Package 'geometry'

July 2, 2014

Maintainer David C. Sterratt <david.c.sterratt@ed.ac.uk>

License GPL (>= 3) + file LICENSE

Title Mesh generation and surface tesselation

Author C. B. Barber, Kai Habel, Raoul Grasman, Robert B. Gramacy, Andreas Stahel and David C. Sterratt

Description This package makes the qhull library (www.qhull.org) available in R, in a similar manner as in Octave and MATLAB. Qhull computes convex hulls, Delaunay triangulations, halfspace intersections about a point, Voronoi diagrams, furthest-site Delaunay triangulations, and furthest-site Voronoi diagrams. It runs in 2-d, 3-d, 4-d, and higher dimensions. It implements the Quickhull algorithm for computing the convex hull. Qhull does not support constrained Delaunay triangulations, or mesh generation of non-convex objects, but the package does include some R functions that allow for this. Currently the package only gives access to Delaunay triangulation and convex hull computation.

Version 0.3-4

URL http://geometry.r-forge.r-project.org/

Date 2014-03-04

BugReports https://r-forge.r-project.org/tracker/?group_id=1149

Depends R (>= 2.5.0), magic

Suggests rgl, R.matlab, tripack

Collate 'convhulln.R' 'delaunayn.R' 'distmesh2d.R' 'distmeshnd.R' 'entry.value.R' 'extprod3d.R' 'matmax.R' 'matmin.R' 'matorder.R' 'matsort.R' 'mesh.dcircle.R' 'mesh.diff.R' 'mesh.drectangle.R' 'mesh.dsphere.R' 'mesh.hunif.R' 'mesh.intersect.R' 'mesh.union.R' 'surf.tri.R' 'tetramesh.R' 'trimesh.R' 'trisplinter.R' 'tsearch.R' 'Unique.R' 'dotprod.R' 'polyarea.R'

Repository CRAN

Repository/R-Forge/Project geometry

Repository/R-Forge/Revision 128

Repository/R-Forge/DateTimeStamp 2014-03-05 17:21:51

Date/Publication 2014-03-06 13:44:47

NeedsCompilation yes

R topics documented:

`bary2cart` *Conversion of Barycentric to Cartesian coordinates*

Description

Given the baryocentric coordinates of one or more points with respect to a simplex, compute the Cartesian coordinates of these points.

Usage

```
bary2cart(X, Beta)
```

Arguments

X	Reference simplex in N dimensions represented by a $N+1$-by-N matrix
Beta	M points in baryocentric coordinates with respect to the simplex X represented by a M-by-$N+1$ matrix

Value

M-by-N matrix in which each row is the Cartesian coordinates of corresponding row of Beta

Author(s)

David Sterratt

---

cart2bary *Conversion of Cartesian to Barycentric coordinates.*

Description

Given the Cartesian coordinates of one or more points, compute the barycentric coordinates of these points with respect to a simplex.

Usage

```
cart2bary(X, P)
```

Arguments

X	Reference simplex in N dimensions represented by a $N+1$-by-N matrix
P	M-by-N matrix in which each row is the Cartesian coordinates of a point.

Details

Given a reference simplex in N dimensions represented by a $N+1$-by-N matrix an arbitrary point $\mathbf{P}$ in Cartesian coordinates, represented by a 1-by-N row vector, can be written as

$$\mathbf{P} = \beta \mathbf{X}$$

where β is a $N+1$ vector of the barycentric coordinates. A criterion on β is that

$$\sum_i \beta_i = 1$$

Now partition the simplex into its first N rows $\mathbf{X}_N$ and its $N+1$th row $\mathbf{X}_{N+1}$. Partition the barycentric coordinates into the first N columns β_N and the $N+1$th column β_{N+1}. This allows us to write

$$\mathbf{P} - \mathbf{X}_{N+1} = \beta_N \mathbf{X}_N + \beta_{N+1}\mathbf{X}_{N+1} - \mathbf{X}_{N+1}$$

which can be written

$$\mathbf{P} - \mathbf{X}_{N+1} = \beta_N(\mathbf{X}_N - \mathbf{1}\mathbf{X}_{N+1})$$

where $\mathbf{1}$ is a N-by-1 matrix of ones. We can then solve for β_N:

$$\beta_N = (\mathbf{P} - \mathbf{X}_{N+1})(\mathbf{X}_N - \mathbf{1}\mathbf{X}_{N+1})^{-1}$$

and compute

$$\beta_{N+1} = 1 - \sum_{i=1}^{N} \beta_i$$

This can be generalised for multiple values of $\mathbf{P}$, one per row.

Value

M-by-N matrix in which each row is the Cartesian coordinates of corresponding row of `P`. If the simplex is degenerate a warning is issued and the function returns `NULL`.

Note

Based on the Octave function by David Bateman.

Author(s)

David Sterratt

`convhulln` *Compute smallest convex hull that encloses a set of points*

Description

Returns an index matrix to the points of simplices ("triangles") that form the smallest convex simplicial complex of a set of input points in N-dimensional space. This function interfaces the Qhull library.

Usage

```
convhulln(p, options = "Tv")
```

Arguments

`p`	An `n`-by-`dim` matrix. The rows of `p` represent `n` points in `dim`-dimensional space.
`options`	String containing extra options for the underlying Qhull command; see details below and Qhull documentation at http://www.qhull.org/html/qconvex.htm#synopsis.

Details

For slient operation, specify the option Pp.

Value

An m-by-dim index matrix of which each row defines a dim-dimensional "triangle". The indices refer to the rows in p. If the option FA is provided, then the output is a list with entries $hull containing the matrix mentioned above, and $area and $vol with the area and volume of the hull described by the matrix.

Note

This is a port of the Octave's (http://www.octave.org) geometry library. The Octave source was written by Kai Habel.

See further notes in delaunayn.

Author(s)

Raoul Grasman, Robert B. Gramacy and David Sterratt <david.c.sterratt@ed.ac.uk>

References

Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull algorithm for convex hulls," ACM Trans. on Mathematical Software, *Dec 1996.*

http://www.qhull.org

See Also

convex.hull, delaunayn, surf.tri, distmesh2d

Examples

```
# example convhulln
# ==> see also surf.tri to avoid unwanted messages printed to the console by qhull
ps <- matrix(rnorm(3000), ncol=3)  # generate points on a sphere
ps <- sqrt(3)*ps/drop(sqrt((ps^2) %*% rep(1, 3)))
ts.surf <- t(convhulln(ps))  # see the qhull documentations for the options
## Not run:
rgl.triangles(ps[ts.surf,1],ps[ts.surf,2],ps[ts.surf,3],col="blue",alpha=.2)
for(i in 1:(8*360)) rgl.viewpoint(i/8)

## End(Not run)
```

`delaunayn` *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.

Usage

```
delaunayn(p, options = "", full = FALSE)
```

Arguments

`p`	`p` is an `n`-by-`dim` matrix. The rows of `p` 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.)
`full`	Return all information asscoiated with triangulation as a list. At present this is the triangulation (`tri`) and a list of neighbours of each facet (`neighbours`).

Details

If neither of the `QJ` or `Qt` options are supplied, the `QJ` is passed to Qhull. The `QJ` prevents the creation of degenerate simplicies (i.e. those that lie in a `N-1`-dimensional subspace). See `../doc/html/qdelaun.html` for more details.

For slient operation, specify the option `Pp`.

Value

The return matrix has `m` rows and `dim+1` columns. It contains for each row a set of indices to the points, which describes a simplex of dimension `dim`. The 3D simplex is a tetrahedron.

Note

This function interfaces the Qhull library and is a port from Octave (`http://www.octave.org`) to R. Qhull computes convex hulls, Delaunay triangulations, halfspace intersections about a point, Voronoi diagrams, furthest-site Delaunay triangulations, and furthest-site Voronoi diagrams. It runs in 2-d, 3-d, 4-d, and higher dimensions. It implements the Quickhull algorithm for computing the convex hull. Qhull handles roundoff errors from floating point arithmetic. It computes volumes, surface areas, and approximations to the convex hull. See the Qhull documentation included in this distribution (the doc directory `../doc/index.html`).

Qhull does not support constrained Delaunay triangulations, triangulation of non-convex surfaces, mesh generation of non-convex objects, or medium-sized inputs in 9-D and higher. A rudimentary algorithm for mesh generation in non-convex regions using Delaunay triangulation is implemented in distmesh2d (currently only 2D).

Author(s)

Raoul Grasman and Robert B. Gramacy; based on the corresponding Octave sources of Kai Habel.

References

Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull algorithm for convex hulls," ACM Trans. on Mathematical Software, *Dec 1996.*

http://www.qhull.org

See Also

tri.mesh, convhulln, surf.tri, distmesh2d

Examples

```
# example delaunayn
d <- c(-1,1)
pc <- as.matrix(rbind(expand.grid(d,d,d),0))
tc <- delaunayn(pc)

# example tetramesh
## Not run:
library(rgl)
rgl.viewpoint(60)
rgl.light(120,60)
tetramesh(tc,pc, alpha=0.9)

## End(Not run)
```

---

distmesh2d *A simple mesh generator for non-convex regions*

Description

An unstructured simplex requires a choice of meshpoints (vertex nodes) and a triangulation. This is a simple and short algorithm that improves the quality of a mesh by relocating the meshpoints according to a relaxation scheme of forces in a truss structure. The topology of the truss is reset using Delaunay triangulation. A (sufficiently smooth) user supplied signed distance function (fd) indicates if a given node is inside or outside the region. Points outside the region are projected back to the boundary.

Usage

```
distmesh2d(fd, fh, h0, bbox, p = NULL,
  pfix = array(0, dim = c(0, 2)), ..., dptol = 0.001,
  ttol = 0.1, Fscale = 1.2, deltat = 0.2,
  geps = 0.001 * h0,
  deps = sqrt(.Machine$double.eps) * h0, maxiter = 1000)
```

Arguments

fd	Vectorized signed distance function, for example mesh.dcircle or mesh.diff, accepting an n-by-2 matrix, where n is arbitrary, as the first argument.
fh	Vectorized function, for example mesh.hunif, that returns desired edge length as a function of position. Accepts an n-by-2 matrix, where n is arbitrary, as its first argument.
h0	Initial distance between mesh nodes. (Ignored of p is supplied)
bbox	Bounding box cbind(c(xmin,xmax), c(ymin,ymax))
p	An n-by-2 matrix. The rows of p represent locations of starting mesh nodes.
pfix	nfix-by-2 matrix with fixed node positions.
...	parameters to be passed to fd and/or fh
dptol	Algorithm stops when all node movements are smaller than dptol
ttol	Controls how far the points can move (relatively) before a retriangulation with delaunayn.
Fscale	"Internal pressure" in the edges.
deltat	Size of the time step in Eulers method.
geps	Tolerance in the geometry evaluations.
deps	Stepsize Δx in numerical derivative computation for distance function.
maxiter	Maximum iterations.

Details

This is an implementation of original Matlab software of Per-Olof Persson.

Excerpt (modified) from the reference below:

'The algorithm is based on a mechanical analogy between a triangular mesh and a 2D truss structure. In the physical model, the edges of the Delaunay triangles of a set of points correspond to bars of a truss. Each bar has a force-displacement relationship $f(\ell, \ell_0)$ depending on its current length ℓ and its unextended length ℓ_0.'

'External forces on the structure come at the boundaries, on which external forces have normal orientations. These external forces are just large enough to prevent nodes from moving outside the boundary. The position of the nodes are the unknowns, and are found by solving for a static force equilibrium. The hope is that (when `fh = function(p) return(rep(1,nrow(p)))`), the lengths of all the bars at equilibrium will be nearly equal, giving a well-shaped triangular mesh.'

See the references below for all details. Also, see the comments in the source file.

Value

n-by-2 matrix with node positions.

Wishlist

- *Implement in C/Fortran
- *Implement an nD version as provided in the matlab package
- *Translate other functions of the matlab package

Author(s)

Raoul Grasman

References

http://www-math.mit.edu/~persson/mesh/

P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB. SIAM Review, Volume 46 (2), pp. 329-345, June 2004

See Also

tri.mesh, delaunayn, mesh.dcircle, mesh.drectangle, mesh.diff, mesh.union, mesh.intersect

Examples

```
# examples distmesh2d
fd <- function(p, ...) sqrt((p^2)%*%c(1,1)) - 1
     # also predefined as `mesh.dcircle'
fh <- function(p,...)  rep(1,nrow(p))
bbox <- matrix(c(-1,1,-1,1),2,2)
p <- distmesh2d(fd,fh,0.2,bbox, maxiter=100)
    # this may take a while:
    # press Esc to get result of current iteration

# example with non-convex region
fd <- function(p, ...) mesh.diff(p , mesh.drectangle, mesh.dcircle, radius=.3)
     # fd defines difference of square and circle

p <- distmesh2d(fd,fh,0.05,bbox,radius=0.3,maxiter=4)
p <- distmesh2d(fd,fh,0.05,bbox,radius=0.3, maxiter=10)
     # continue on previous mesh
```

---

distmeshnd *A simple mesh generator for non-convex regions in n-D space*

Description

An unstructured simplex requires a choice of meshpoints (vertex nodes) and a triangulation. This is a simple and short algorithm that improves the quality of a mesh by relocating the meshpoints according to a relaxation scheme of forces in a truss structure. The topology of the truss is reset using Delaunay triangulation. A (sufficiently smooth) user supplied signed distance function (fd) indicates if a given node is inside or outside the region. Points outside the region are projected back to the boundary.

Usage

```
distmeshnd(fdist, fh, h, box,
  pfix = array(dim = c(0, ncol(box))), ..., ptol = 0.001,
  ttol = 0.1, deltat = 0.1, geps = 0.1 * h,
  deps = sqrt(.Machine$double.eps) * h)
```

Arguments

`fdist`	Vectorized signed distance function, for example `mesh.dsphere`, accepting an `m`-by-`n` matrix, where `m` is arbitrary, as the first argument.
`fh`	Vectorized function, for example `mesh.hunif`, that returns desired edge length as a function of position. Accepts an `m`-by-`n` matrix, where `n` is arbitrary, as its first argument.
`h`	Initial distance between mesh nodes.
`box`	2-by-`n` matrix that specifies the bounding box. (See distmesh2d for an example.)
`pfix`	`nfix`-by-2 matrix with fixed node positions.
`...`	parameters that are passed to `fdist` and `fh`
`ptol`	Algorithm stops when all node movements are smaller than `dptol`
`ttol`	Controls how far the points can move (relatively) before a retriangulation with `delaunayn`.
`deltat`	Size of the time step in Eulers method.
`geps`	Tolerance in the geometry evaluations.
`deps`	Stepsize Δx in numerical derivative computation for distance function.

Details

This is an implementation of original Matlab software of Per-Olof Persson.

Excerpt (modified) from the reference below:

'The algorithm is based on a mechanical analogy between a triangular mesh and a n-D truss structure. In the physical model, the edges of the Delaunay triangles of a set of points correspond to bars of a truss. Each bar has a force-displacement relationship $f(\ell, \ell_0)$ depending on its current length ℓ and its unextended length ℓ_0.'

'External forces on the structure come at the boundaries, on which external forces have normal orientations. These external forces are just large enough to prevent nodes from moving outside the boundary. The position of the nodes are the unknowns, and are found by solving for a static force equilibrium. The hope is that (when `fh = function(p) return(rep(1,nrow(p)))`), the lengths of all the bars at equilibrium will be nearly equal, giving a well-shaped triangular mesh.'

See the references below for all details. Also, see the comments in the source file of `distmesh2d`.

Value

`m`-by-`n` matrix with node positions.

Wishlist

- *Implement in C/Fortran
- *Translate other functions of the matlab package

Author(s)

Raoul Grasman; translated from original Matlab sources of Per-Olof Persson.

References

http://www-math.mit.edu/~persson/mesh/

P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB. SIAM Review, Volume 46 (2), pp. 329-345, June 2004

See Also

distmesh2d, tri.mesh, delaunayn, mesh.dsphere, mesh.hunif, mesh.diff, mesh.union, mesh.intersect

Examples

```
## Not run:
# examples distmeshnd
require(rgl)

fd = function(p, ...) sqrt((p^2)%*%c(1,1,1)) - 1
     # also predefined as `mesh.dsphere'
fh = function(p,...)  rep(1,nrow(p))
     # also predefined as `mesh.hunif'
bbox = matrix(c(-1,1),2,3)
p = distmeshnd(fd,fh,0.2,bbox, maxiter=100)
    # this may take a while:
    # press Esc to get result of current iteration

## End(Not run)
```

---

dot *Compute the dot product of two vectors*

Description

If x and y are matrices, calculate the dot-product along the first non-singleton dimension. If the optional argument d is given, calculate the dot-product along this dimension.

Usage

```
dot(x, y, d = NULL)
```

Arguments

x	Matrix of vectors
y	Matrix of vectors
d	Dimension along which to calculate the dot product

Value

Vector with length of dth dimension

Author(s)

David Sterratt

---

entry.value *Retrieve or set a list of array element values*

Description

entry.value retrieves or sets the values in an array a at the positions indicated by the rows of a matrix idx.

Usage

```
entry.value(a, idx)
```

Arguments

a	An array.
idx	Numerical matrix with the same number of columns as the number of dimensions of a. Each row indices a cell in a of which the value is to be retrieved or set.
value	An array of length nrow(idx).

Value

entry.value(a,idx) returns a vector of values at the indicated cells. entry.value(a,idx) <- val changes the indicated cells of a to val.

Author(s)

Raoul Grasman

Examples

```
a = array(1:(4^4),c(4,4,4,4))
entry.value(a,cbind(1:4,1:4,1:4,1:4))
entry.value(a,cbind(1:4,1:4,1:4,1:4)) <- 0

entry.value(a, as.matrix(expand.grid(1:4,1:4,1:4,1:4)))
     # same as `c(a[1:4,1:4,1:4,1:4])' which is same as `c(a)'
```

`extprod3d` *Compute external- or 'cross'- product of 3D vectors.*

Description

Computes the external product

$$(x_2y_3 - x_3y_2,\ x_3y_1 - x_1y_3,\ x_1y_2 - x_2y_1)$$

of the 3D vectors in **x** and **y**.

Usage

```
extprod3d(x, y)
```

Arguments

`x`	n-by-3 matrix. Each row is one **x**-vector
`y`	n-by-3 matrix. Each row is one **y**-vector

Value

n-by-3 matrix

Author(s)

Raoul Grasman

matmax *Row-wise matrix functions*

Description

Compute maximum or minimum of each row, or sort each row of a matrix, or a set of (equal length) vectors.

Usage

```
matmax(...)
```

Arguments

...	A numeric matrix or a set of numeric vectors (that are column-wise bind together into a matrix with cbind).

Value

matmin and matmax return a vector of length nrow(cbind(...)). matsort returns a matrix of dimension dim(cbind(...)) with in each row of cbind(...) sorted. matsort(x) is a lot faster than, e.g., 't(apply(x,1,sort))', if x is tall (i.e., nrow(x)»ncol(x) and ncol(x)<30. If ncol(x)>30 then matsort simply calls 't(apply(x,1,sort))'. matorder returns a permutation which rearranges its first argument into ascending order, breaking ties by further arguments.

Author(s)

Raoul Grasman

Examples

```
example(Unique)
```

mesh.dcircle *Circle distance function*

Description

Signed distance from points p to boundary of circle to allow easy definition of regions in distmesh2d.

Usage

```
mesh.dcircle(p, radius = 1, ...)
```

Arguments

p	A matrix with 2 columns (3 in mesh.dsphere), each row representing a point in the plane.
radius	radius of circle
...	additional arguments (not used)

Value

A vector of length nrow(p) containing the signed distances to the circle

Author(s)

Raoul Grasman; translated from original Matlab sources of Per-Olof Persson.

References

http://www-math.mit.edu/~persson/mesh/

P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB. SIAM Review, Volume 46 (2), pp. 329-345, June 2004

See Also

distmesh2d, mesh.drectangle, mesh.diff, mesh.intersect, mesh.union

Examples

```
example(distmesh2d)
```

---

mesh.diff *Difference, union and intesection operation on two regions*

Description

Compute the signed distances from points p to a region defined by the difference, union or intersection of regions specified by the functions regionA and regionB. regionA and regionB must accept a matrix p with 2 columns as their first argument, and must return a vector of length nrow(p) containing the signed distances of the supplied points in p to their respective regions.

Usage

```
mesh.diff(p, regionA, regionB, ...)
```

Arguments

p	A matrix with 2 columns (3 in mesh.dsphere), each row representing a point in the plane.
regionA	vectorized function describing region A in the union / intersection / difference
regionB	vectorized function describing region B in the union / intersection / difference
...	additional arguments passed to regionA and regionB

Value

A vector of length nrow(p) containing the signed distances to the boundary of the region.

Author(s)

Raoul Grasman; translated from original Matlab sources of Per-Olof Persson.

See Also

distmesh2d, mesh.dcircle, mesh.drectangle mesh.dsphere

---

mesh.drectangle *Rectangle distance function*

Description

Signed distance from points p to boundary of rectangle to allow easy definition of regions in distmesh2d.

Usage

```
mesh.drectangle(p, x1 = -1/2, y1 = -1/2, x2 = 1/2,
  y2 = 1/2, ...)
```

Arguments

p	A matrix with 2 columns, each row representing a point in the plane.
x1	lower left corner of rectangle
y1	lower left corner of rectangle
x2	upper right corner of rectangle
y2	upper right corner of rectangle
...	additional arguments (not used)

Value

a vector of length nrow(p) containing the signed distances

Author(s)

Raoul Grasman; translated from original Matlab sources of Per-Olof Persson.

References

http://www-math.mit.edu/~persson/mesh/

P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB. SIAM Review, Volume 46 (2), pp. 329-345, June 2004

See Also

distmesh2d, mesh.drectangle, mesh.diff, mesh.intersect, mesh.union

Examples

```
example(distmesh2d)
```

---

mesh.dsphere *Sphere distance function*

Description

Signed distance from points p to boundary of sphere to allow easy definition of regions in distmeshnd.

Usage

```
mesh.dsphere(p, radius = 1, ...)
```

Arguments

p	A matrix with 2 columns (3 in mesh.dsphere), each row representing a point in the plane.
radius	radius of sphere
...	additional arguments (not used)

Value

A vector of length nrow(p) containing the signed distances to the sphere

Author(s)

Raoul Grasman; translated from original Matlab sources of Per-Olof Persson.

References

http://www-math.mit.edu/~persson/mesh/

P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB. SIAM Review, Volume 46 (2), pp. 329-345, June 2004

See Also

distmeshnd

Examples

```
example(distmeshnd)
```

---

mesh.hunif *Uniform desired edge length*

Description

Uniform desired edge length function of position to allow easy definition of regions when passed as the fh argument of distmesh2d or distmeshnd.

Usage

```
mesh.hunif(p, ...)
```

Arguments

p	A n-by-m matrix, each row representing a point in an m-dimensional space.
...	additional arguments (not used)

Value

Vector of ones of length n.

Author(s)

Raoul Grasman; translated from original Matlab sources of Per-Olof Persson.

See Also

distmesh2d and distmeshnd.

polyarea *Determines area of a polygon by triangle method.*

Description

Determines area of a polygon by triangle method. The variables x and y define the vertex pairs, and must therefore have the same shape. They can be either vectors or arrays. If they are arrays then the columns of x and y are treated separately and an area returned for each.

Usage

```
polyarea(x, y, d = 1)
```

Arguments

x	X coordinates of verticies.
y	Y coordinates of verticies.
d	Dimension of array to work along.

Details

If the optional dim argument is given, then polyarea works along this dimension of the arrays x and y.

Value

Area(s) of polygon(s).

Author(s)

David Sterratt based on the octave sources by David M. Doolin

Examples

```
x <- c(1, 1, 3, 3, 1)
y <- c(1, 3, 3, 1, 1)
polyarea(x, y)
polyarea(cbind(x, x), cbind(y, y)) ##  c(4, 4)
polyarea(cbind(x, x), cbind(y,y), 1) ##  c(4, 4)
polyarea(rbind(x, x), rbind(y,y), 2) ##  c(4, 4)
```

surf.tri *Find surface triangles from tetrahedra mesh*

Description

Find surface triangles from tetrahedron mesh typically obtained with delaunayn.

Usage

```
surf.tri(p, t)
```

Arguments

p	An n-by-3 matrix. The rows of p represent n points in dim-dimensional space.
t	Matrix with 4 columns, interpreted as output of delaunayn.

Details

surf.tri and convhulln serve a similar purpose in 3D, but surf.tri also works for non-convex meshes obtained e.g. with distmeshnd. It also does not produce currently unavoidable diagnostic output on the console as convhulln does at the Rterm console–i.e., surf.tri is silent.

Value

An m-by-3 index matrix of which each row defines a triangle. The indices refer to the rows in p.

Note

surf.tri was based on matlab code for mesh of Per-Olof Persson (http://www-math.mit.edu/~persson/mesh/index.html).

Author(s)

Raoul Grasman

See Also

tri.mesh, convhulln, surf.tri, distmesh2d

Examples

```
## Not run:
# more extensive example of surf.tri
library(rgl)        # to render tesselation
library(R.matlab)   # to read matlab .mat files

# url's of publically available data:
data1.url = "http://neuroimage.usc.edu/USCPhantom/mesh_data.bin"
data2.url = "http://neuroimage.usc.edu/USCPhantom/CT_PCS_trans.bin"
```

```
meshdata = readMat(url(data1.url))
elec = readMat(url(data2.url))$eeg.ct2pcs/1000
brain = meshdata$mesh.brain[,c(1,3,2)]
scalp = meshdata$mesh.scalp[,c(1,3,2)]
skull = meshdata$mesh.skull[,c(1,3,2)]
tbr = t(surf.tri(brain, delaunayn(brain)))
tsk = t(surf.tri(skull, delaunayn(skull)))
tsc = t(surf.tri(scalp, delaunayn(scalp)))
rgl.triangles(brain[tbr,1], brain[tbr,2], brain[tbr,3],col="gray")
rgl.triangles(skull[tsk,1], skull[tsk,2], skull[tsk,3],col="white", alpha=0.3)
rgl.triangles(scalp[tsc,1], scalp[tsc,2], scalp[tsc,3],col="#a53900", alpha=0.6)
rgl.viewpoint(-40,30,.4,zoom=.03)
lx = c(-.025,.025); ly = -c(.02,.02);
rgl.spheres(elec[,1],elec[,3],elec[,2],radius=.0025,col='gray')
rgl.spheres( lx, ly,.11,radius=.015,col="white")
rgl.spheres( lx, ly,.116,radius=.015*.7,col="brown")
rgl.spheres( lx, ly,.124,radius=.015*.25,col="black")

## End(Not run)
```

`tetramesh` *Render tetrahedron mesh (3D)*

Description

`tetramesh(T, X, col)` uses the rgl package to display the tetrahedrons defined in the m-by-4 matrix T as mesh. Each row of `T` specifies a thetrahedron by giving the 4 indices of its points in `X`.

Usage

```
tetramesh(T, X, col = heat.colors(nrow(T)), clear = TRUE,
  ...)
```

Arguments

`T`	T is a `m`-by-3 matrix in trimesh and `m`-by-4 in tetramesh. A row of `T` contains indices into `X` of the vertices of a triangle/tetrahedron. `T` is usually the output of delaunayn.
`X`	X is an n-by-2/n-by-3 matrix. The rows of X represent `n` points in 2D/3D space.
`col`	The tetrahedron color. See rgl documentation for details.
`clear`	Should the current rendering device be cleared?
`...`	Parameters to the rendering device. See the rgl package.

Author(s)

Raoul Grasman

See Also

trimesh, rgl, delaunayn, convhulln, surf.tri

Examples

```
## Not run:
# example delaunayn
d = c(-1,1)
pc = as.matrix(rbind(expand.grid(d,d,d),0))
tc = delaunayn(pc)

# example tetramesh
library(rgl)
clr = rep(1,3) %o% (1:nrow(tc)+1)
rgl.viewpoint(60,fov=20)
rgl.light(270,60)
tetramesh(tc,pc,alpha=0.7,col=clr)

## End(Not run)
```

---

trimesh *Display triangles mesh (2D)*

Description

trimesh(T, p) displays the triangles defined in the m-by-3 matrix T and points p as a mesh. Each row of T specifies a triangle by giving the 3 indices of its points in X.

Usage

```
trimesh(T, p, p2, add = FALSE, axis = FALSE,
  boxed = FALSE, ...)
```

Arguments

T	T is a m-by-3 matrix. A row of T contains indices into X of the vertices of a triangle. T is usually the output of delaunayn.
p	A vector or a matrix.
p2	if p is not a matrix p and p2 are bind to a matrix with cbind.
add	Add to existing plot in current active device?
axis	Draw axes?
boxed	Plot box?
...	Parameters to the rendering device. See the rgl package.

Author(s)

Raoul Grasman

See Also

tetramesh, rgl, delaunayn, convhulln, surf.tri

Examples

```
#example trimesh
p = cbind(x=rnorm(30), y=rnorm(30))
tt = delaunayn(p)
trimesh(tt,p)
```

---

tsearch *Search for the enclosing Delaunay convex hull*

Description

For `t = delaunay(cbind(x, y))`, where `(x, y)` is a 2D set of points, `tsearch(x, y, t, xi, yi)` finds the index in `t` containing the points `(xi, yi)`. For points outside the convex hull the index is `NA`.

Usage

```
tsearch(x, y, t, xi, yi, bary = FALSE)
```

Arguments

`x`	X-coordinates of triangluation points
`y`	Y-coordinates of triangluation points
`t`	Triangulation, e.g. produced by `t = delaunayn(cbind(x, y))`
`xi`	X-coordinates of points to test
`yi`	Y-coordinates of points to test
`bary`	If `TRUE` return barycentric coordinates as well as index of triangle.

Value

If `bary` is `FALSE`, the index in `t` containing the points `(xi, yi)`. For points outside the convex hull the index is `NA`. If `bary` is `TRUE`, a list containing:

`list("idx")`	the index in `t` containing the points `(xi, yi)`
`list("p")`	a 3-column matrix containing the barycentric coordinates with respect to the enclosing triangle of each point code(xi, yi).

Note

Based on the Octave function Copyright (C) 2007-2012 David Bateman.

Author(s)

David Sterratt

See Also

tsearchn, delaunayn

`tsearchn` *Search for the enclosing Delaunay convex hull*

Description

For `t = delaunayn(x)`, where `x` is a set of points in `d` dimensions, `tsearchn(x, t, xi)` finds the index in `t` containing the points `xi`. For points outside the convex hull, `idx` is `NA`. `tsearchn` also returns the barycentric coordinates `p` of the enclosing triangles.

Usage

```
tsearchn(x, t, xi, fast = TRUE)
```

Arguments

`x`	An `n`-by-`d` matrix. The rows of `x` represent `n` points in `d`-dimensional space.
`t`	A `m`-by-`d+1` matrix. A row of `t` contains indices into `x` of the vertices of a `d`-dimensional simplex. `t` is usually the output of delaunayn.
`xi`	An `ni`-by-`d` matrix. The rows of `xi` represent `n` points in `d`-dimensional space whose positions in the mesh are being sought.
`fast`	If the data is in 2D, use the fast C-based `tsearch` function to produce the results.

Value

A list containing:

`list("idx")`	An `ni`-long vector containing the indicies of the row of `t` in which each point in `xi` is found.
`list("p")`	An `ni`-by-`d+1` matrix containing the barycentric coordinates with respect to the enclosing simplex of each point in `xi`.

Note

Based on the Octave function Copyright (C) 2007-2012 David Bateman.

Author(s)

David Sterratt

See Also

tsearch, delaunayn

`Unique` *Extract Unique Rows*

Description

'Unique' returns a vector, data frame or array like 'x' but with duplicate elements removed.

Usage

```
Unique(X, rows.are.sets = FALSE)
```

Arguments

`X`	Numerical matrix.
`rows.are.sets`	If 'TRUE', rows are treated as sets - i.e., to define uniqueness, the order of the rows does not matter.

Value

Matrix of the same number of columns as `x`, with the unique rows in `x` sorted according to the columns of `x`. If `rows.are.sets = TRUE` the rows are also sorted.

Note

'`Unique`' is (under circumstances) much quicker than the more generic base function '`unique`'.

Author(s)

Raoul Grasman

Examples

```
# `Unique' is faster than `unique'
x = matrix(sample(1:(4*8),4*8),ncol=4)
y = x[sample(1:nrow(x),3000,TRUE), ]
gc(); system.time(unique(y))
gc(); system.time(Unique(y))

#
z = Unique(y)
x[matorder(x),]
z[matorder(z),]
```

Index