)))
}
points <- t(apply(hexacosichoron, 1L, proj))
# we will assign a color to each edge
#   according to the norms of its two vertices
norms2 <- round(apply(points, 1L, crossprod), 1L)
( tbl <- table(norms2) )
#> 0.4 1.6 2.4 3.6
#>  30  30  30  30
values <- as.numeric(names(tbl))
grd <- as.matrix(expand.grid(values, values))
grd <- grd[grd[, 1L] <= grd[, 2L], ]
pairs <- apply(grd, 1L, paste0, collapse = "-")
colors <- hcl.colors(nrow(grd), palette = "Hawaii", rev = TRUE)
if(require("colorspace")) {
  colors <- colorspace::darken(colors, amount = 0.3)
}
names(colors) <- pairs
# plot ####
opar <- par(mar = c(0, 0, 0, 0))
plot(
  points[!duplicated(points), ], pch = 19, cex = 0.3, asp = 1,
  axes = FALSE, xlab = NA, ylab = NA
)
for(i in 1L:nrow(edges)){
  twopoints <- points[edges[i, ], ]
  nrms2 <- round(sort(apply(twopoints, 1L, crossprod)), 1L)
  pair <- paste0(nrms2, collapse = "-")
  lines(twopoints, lwd = 0.5, col = colors[pair])
}
par(opar)
```

`daVinciSphere` *Leonardo da Vinci's 72-sided sphere*

Description

A matrix giving the 62 vertices of da Vinci's 72-sided sphere, a convex polyhedra with 72 faces.

Usage

```
daVinciSphere
```

Format

A matrix with 62 rows and 3 columns.

Source

http://www.matematicasvisuales.com/english/html/geometry/space/sphereCampanus.html

---

dihedralAngles *Dihedral angles*

Description

Dihedral angles of a convex hull.

Usage

```
dihedralAngles(hull)
```

Arguments

hull	an output of cxhull applied to 3D points

Value

A dataframe with three columns. The two first columns represent the edges, given as a pair of vertex indices. The third column provides the dihedral angle in degrees corresponding to the edge, that is the angle between the two faces incident to this edge. This is useful to find edges between two coplanar faces: if the faces are exactly coplanar then the dihedral angle is 180, but because of numerical approximation one can consider that there is coplanarity when the dihedral angle is greater than 179, for example. This function is used in plotConvexHull3d to get rid of such edges (if the user sets a value to the argument angleThreshold).

Examples

```
# a cube ####
library(cxhull)
points <- rbind(
 c(0.5,0.5,0.5),
 c(0,0,0),
 c(0,0,1),
 c(0,1,0),
 c(0,1,1),
 c(1,0,0),
 c(1,0,1),
 c(1,1,0),
 c(1,1,1)
)
hull <- cxhull(points)
dihedralAngles(hull)
```

EdgesAB *Edges of a triangulated 3D convex hull*

Description

Edges of a triangulated 3D convex hull given by the ids of the vertices in a matrix, plus a column indicating the border edges.

Usage

```
EdgesAB(hull)
```

Arguments

hull	an output of cxhull applied to 3D points and with the option triangulate=TRUE

Value

A character matrix with three columns. Each row provides the ids of the two vertices of an edge, and a yes/no indicator of whether the edge is a border edge.

Examples

```
library(cxhull)
library(rgl)
dodecahedron <- t(dodecahedron3d()$vb[-4L, ])
hull <- cxhull(dodecahedron, triangulate = TRUE)
triangles <- TrianglesXYZ(hull)
triangles3d(triangles, color = "yellow")
edges <- EdgesAB(hull)
trueEdges <- edges[edges[, 3L] == "yes", c(1L, 2L)]
otherEdges <- edges[edges[, 3L] == "no", c(1L, 2L)]
vertices <- VerticesXYZ(hull)
for(i in 1:nrow(trueEdges)){
  lines3d(vertices[trueEdges[i, ], ], color = "blue", lwd = 3)
}
for(i in 1:nrow(otherEdges)){
  lines3d(vertices[otherEdges[i, ], ], color = "red", lwd = 3)
}
```

EdgesXYZ *Edges coordinates*

Description

The coordinates of the extremities of the edges in a matrix, plus a column indicating which edges are border edges.

Usage

```
EdgesXYZ(hull)
```

Arguments

hull	an output of cxhull applied to 3D points and with the option triangulate=TRUE

Value

A numeric matrix with four columns. The first three values of a row are the coordinates of a vertex at the extremity of an edge, and the fourth column indicates whether the edge is a border edge.

hexacosichoron *Vertices of the 600-cell*

Description

A matrix giving the 120 vertices of the hexacosichoron, a regular convex 4D polytope also known as the "600-cell", with edge length 2/phi, where phi is the golden number. It has 720 edges.

Usage

```
hexacosichoron
```

Format

A matrix with 120 rows and 4 columns.

Source

https://www.qfbox.info/4d/600-cell

`hullMesh` *Mesh of a 3d convex hull*

Description

Extract the vertices and the faces from a 3d convex hull.

Usage

```
hullMesh(hull, simplify = TRUE, rgl = FALSE)
```

Arguments

`hull`	a 3d convex hull, output of `cxhull`
`simplify`	Boolean, whether to return the faces as a matrix instead of a list if possible, i.e. if all faces have the same number of edges; this argument is possibly ignored if `rgl=TRUE`, see below
`rgl`	Boolean, whether to return a **rgl** mesh (class `mesh3d`) if possible, i.e. if each face has three or four edges; if `TRUE` and the **rgl** mesh is possible, then the `simplify` argument has no effect

Value

A list giving the vertices and the faces, or a **rgl** mesh.

Note

Unless all faces are triangular, the output does not define a mesh with coherently oriented faces.

Examples

```
library(cxhull)
hull <- cxhull(daVinciSphere)
septuaginta <- hullMesh(hull, rgl = TRUE)
library(rgl)
open3d(windowRect = c(50, 50, 562, 562))
shade3d(septuaginta, color = "darkred")
# some quad faces are misoriented:
open3d(windowRect = c(50, 50, 562, 562))
shade3d(septuaginta, color = "tomato", back = "culled")
```

hullSummary *Summary of 3D convex hull*

Description

Summary of a triangulated 3D convex hull

Usage

```
hullSummary(hull)
```

Arguments

hull	an output of cxhull applied to 3D points and with the option triangulate=TRUE

Value

A list with the vertices and the facets.

Examples

```
library(cxhull)
# pyramid
pts <- rbind(
  c(0, 0, 0),
  c(1, 0, 0),
  c(1, 1, 0),
  c(0.5, 0.5, 1),
  c(0.5, 0.5, 0.9),
  c(0, 1, 0)
)
hull <- cxhull(pts, triangulate = TRUE)
hullSummary(hull)
```

plotConvexHull3d *Plot triangulated 3d convex hull*

Description

Plot a triangulated 3d convex hull with **rgl**.

Usage

```
plotConvexHull3d(
  hull,
  angleThreshold = NULL,
  edgesAsTubes = TRUE,
  verticesAsSpheres = TRUE,
  palette = NULL,
  bias = 1,
  interpolate = "linear",
  g = identity,
  facesColor = "navy",
  edgesColor = "gold",
  tubesRadius = 0.03,
  spheresRadius = 0.05,
  spheresColor = edgesColor,
  alpha = 1
)
```

Arguments

`hull`	an output of `cxhull` applied to 3d points and with the option `triangulate=TRUE`
`angleThreshold`	a threshold angle in degrees, typically `179`, to get rid of edges between coplanar faces: edges whose corresponding dihedral angle is greater than this threshold are removed; `NULL` to use another method (see the Leonardo example)
`edgesAsTubes`	Boolean, whether to draw the edges as tubes
`verticesAsSpheres`	Boolean, whether to draw the vertices as spheres
`palette`	a vector of colors to make a color gradient for the faces; if `NULL`, the colors of the faces are controlled by the `facesColor` argument
`bias`, `interpolate`	if `palette` is not `NULL`, these arguments are passed to `colorRamp`
`g`	a function defined on [0, 1] and taking its values in [0, 1]; it is composed with the function created by `colorRamp`, based on `palette`
`facesColor`	the color(s) for the faces; this argument is ignored if the argument `palette` is not `NULL`; otherwise there are three possibilities for `facesColor`: a single color, a vector of colors with length the number of triangles, in which case one color is assigned per triangle, or a vector of colors with length the number of faces, after merging the triangles, in which case one color is assigned per face; use `hullSummary` to know the number of faces
`edgesColor`	the color for the edges
`tubesRadius`	the radius of the tubes when `edgesAsTubes=TRUE`
`spheresRadius`	the radius of the spheres when `verticesAsSpheres=TRUE`
`spheresColor`	the color of the spheres when `verticesAsSpheres=TRUE`
`alpha`	number between 0 and 1 controlling the opacity of the faces

Value

No value.

Examples

```
library(cxhull)
library(rgl)
cuboctahedron <- t(cuboctahedron3d()$vb[-4L, ])
hull <- cxhull(cuboctahedron, triangulate = TRUE)
# single color ####
open3d(windowRect = c(50, 50, 562, 562))
plotConvexHull3d(hull)
# gradient ####
open3d(windowRect = c(50, 50, 562, 562))
if(getRversion() < "4.1.0"){
  palette <- "Viridis"
}else{
  palette <- "Rocket"
}
plotConvexHull3d(hull, palette = hcl.colors(256, palette), bias = 0.5)


library(cxhull)
library(rgl)
# Leonardo da Vinci's 72-sided sphere ####
hull <- cxhull(daVinciSphere, triangulate = TRUE)
# there are some undesirable edges:
plotConvexHull3d(
  hull, tubesRadius = 0.07, spheresRadius = 0.1
)
# => use `angleThreshold` to get rid of these edges:
plotConvexHull3d(
  hull, angleThreshold = 179,
  tubesRadius = 0.07, spheresRadius = 0.1
)
```

TrianglesXYZ *Triangles of a triangulated 3D convex hull*

Description

Coordinates of the vertices of the triangles of a triangulated 3D convex hull.

Usage

```
TrianglesXYZ(hull)
```

Arguments

hull	an output of cxhull applied to 3D points and with the option triangulate=TRUE

Value

A matrix with three columns. Each row represents the coordinates of a vertex of a triangle.

Examples

```
library(cxhull)
library(rgl)
dodecahedron <- t(dodecahedron3d()$vb[-4L, ])
hull <- cxhull(dodecahedron, triangulate = TRUE)
triangles <- TrianglesXYZ(hull)
triangles3d(triangles, color = "firebrick")
```

VerticesXYZ *Convex hull vertices*

Description

The coordinates of the vertices of a 3D convex hull.

Usage

```
VerticesXYZ(hull)
```

Arguments

hull	an output of cxhull applied to 3D points

Value

A matrix with three columns. Each row represents the coordinates of a vertex and the row names are the ids of the vertices.

Index