Package 'SpatialfdaR'

July 21, 2025

Type Package

Date 2022-10-10

Version 1.0.0

Title Spatial Functional Data Analysis

Author James Ramsay [aut, cre],
Spencer Graves [ctb]

Maintainer James Ramsay <james.ramsay@mcgill.ca>

Depends R (>= 3.5.0), fda, splines, graphics, rgl, geometry

Description Finite element modeling (FEM) uses meshes of triangles to define surfaces.
A surface within a triangle may be either linear or quadratic.
In the order one case each node in the mesh is associated with a basis
function and the basis is called the order one finite element basis.
In the order two case each edge mid-point is also associated with a basis
function. Functions are provided for smoothing, density function estimation
point evaluation and plotting results. Two papers illustrating the
finite element data analysis are Sangalli, L.M., Ramsay, J.O.,
Ramsay, T.O. (2013)<http://www.mox.polimi.it/~sangalli> and
Bernardi, M.S, Carey, M., Ramsay, J. O., Sangalli, L.
(2018)<http://www.mox.polimi.it/~sangalli>. Modelling
spatial anisotropy via regression with partial differential
regularization Journal of Multivariate Analysis, 167, 15-30.

License GPL (>= 2)

URL http://www.functionaldata.org

LazyData true

Imports knitr, rmarkdown

VignetteBuilder knitr

BuildVignettes yes

Language en-US

Suggests spelling

NeedsCompilation no

Repository CRAN

Date/Publication 2022-10-11 09:12:37 UTC

Contents

`create.FEM.basis` *Create a FEM Basis with Triangular Finite Element Basis Functions*

Description

Functional data objects are constructed by specifying a set of basis functions and a set of coefficients defining a linear combination of these basis functions.

The FEM basis is used for functions defined over spatial regions with complicated boundaries. There is an outer boundary outside of which no spatial location is found and there may be, in addition, one or more interior boundaries defining holes inside of which no data are found as well. For example, the outer boundary may define a geographical region, and an inner boundary can define a lake within which no data of interest are found.

The interior of the region not within the holes is subdivided into a set of triangles by a mesh generation procedure. See function `create.mesh.object` for a description of this process.

FEM basis functions are centered on points called nodes that are on a boundary or within the interior region not within holes at points called nodes. Some or all of these nodes are also vertices of the triangles. FEM basis functions are the two-dimensional analogues of B-spline basis functions defined over one dimension. Like splines, each FEM basis function is zero everywhere except in the immediate neighborhood defined by the triangles that share a node, and the nodes play the role of knots for splines.

Like splines, FEM functions are piecewise bivariate polynomials, with a loss of smoothness across edges of triangles. Linear FEM functions are once-differentiable within triangles, and quadratic

functions are twice-differentiable. But both types of function are only continuous across edges. The second example below will highlight these features.

Function `create.FEM.basis` supports only linear or quadratic functions. All nodes for linear basis functions correspond to triangle vertices, and consequently there are three nodes per triangle. For quadratic basis functions, vertices are nodes, but mid-points of edges are also nodes, so that there are six nodes per triangle.

Usage

```
create.FEM.basis(pts, edg=NULL, tri, order=1, nquad=0)
```

Arguments

`pts`	The `nbasis` by 2 matrix of vertices of triangles containing the X- and Y-coordinates of the vertices.
`edg`	The number of edges by 2 matrix defining the segments of the boundary of the region which are also edges of the triangles adjacent to the boundary. The values in matrix `edg` are the indices of the vertices in matrix `pts` of the starting and ending points of the edges.
`tri`	The no. of triangles by 3 matrix specifying triangles and their properties. The indices in `pts` of the vertices of each triangle are in counter-clockwise order.
`order`	The order of the finite element basis functions. This may be either one or two. Order one functions are piecewise linear, and order two functions are piecewise quadratic.
`nquad`	An integer determining the number of quadrature points and weights used to estimate the value of an integral if a function over a triangle using Gaussian quadrature. The number of quadrature points and weights is equal to the square of `nquad`. The `nquad <- 4` usually provides sufficient accuracy for statistical purposes, but the default value of zero implies that no such integration will be needed. Data smoothing does not require integration, but data density estimation does.

Details

The mesh generation step is critical, and if done carelessly or naively, can easily cause the smoothing process to fail, usually due to a singular set of coefficient equations. Careful attention to generating a mesh is required in order to obtain a mesh that is both well-conditioned and has sufficient density to permit a faithful rendition of the smoothing surface required by the data.

Well-conditioned meshes do not have triangles with small angles, and the two mesh-generations functions can allow the user to specify the minimal angle in the mesh.

On the other hand, the finer the mesh, the greater the number of basis functions, so that, as in all smoothing procedures, one wants to avoid meshes so fine that the possibly noisy observations are interpolated rather than smoothed.

Triangles with no data in them are in general to be avoided. In fact, it is common but not necessary that the mesh be constructed so that data observation points are at the vertices of the triangles in the mesh. This special case also leads to very fast computation of the smoothing surface. In some cases, it may make sense to interpolate data to prespecified mesh points before undertaking smoothing.

Piecewise linear basis functions are generally preferable unless estimates of the second partial derivatives of the surface are required because quadratic elements require twice as many basis functions. First order partial derivatives of surfaces are piecewise constant over each triangle. Even though the estimated surface is piecewise linear, its total curvature is still controlled by the size of the smoothing parameter specified in function `smooth.FEM.basis`. The larger the smoothing parameter, the more flat the surface will become.

Value

A `basisfd` object of the type `FEM`. See the help file for `basisfd` for further details about this class.

The `params` slot or member for an FEM basis contains a good deal of information that is used in other functions, such as `smooth.FEM.basis`. Consequently, `basisfd$params` is itself a list with named members, their contents are:

`mesh` an object of class `MESH` specifying the structure of the triangular mesh. See the help file for this class for further details

`order` either 1 for linear elements or 2 for quadratic elements

`nodeList` a `list` object with named members. The list contains information that is required for other functions that may be repeatedly called, such as `smooth.FEM.basis`. The names and their contents are:

1. nodesa K by 2 matrix of coordinates for the K nodes in the mesh.
2. nodesindexthe index of each node
3. Jthe Jacobian for the transformation of each triangle to the standard right triangle
4. metrica three-dimensional array with the length of the leading dimension equal to the number of triangles, and the next two dimensions of length 2. The 2 by 2 matrices are the transformation matrices for mapping each triangle into the standard triangle
5. quadmatNULL if argument `nquad` is zero, or otherwise, a list with a member for each triangle containing the quadrature points and weights for approximating the integral of a function over that triangle. See the help file for function `triquad` for further details.
6. Cart2BaryA three-dimensional array with the leading dimension equal to the number of triangles, and the remaining dimensions of length 3. Each order 3 matrix maps the cartesian coordinate vector `c(1, p(1), p(2))` into the corresponding barycentric coordinates for a triangle, where `p` contains the Cartesian coordinates of a point.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

plotFEM.mesh, smooth.FEM.basis

Examples

```
#  ------------------------------------------------------------------------
#  Example 1:  Set up the simplest possible mesh by hand so as to see
#  the essential elements of a finite element basis
#  Specify the three vertices, in counter-clockwise order, for a single
#  right triangle
#  ------------------------------------------------------------------------
pts <- matrix(c(0, 0, 1, 0, 0, 1),3,2,byrow=TRUE)
#  These points also specify the boundary edges by a 3 by 2 matrix
#  with each row containing the index of the starting point and of the
#  corresponding ending point
edg <- matrix(c(1, 2, 2, 3, 3, 1), 3, 2, byrow=TRUE)
#  The triangles are defined by a 1 by 3 matrix specifying the indices
# of the vertices in counter-clockwise order
tri <- matrix(c(1, 2, 3), 1, 3)
#  FEM basis objects can be either linear (order 1) or quadratice (order 2)
#  both are illustrated here:
#  Set up a FEM basis object of order 1 (piecewise linear) basis functions
order <- 1
FEMbasis1 <- create.FEM.basis(pts, edg, tri, order)
#  display the number of basis functions
print(paste("Number of basis functions =",FEMbasis1$nbasis))
#  plot the basis, the boundary being plotted in red
plotFEM.mesh(pts,tri)
#  Set up an FEM basis object of order 2 (piecewise quadratic) basis functions
order <- 2
FEMbasis2 <- create.FEM.basis(pts, edg, tri, order)
#  display the number of basis functions
print(paste("Number of basis functions =",FEMbasis2$nbasis))
#  plot the basis, the boundary being plotted in red
plotFEM.mesh(pts,tri)
#  ------------------------------------------------------------------------
#  Example 2:  Set up an FEM object with a hexagonal boundary, and a single
#  interior point
#  ------------------------------------------------------------------------
angle <- seq(0,2*pi,len=7)
x     <- cos(angle); y <- sin(angle)
pts   <- rbind(cbind(x[1:6], y[1:6]), c(0, 0))
edg   <- cbind((1:6),c((2:6), 1))
tri   <- matrix(c(7*rep(1,6), 1:6, 2:6, 1),6,3)
hexFEMbasis <- create.FEM.basis(pts, edg, tri)
#  display the number of basis functions
print(paste("Number of basis functions =",hexFEMbasis$nbasis))
#  plot the basis, the boundary being plotted in red
plotFEM.mesh(pts,tri)
```

`eval.FEM.basis` *Values of a Finite Element Functional Data Object*

Description

Evaluate a finite element (FEM) functional data object at specified locations, or evaluate a pair of partial derivatives of the functional object.

Usage

```
eval.FEM.basis(obspts, FEMbasis, nderivs=rep(0,2))
```

Arguments

`obspts`	a two-column matrix of x- and Y-coordinate values at which the functional data object is to be evaluated.
`FEMbasis`	an FEM functional basis object to be evaluated.
`nderivs`	a vector length 2 containing a pair of orders of derivatives. If the FEM basis is piecewise linear (of order 1), the only admissible pairs are (0,0), (0,1) and (1,0). If the FEM basis is piecewise quadratic (of order 2), the pairs (1,1), (0,2) and (2,0) are also admissible.

Details

Two common uses of this function are (1) to evaluate the surface at one or more significant points, and (2) evaluate the surface over a bounding rectangle so that the surface can be plotted as either a contour plot are as a persp type plot. Points that are outside of polygon defining the surface are given the value NA, and are therefore not plotted.

Value

an array of 2 or 3 dimensions containing the surface values. The first two dimension correspond to the arguments `obspts` and, if present, the third dimension corresponds to the surface in the situation where the coefficient matrix has multiple columns.

See Also

`eval.FEM.fd`, `smooth.FEM.basis`

`eval.FEM.fd` *Evaluate a functional data object with an FEM basis.*

Description

A set of points is supplied, and the height of the surface is evaluated at these points.‘

Usage

```
eval.FEM.fd(pts, fdobj, nderivs=rep(0,2))
```

Arguments

`pts`	A two column matrix of pts within a triangular mesh.
`fdobj`	A functional data object with an FEM basis.
“1	
`nderivs`	A vector of length 2 containing derivative orders with respect to the X and Y coordinates of the triangular mesh. The derivative orders are restricted to 0, 1 or 2, and 2 can only be used for a quadratic basis system. The default is both orders being 0.

Value

A matrix with number of rows equal to the number of rows of `pts` containing the values of of one or more surfaces at these points.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

`eval.FEM.basis`

Examples

```
#  ------------------------------------------------------------------------
#  Example 1:  smoothing data over a circular region
#  ------------------------------------------------------------------------
#  Set up a FEM object with a approximatedly circular boundary with 12 sides,
#  and two rings of 12 points plus a point at the centre.
angle = seq(0,11*pi/6,len=12)
```

```
#  define the 24 point locations
rad = 2
ctr = c(0,0)
pts  = matrix(0,25,2);
pts[ 1:12,1] = rad*cos(angle)   + ctr[1]
pts[ 1:12,2] = rad*sin(angle)   + ctr[2]
pts[13:24,1] = rad*cos(angle)/2 + ctr[1]
pts[13:24,2] = rad*sin(angle)/2 + ctr[2]
#  define the edge indices
edg <- matrix(0,12,2)
for (i in 1:11) {
  edg[i,1] <- i
  edg[i,2] <- i+1
}
edg[12,] <- c(12,1)
#  use delaunay triangulation to define the triangles
tri = delaunayn(pts)
plotFEM.mesh(pts, tri, shift = 0.02)
title("A 25-point curcular mesh")
#  Create an order 1 basis
hexFEMbasis <- create.FEM.basis(pts, edg, tri, 1)
#  locations of data
obspts <- pts
npts <- dim(obspts)[1]
points(obspts[,1], obspts[,2], col=2, cex=2, lwd=2)
hexfd = fd(matrix(1-c(obspts[,1]^2+obspts[,2]^2),npts,1),hexFEMbasis)
#  heights of a hemisphere at these locations
hexhgts <- round(eval.FEM.fd(obspts,hexfd),2)
```

`FEMdensity` *Evaluate the function value and gradient for a penalized likelihood estimate of spatial density.*

Description

A spatial density defined over a region with complicated a boundary that is defined by a triangulation is estimated. The basis functions are linear finite elements (FEM) defined at each vertex. The data are the spatial coordinates of a sample of a defined spatial event. The density surface is the logarithm of the density, and the smoothness of the log surface is controlled by a smoothing parameter `lambda` multiplying the stiffness matrix.

Usage

```
FEMdensity(cvec, obspts, FEMbasis, K1=NULL, lambda=0)
```

Arguments

`cvec`	A matrix each column of which contains the coefficients for an FEM basis function expansion of a surface. The number of coefficients equals the number of vertices in the triangulation, also called the nodes of the FEM expansion.

obspts	A two-column matrix containing the locations at which the logarithm of the density is to be evaluated.
FEMbasis	This argument provides the FEM basis (class basisfd), and may be in the form of an FEM basis object, an FEM function object (class fd), or an FEM functional parameter object (class fdPar.
K1	The stiffness matrix associated with the triangulation. It is computed using function .
lambda	A non-negative real number controlling the smoothness of the surface by a roughness penalty consisting of lambda multiplying the stiffness matrix K1.

Details

A probability density surface of a two-dimensional region bounded by line segments is defined by a set of linear or planar triangular surfaces. Each triangular segment is associated with three vertices, and with with three basis functions, each associated with a local linear surface that has height one at one of the vertices and height zero along the two opposing edges. The basis function expansion is actually of the logarithm of the density surface, and has one set of two partial derivatives everywhere except along edges and at vertices of the triangles. The surface is continuous over edges and vertices. The density surface is computed by dividing the exponential of the log surface by the integral of the density over the region, which is returned along with the coefficients of the expansion within a list object. Multiple density surfaces may be estimated, in which case each column of argument cvec is associated with a surface.

Value

F	The penalized log likelihood value associated with the coefficient vector(s).
grad	The gradient vector(s) associated with the coefficient vector(s).
norm	The L2 norm(s) of the gradient vector(s).
Pvec	A matrix, each column of which contains density function values at the nodes or vertices.
Pden	A vector of values of the normalizing constants defined for each surface defined by integrating the exponential of a log surface over the triangular region.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

basisfd, smooth.FEM.density

Examples

```
#  ---------  density on right triangle  --------------
#  Define the locations of the vertices of the right triangle
pts  <- matrix(c(0,0, 1,0, 0,1), 3, 2, byrow=TRUE)
npts <- dim(pts)[1]
#  Specify the single triangle defined by these vertices.
#  The vertices are listed in counter-clockwise order
tri  <- matrix(c(1,2,3),1,3)
ntri <- dim(tri)[1]
#  Set up the FEM basis object for this region
edg <- NULL
RtTriBasis <- create.FEM.basis(pts, edg, tri, 1)
#  List object containing details of nodes.
nodeList <- makenodes(pts,tri,1)
#  number of random points required
Nsample <- 100
#  generate random points ... define the function
obspts <- randomFEMpts(Nsample, pts, tri)
#  Define a starting value for the coefficient vector of length 3
cvec0 <- matrix(rnorm(npts),npts,1)
#  Evaluate the function and gradient vector
result <- FEMdensity(cvec0, obspts, RtTriBasis)
print(paste("Function value =",result$F))
print("gradient:")
print(result$grad)
#  ---------- density on a unit square,  4 triangles, 5 nodes  ----------
#  Generate a unit square with a node in its center defining four
#  triangles and five nodes.
result <- squareMesh(1)
pts <- result$p
edg <- result$e
tri <- result$t
#  plot the mesh
plotFEM.mesh(pts, tri)
npts <- dim(pts)[1]
ntri <- dim(tri)[1]
#  Define the true log density function
SinCosIntensFn <- function(x,y) {
  return(sin(2*pi*x)*cos(2*pi*y))
}
#  Compute a sine*cosine intensity surface.
intDensityVec <- triDensity(pts, tri, SinCosIntensFn)
#  Set up and plot an FEM basis object
SquareBasis1 <- create.FEM.basis(pts, edg, tri)
N <- 100
obspts <- randomFEMpts(N, pts, tri, intDensityVec)
#  plot the random points
points(obspts[,1],obspts[,2])
#  Estimate the smooth density surface with light smoothing
order    <- 1
nodeList <- makenodes(pts,tri,order)
K1       <- stiff.FEM(SquareBasis1)
```

```
lambda   <- 1e-4
#  define a random coefficient vector
cvec     <- rnorm(npts,1)
#  display function value and gradient
result <- FEMdensity(cvec, obspts, SquareBasis1, K1, lambda)
print(paste("Function value =",round(result$F,3)))
print("gradient:")
print(round(result$grad,3))
```

`insideIndex` *Index of the triangle containing a point.*

Description

Evaluate the index of the triangle containing the point `(X,Y)` if such a triangle exists, and `NaN` otherwise.

Usage

```
insideIndex(obspts, pts, tri, tricoef)
```

Arguments

`obspts`	A two-column matrix of observation location points.
`pts`	A two-column matrix of the locations of the vertices of the mesh.
`tri`	A three-column matrix of the indices in `pts` of the triangle vertices.
`tricoef`	Four-column matrix of coefficients for computing barycentric coordinates.

Details

Each triangle is checked to see if the point is contained within it or on its boundary. This is verified if all three of the barycentric coordinates are non-negative.

Value

A vector of integers indicating which triangle in `tri` contains a point.

See Also

`FEMdensity`, `eval.FEM.basis`

Examples

```
#  ---------- density on a unit square,  4 triangles, 5 vertices  ----------
#  Generate a unit square with a node in its center defining four
#  triangles and five nodes.
result <- squareMesh(1)
pts  <- result$p
edg  <- result$e
tri  <- result$t
npts <- dim(pts)[1]
ntri <- dim(tri)[1]
#  define function for sine*cosine function
SinCosIntensFn <- function(x,y) {
  return(sin(2*pi*x)*cos(2*pi*y))
}
#  Compute a sine*cosine intensity surface.
intDensityVec <- triDensity(pts, tri, SinCosIntensFn)
#  Set up and plot an FEM basis object
SquareBasis1 <- create.FEM.basis(pts, edg, tri)
oldpar <- par(cex.lab=2)
plotFEM.mesh(pts, tri)
#  generate random points
N = 1000
obspts <- randomFEMpts(N, pts, tri, intDensityVec)
#  plot the random points
points(obspts[,1],obspts[,2])
#  find the triangle number containing each point
triIndex <- insideIndex(obspts, pts, tri)
par(oldpar)
```

---

makenodes *Compute the matrix* nodes *containing all the nodes in the mesh.*

Description

Each basis function is associated with a node. If the order of the finite elements is 1, the nodes are the vertices of the triangles. If the order is 2, the nodes are also at the midpoints of the edges.

Usage

```
makenodes(pts, tri, order=2)
```

Arguments

pts	A two-column matrix of the locations of the vertices of the mesh.
tri	A three-column matrix of the indices in pts of the triangle vertices.
order	The order of the finite element, which is either 1 or 2.

Details

Computes a matrix nodes containing coordinates for all of the nodes to be used, a matrix nodeindex defining which nodes correspond to each element. If norder is 1, nodes corresponds to vertices and nodeindex is identical to triangles. If norder is 2, the midpoint of each edge is computed and added to points to obtain matrix nodes. The row index of that midpoint is then added to the rows of triangles containing that edge to define nodeindex.

Value

A list object containing these members:

- order The order of the finite elements, which is either 1 or 2.
- nodes A numnodes*2 matrix whose i'th row contains the coordinates of the i'th nodal variable. Nodes for the second order element consist of vertices and midpoints of edges, that is, 6 per triangle. Points are listed first followed by midpoints. Nodes for the first order element consist of only vertices.
- nodeindex If norder == 1, nodeindex is triangles. If norder == 2, an nele*6 matrix whose i'th row contains the row numbers (in nodes) of the nodal variables defining the i'th finite element. If the i'th row of FMESH is [V1 V2 V3] then the i'th row of nodeindex is [V1 V(12) V2 V(23) V3 V(31)], where Vi is the row number of the i'th point and V(ij) is the row number of the midpoint of the edge defined by the i'th and j'th points. If norder == 1, nodeindex is triangles
- jvec Jacobian, which is the area of triangle.
- metric Metric matrix.

See Also

`mass.FEM`, `stiff.FEM`

`mass.FEM` *Compute the mass matrix for a finite element basis.*

Description

mass.FEM produces the mass matrix containing integrals of products of the nodal functions.

Usage

```
mass.FEM(FEMbasis)
```

Arguments

`FEMbasis`	A List object produced by function `create.FEM.basis`. It contains: • orderThe order of the element (1 or 2). • nodesCoordinates of node points. • nodeindexIndices of node points for each element. • jvecJacobian of the affine transformation of each element to the master element.

Value

A matrix `k0`: the `nnod` by `nnod` matrix of sums of products of nodal basis functions. For each element i, the integral of the product of the j'th and k'th shape functions over the i'th element is computed. Then that value is the `(nodeindex[i,j],nodeindex[i,k])` entry of the i'th elemental mass matrix.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

`stiff.FEM`

`MeuseData` *Objects Defining Meuse River Mesh*

Description

The data are the points, edges and triangles required to mesh a bank of the Meuse river in Belgium that is of concern for heavy metal polution. The data to be analyzed are locations and size of zinc assays. A covariate is distances of locations from the river.

Usage

```
MeuseData
```

Format

A named list.

Details

A named list with four members. The number of points is 155, the number of edges is 52, and the number of triangles is 257.

covdata: A 155 by 2 matrix. The first column contains the distance from the river of each location, and the second the amount of zinc detected.

pts: A 155 by 2 matrix containing the locations of the deposits.

edg: A 52 by 2 matrix containing the indices of the points that are the beginning and end of each of the 52 boundary segments.

tri: A 257 by 3 matrix containing the indices of the vertices of each triangle in counter-clockwise order.

plotFEM.fd *Plots an FEM functional data object.*

Description

Plots a FEM object fdobj over a rectangular grid defined by vectors Xgrid and Ygrid.

Usage

```
plotFEM.fd(fdobj, Xgrid=NULL, Ygrid=NULL, nderivs=rep(0,2),
                    xlab="X", ylab="Y", zlab="Z", main="",
                    ticktype="detailed", r=10, phi=15, theta=30)
```

Arguments

fdobj	A surface defined by a finite element FEM basis.
Xgrid	A vector of X coordinate values.
Ygrid	A vector of Y coordinate values.
nderivs	A vector of length 2 indicating order of partial derivatives.
xlab	A character vector for labelling the X-axis when using function persp.
ylab	A character vector for labelling the Y-axis when using function persp.
zlab	A character vector for labelling the Z-axis when using function persp.
main	A character vector for the title when using function persp.
ticktype	A character vector for defining the tick marks when using function persp.
r	A positive number for the distance of the eyepoint from the center of the plot when using function persp.
phi	A number specifying the collatitude when using function persp.
theta	A number specifying the azimuthal direction when using function persp.

Value

No return value, called for side effects.

See Also

plotFEM.mesh, eval.FEM.fd

Examples

```
#  Set up and plot a square with side length one with 32 interior triangles,
#  and then create an order one functional basis object of type FEM.
m <- 4
petList <- squareMesh(m)
pts <- petList$p
edg <- petList$e
```

```
tri <- petList$t
pts <- pts/m
plotFEM.mesh(pts, tri)
norder <- 1
FEMbasis <- create.FEM.basis(pts, edg, tri, norder)
#  set up surface values on the nodes of this mesh
data     <- sin(2*pi*pts[,1])*cos(2*pi*pts[,2])
#  smooth the data
FEMList  <- smooth.FEM.basis(pts, data, FEMbasis)
sincosfd <- FEMList$fd
#  set up a 21 by 21 grid of plotting points
Npts <- 21
Xpts <- seq(0,1,len=Npts)
Ypts <- Xpts
#  plot the surface
oldpar <- par(no.readonly = TRUE)
plotFEM.fd(sincosfd, Xpts, Ypts)
#  plot the X partial derivative of the surface
plotFEM.fd(sincosfd, Xpts, Ypts, c(1,0))
#  plot the Y partial derivative of the surface
plotFEM.fd(sincosfd, Xpts, Ypts, c(0,1))
par(oldpar)
```

`plotFEM.mesh` *Plot a finite element mesh.*

Description

A finite element mesh is a set of triangles defining a two-dimensional region.

Usage

```
plotFEM.mesh(pts, tri, xlabel="x", ylabel="y", xlim=plim1, ylim=plim2,
             shift=0.05, nonum=TRUE)
```

Arguments

`pts`	A two-column matrix of the locations of the vertices of the mesh.
`tri`	A three-column matrix of the indices in `pts` of the triangle vertices.
`xlabel`	A character string for the label of the abscissa or horizontal axis.
`ylabel`	A character string for the label of the ordinate or vertical axis.
`xlim`	A vector of length 2 containing the left and right plotting limits, respectively.
`ylim`	A vector of length 2 containing the bottom and plot plotting limits, respectively.
`shift`	A lateral character shift for text plotting.
`nonum`	A logical constant. If FALSE, numbers for nodes and triangles will not be displayed.

Value

The nodes and mesh lines are plotted. The number of each node and triangle is plotted, with node numbers in black and triangle number in blue.

See Also

plotFEM.fd

Examples

```
#  Set up and plot a square with side length one with 32 interior triangles,
#  and then create an order one functional basis object of type FEM.
m <- 4
petList <- squareMesh(m)
pts <- petList$p
edg <- petList$e
tri <- petList$t
pts <- pts/m
plotFEM.mesh(pts, tri)
```

---

randomFEMpts *Generate Random Locations in a Mesh with a Specified Density*

Description

The probability that a random location is assigned to a triangle is defined by an input vector of probabilities, one per triangle. But if DensityVec = 1, any location within the mesh is accepted.

Usage

```
randomFEMpts(Nsample, pts, tri, logDensityVec=rep(0,ntri))
```

Arguments

Nsample	The number of random locations to be generated.
pts	A two-column matrix of locations of nodes.
tri	A three-column matrix of integers specifying the vertices of triangles in anti-clockwise order.
logDensityVec	A one-column matrix of log probability density values, one for each triangle.

Details

Within triangles, the locations are uniformly distributed. If DensityVec = 1, they are uniformly distributed over the whole mesh.

Value

A two-column matrix with N rows specifying the locations of the random points.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

link{plotFEM.mesh}

Examples

```
#  ---------  density on right triangle  --------------
#  Define the locations of the vertices of the right triangle
pts  <- matrix(c(0,0, 1,0, 0,1), 3, 2, byrow=TRUE)
npts <- dim(pts)[1]
#  Specify the single triangle defined by these vertices.
#  The vertices are listed in counter-clockwise order
tri  <- matrix(c(1,2,3),1,3)
ntri <- dim(tri)[1]
#  Set up the FEM basis object for this region
edg <- NULL
RtTriBasis <- create.FEM.basis(pts, edg, tri, 1)
#  List object containing details of nodes.
nodeList <- makenodes(pts,tri,1)
#  number of random points required
Nsample <- 100
#  generate random points ... define the function
obspts <- randomFEMpts(Nsample, pts, tri)
#  plot the random points
plotFEM.mesh(pts, tri)
points(obspts[,1],obspts[,2])
#  -----------------  density on a circle  ---------------
angle = seq(0,11*pi/6,len=12)
#  define the 24 point locations
rad = 2
ctr = c(0,0)
pts  = matrix(0,25,2);
pts[ 1:12,1] = rad*cos(angle)   + ctr[1]
pts[ 1:12,2] = rad*sin(angle)   + ctr[2]
pts[13:24,1] = rad*cos(angle)/2 + ctr[1]
pts[13:24,2] = rad*sin(angle)/2 + ctr[2]
#  define the edge indices
edg <- matrix(0,12,2)
for (i in 1:11) {
  edg[i,1] <- i
  edg[i,2] <- i+1
}
edg[12,] <- c(12,1)
#  use delaunay triangulation to define the triangles
```

```
tri = delaunayn(pts)
#  plot the triangulation of the circle
plotFEM.mesh(pts, tri, xlim=c(-2,2), ylim=c(-2,2))
#  Define the true log density function
SinCosIntensFn <- function(x,y) {
  return(sin(pi*x/2)*cos(pi*y/2))
}
#  locate observation points with sin*cos log density
intDensityVec <- triDensity(pts, tri, SinCosIntensFn)
#  number of random points required
Nsample <- 100
#  generate random points ... define the function
obspts <- randomFEMpts(Nsample, pts, tri, intDensityVec)
points(obspts[,1], obspts[,2], pch="o")
```

`smooth.FEM.basis` *Construct a functional data object by smoothing spatial data distributed over a region with a complicated boundary using a roughness penalty.*

Description

Discrete observations of a surface are fit with a smooth surface defined by an expansion in a set of FEM type basis functions. The fitting criterion is weighted least squares, and smoothness can be defined in terms of a roughness penalty.

Data smoothing requires at a bare minimum three elements: (1) a set of observed noisy values, (b) a set of argument values associated with these data, and (c) a basis function system used to define the surfaces.

Optionally, a set covariates may be also used to take account of various non-smooth contributions to the data. Smoothing without covariates is often called nonparametric regression, and with covariates is termed semiparametric regression.

Usage

```
smooth.FEM.basis(FEMloc, FEMdata, FEMbasis, lambda=1e-12, wtvec=NULL,
                 covariates=NULL, Laplace=NULL)
```

Arguments

`FEMloc`	A matrix with two columns containing the coordinates of the points where the data are observed.
`FEMdata`	If a single surface, column vector containing the values to be smoothed. Otherwise, a matrix of observation values that is number of pts by number of surfaces.
`FEMbasis`	Either (1) a functional basis object of the `FEM` type if the first argument contains the X-coordinates, or, if the first argument is a matrix with three columns, an `N` by `q` matrix of covariate values, where `N` is the number of observations and `q` is the number of covariates.

lambda	Either (1) a nonnegative smoothing parameter value that controls the amount of penalty on the curvature of the fitted surface, or (2) a vector of weights for the observations. The default value for lambda is 1e-12, a value too small to noticeably affect the curvature of the fitted surface, but large enough to ensure nonsingularity in the equations defining the coefficients of the basis function expansion.
wtvec	a vector of length N that is the length of Xvec containing weights for the values to be smoothed, However, it may also be a symmetric matrix of order n. wt defaults to all weights equal to 1.
covariates	An N by q matrix of covariate values, one for each covariate and each observed value. By default this argument is NULL and no covariates are used to define the surface.
Laplace	If TRUE, the Laplacian surface is computed, otherwise not.

Details

The surface fitted to the data by smooth.FEM.basis is an expansion in terms of finite element basis functions defined by a triangular mesh of points defining the region over which the smooth is defined. The mesh also determines the location and shape of the basis functions. The mesh therefore plays a role that is like that of knots for B-spline basis functions, but is perhaps even more critical to the success of the smoothing process.

Designing and refining the mesh is therefore a preliminary step in finite element smoothing, and requires considerable care and effort. Consult the help pages for MESH and create.FEM.basis for more details.

A roughness penalty is a quantitative measure of the roughness of a surface that is designed to fit the data. For this function, this penalty consists of the product of two parts. The first is an approximate integral over the argument range of the square of a derivative of the surface. A typical choice of derivative order is 2, whose square is often called the local curvature of the function. Since a rough function has high curvature over most of the function's range, the integrated square of of the second derivative quantifies the total curvature of the function, and hence its roughness. The second factor is a positive constant called the bandwidth of smoothing parameter, and given the variable name lambda here.

The roughness penalty is added to the weighted error sum of squares and the composite is minimized, usually in conjunction with a high dimensional basis expansion such as a spline function defined by placing a knot at every observation point. Consequently, the smoothing parameter controls the relative emphasis placed on fitting the data versus smoothness; when large, the fitted surface is more smooth, but the data fit worse, and when small, the fitted surface is more rough, but the data fit much better. Typically smoothing parameter lambda is manipulated on a logarithmic scale by, for example, defining it as a power of 10.

A good compromise lambda value can be difficult to define, and minimizing the generalized cross-validation or "gcv" criterion that is output by smooth.FEM.basis is a popular strategy for making this choice, although by no means foolproof. One may also explore lambda values for a few log units up and down from this minimizing value to see what the smoothing function and its derivatives look like. The function plotfit.fd was designed for this purpose.

The size of common logarithm of the minimizing value of lambda can vary widely, and spline functions depends critically on the typical spacing between knots. While there is typically a "natural"

measurement scale for the argument, such as time in milliseconds, seconds, and so forth, it is better from a computational perspective to choose an argument scaling that gives knot spacings not too different from one.

Value

an object of class `fdSmooth`, which is a named list of length 8 with the following components:

`fd`	an FEM functional data object containing a smooth of the data
`df`	a degrees of freedom measure of the smooth
`gcv`	the value of the generalized cross-validation or GCV criterion. If there are multiple surfaces, this is a vector of values, one per surface. If the smooth is multivariate, the result is a matrix of gcv values, with columns corresponding to variables. $$gcv = n * SSE/((n - df)^2)$$
`beta`	the regression coefficients associated with covariate variables. These are vector, matrix or array objects depending on whether there is a single surface, multiple surfaces or multiple surfaces and variables, respectively.
`SSE`	the error sums of squares. SSE is a vector or a matrix of the same size as GCV.
`penmat`	the penalty matrix

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

`create.FEM.basis`, `lambda2df`, `lambda2gcv`, `df2lambda`, `plotFEM.fd`, `plotFEM.mesh`, `smooth.FEM.basis`

Examples

```
#  ------------------------------------------------------------------------
#  Example 1:  smoothing data over a circular region
#  ------------------------------------------------------------------------
#  Set up a FEM object with a approximatedly circular boundary with 12 sides,
#  and two rings of 12 points plus a point at the centre.
angle = seq(0,11*pi/6,len=12)
#  define the 24 point locations
rad = 2
ctr = c(0,0)
pts  = matrix(0,25,2)
pts[ 1:12,1] = rad*cos(angle)   + ctr[1]
pts[ 1:12,2] = rad*sin(angle)   + ctr[2]
```

```
pts[13:24,1] = rad*cos(angle)/2 + ctr[1]
pts[13:24,2] = rad*sin(angle)/2 + ctr[2]
npts = nrow(pts)
#  define the edge indices
edg <- matrix(0,12,2)
for (i in 1:11) {
  edg[i,1] <- i
  edg[i,2] <- i+1
}
edg[12,] <- c(12,1)
#  use delaunay triangulation to define the triangles
#  These geometry and RTriangle packages give different but legitimate answers
tri_GM <- geometry::delaunayn(pts)
plotFEM.mesh(pts, tri_GM, nonum=FALSE, shift = 0.1)
FEMbasis_GM <- create.FEM.basis(pts, edg, tri_GM, 1)
ntri <- nrow(tri_GM)
#  plot the two meshes
plotFEM.mesh(pts, tri_GM, nonum=FALSE, shift = 0.1)
title("A 25-point circular mesh from geometry")
#  set up the FEM basis objects
FEMbasis_GM <- create.FEM.basis(pts, edg, tri_GM, 1)
#  locations of locations and data  (kept same for both triangulations)
nobs <- 201
FEMloc <- randomFEMpts(nobs, pts, tri_GM)
#  heights of a hemisphere at these locations (kept same for both triangulations)
sig <- 0.2
FEMdata <- sqrt(4 - FEMloc[,1]^2 - FEMloc[,2]^2) +
  matrix(rnorm(nobs),nobs,1)*sig
#  Smooth the data
FEMList_GM <- smooth.FEM.basis(FEMloc, FEMdata, FEMbasis_GM, lambda=1)
FEMfd_GM <- FEMList_GM$fd
round(FEMList_GM$SSE,3)
round(FEMList_GM$df,3)
Xgrid = seq(-2,2,len=21)
Ygrid = seq(-2,2,len=21)
#  plot the result
plotFEM.fd(FEMfd_GM, Xgrid, Ygrid,
          main="A hemisphere over a 25-point circle")
```

`smooth.FEM.density` *Compute a smooth FEM density surface of a triangulated region.*

Description

A spatial density defined over a region with complicated a boundary that is defined by a triangulation is estimated. The basis functions are linear finite elements (FEM) defined at each vertex. The data are the spatial coordinates of a sample of a defined spatial event. The density surface is the logarithm of the density, and the smoothness of the log surface is controlled by a smoothing parameter `lambda` multiplying the stiffness matrix.

Usage

```
smooth.FEM.density(obspts, cvec, FEMbasis, K1=NULL, lambda=0,
                   conv=1e-4, iterlim=50, dbglev=FALSE)
```

Arguments

obspts	A two-column matrix with each row corresponding to a location within a two-dimensional region bounded by line segments and containing a triangular mesh.
cvec	A matrix each column of which contains the coefficients for an FEM basis function expansion of a surface. The number of coefficients equals the number of vertices in the triangulation, also called the nodes of the FEM expansion.
FEMbasis	This argument provides the FEM basis (class basisfd), and may be in the form of an FEM basis object, an FEM function object (class fd), or an FEM functional parameter object (class fdPar.
K1	The stiffness matrix associated with the triangulation. It is computed using function stiff.FEM.
lambda	A non-negative real number controlling the smoothness of the surface by a roughness penalty consisting of lambda.
conv	The convergence criterion using the function value.
iterlim	The maximum number of permitted iterations.
dbglev	print debugging output.

Details

The penalized log likelihood of a density surface is minimized with respect to the coefficient vector. with the initial value in cvec. The spatial event locations are in the two-column matrix pts. The FEM basis object is extracted from IntensityfdPar, which may also be an FEM functional data object or an FEM basis object. The roughness penalty, if required, is the stiff matrix K1 multiplied by the roughness parameter lambda.

Value

cvec:	The final coefficient vector or matrix.
Intensityfd:	An FEM functional data object for the log density.
Flist:	A list object with F the final penalized log likelihood value, grad the final gradient and norm the final norm of the gradient.
iterhist:	A matrix with three columns displaying the iteration number, the function value and the norm of the gradient vector for the initial iteration and each subsequent iteration.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

link{FEMdensity}

Examples

```
#  ---------  Density on right triangle  --------------

#  Define the locations of the vertices of the right triangle
pts <- matrix(c(0,0, 1,0, 0,1), 3, 2, byrow=TRUE)
npts <- dim(pts)[1]
#  Specify the single triangle defined by these vertices.
#  The vertices are listed in counter-clockwise order
tri  <- matrix(c(1,2,3),1,3)
ntri <- dim(tri)[1]
#  Set up the FEM basis object for this region
RtTriBasis <- create.FEM.basis(pts, NULL, tri, 1)
plotFEM.mesh(pts, tri, RtTriBasis)
#  List object containing details of nodes.
nodeList <- makenodes(pts,tri,1)
K1 <- stiff.FEM(RtTriBasis)
#  Define the true log density, which is flat with height 0
ZeroFn <- function(x,y) {
  xdim <- dim(x)
  ZeroIntens <- matrix(0,xdim[1],xdim[1])
  return(ZeroIntens)
}
#  Compute of probabilities for each triangle of having a
#  location withinx it.
intDensityVec <- triDensity(pts, tri, ZeroFn)
#  number of random points required
N <- 100
#  generate random points ... define the function
obspts <- randomFEMpts(N, pts, tri, intDensityVec)
#  plot the random points
points(obspts[,1],obspts[,2])
#  Define a starting value for the coefficient vector of length 3
cvec <- matrix(0,npts,1)
#  Estimate the smooth density surface with no smoothing
densityList <- smooth.FEM.density(obspts, cvec, RtTriBasis, dbglev=2, iterlim=10)
#  The estimate of the coefficient vector
cvec <- densityList$cvec
#  Define the log density FEM functional data object
lnlamfd  <- fd(cvec,  RtTriBasis)
#  Plot the log density surface
Xgrid <- seq(0,1,len=51)
Ygrid <- Xgrid
plotFEM.fd(lnlamfd, Xgrid, Ygrid)
```

```
#  Plot the log density surface
plotFEM.fd(lnlamfd, Xgrid, Ygrid)
#  Plot the log density surface using function contour
logdensitymat <- matrix(0,51,51)
for (i in 1:51) {
  for (j in 1:51) {
    logdensitymat[i,j] <- eval.FEM.fd(matrix(c(Xgrid[i],Ygrid[j]),1,2),lnlamfd)
  }
}
contour(Xgrid, Ygrid, logdensitymat)
#  Plot the density surface using function contour
densitymat <- matrix(0,51,51)
for (i in 1:51) {
  for (j in 1:51) {
    densitymat[i,j] <- exp(eval.FEM.fd(matrix(c(Xgrid[i],Ygrid[j]),1,2),lnlamfd))
  }
}
contour(Xgrid, Ygrid, densitymat)

#  ---------- density on a unit square,  4 triangles, 5 vertices  ----------

#  Generate a unit square with a node in its center defining four
#  triangles and five nodes.
result <- squareMesh(1)
pts    <- result$p
edg    <- result$e
tri    <- result$t
npts   <- dim(pts)[1]
ntri   <- dim(tri)[1]
#  Compute a sine*cosine intensity surface.
SinCosIntensFn <- function(x,y) {
  return(sin(2*pi*x)*cos(2*pi*y))
}
logDensityVec <- triDensity(pts, tri, SinCosIntensFn)
#  Set up and plot an FEM basis object
par(cex.lab=2)
SquareBasis1 <- create.FEM.basis(pts, edg, tri)
plotFEM.mesh(pts, tri)
#  generate random points
N = 100
obspts <- randomFEMpts(N, pts, tri, logDensityVec)
#  plot the random points
points(obspts[,1],obspts[,2])
#  Estimate the smooth density surface with light smoothing
order <- 1
nodeList <- makenodes(pts,tri,order)
K1 <- stiff.FEM(SquareBasis1)
lambda <- 1e-4
#  define a random coefficient vector
cvec  <- matrix(0,npts,1)
#  Estimate the smooth density surface with no smoothing
densityList <- smooth.FEM.density(obspts, cvec, SquareBasis1, dbglev=2, iterlim=10)
#  The estimate of the coefficient vector
```

```
cvec <- densityList$cvec
#  Define the log density FEM functional data object
lnlamfd  <- fd(cvec,  SquareBasis1)
#  Plot the log density surface
Xgrid <- seq(0,1,len=51)
Ygrid <- Xgrid
plotFEM.fd(lnlamfd, Xgrid, Ygrid)
#  Plot the log density surface
plotFEM.fd(lnlamfd, Xgrid, Ygrid)
#  Plot the log density surface using function contour
logdensitymat <- matrix(0,51,51)
for (i in 1:51) {
  for (j in 1:51) {
    logdensitymat[i,j] <- eval.FEM.fd(matrix(c(Xgrid[i],Ygrid[j]),1,2),lnlamfd)
  }
}
contour(Xgrid, Ygrid, logdensitymat)
#  Plot the density surface using function contour
densitymat <- matrix(0,51,51)
for (i in 1:51) {
  for (j in 1:51) {
    densitymat[i,j] <- exp(eval.FEM.fd(matrix(c(Xgrid[i],Ygrid[j]),1,2),lnlamfd))
  }
}
contour(Xgrid, Ygrid, densitymat)
```

`squareMesh` *Generate a Triangulation of a Square.*

Description

The square is subdivided into `m` internal squares. Within each internal square nodes are set at the corners and the middle. Each square is thus subdivided into four triangles.

Usage

```
squareMesh(m=1)
```

Arguments

`m`	The number of squares within the outer square.

Details

The outer square has sides of length `m`. If another length $x > 0$ of the side is required, this can easily be achieved by the command `p = p*x/m`.

Value

A named list containing locations of nodes and which nodes define the edge segments and the triangles. The members are:

- p: A two-column matrix specifying the nodes of the mesh.
- e: A two-column matrix of integers specifying which nodes define each edge segment.
- t: A three-column matrix of integers specifying which nodes define each triangle in anti-clockwise order.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

link{plotFEM.mesh}

Examples

```
m <- 3
#  A square of size 3 with 9 internal squares,
#  25 nodes, 12 edge segments and 36 triangles.
result = squareMesh(m)
pts    = result$pts
edg    = result$edg
tri    = result$tri
#  plot the mesh
plotFEM.mesh(pts,tri)
```

---

squareMesh_RL *Generate a Triangulation of a Square.*

Description

The square is subdivided into m internal squares. Within each internal square, nodes are set at the corners. Each square is subdivided into two triangles by a line either going from lower left to upper right if orientation="L", of by a line from lower right to upper left is orientation="R".

Usage

```
squareMesh_RL(m=1,orientation="L")
```

Arguments

m	The number of squares within the outer square.
orientation	The position of the lower corner of the diagonal line subdividing each square. This must be either "L" or "R".

Details

The outer square has sides of length m. If another length $x > 0$ of the side is required, this can easily be achieved by the command p = p*x/m.

Value

A named list containing locations of nodes and which nodes define the edge segments and the triangles. The members are:

- p: A two-column matrix specifying the nodes of the mesh.
- e: A two-column matrix of integers specifying which nodes define each edge segment.
- t: A three-column matrix of integers specifying which nodes define each triangle in anti-clockwise order.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

link{plotFEM.mesh}

---

stiff.FEM *Compute the stiffness matrix for a finite element basis.*

Description

stiff.FEM produces the stiffness matrix containing integrals of products of second partial derivatives of the nodal functions.

Usage

```
stiff.FEM(FEMbasis)
```

Arguments

FEMbasis	A List object produced by function create.FEM.basis. It contains:

- orderThe order of the element (1 or 2).
- nodesCoordinates of node points.
- nodeindexIndices of node points for each element.
- jvecjacobian of the affine transformation of each element to the master element.

Value

A matrix K0: the NNOD by NNOD matrix of sums of products of nodal basis functions. For each element i, the integral of the product of the j'th and k'th second partial derivatives of the shape functions over the i'th element is computed. Then that value is the (nodeindex[i,j],nodeindex[i,k])th entry of the i'th elemental stiff matrix.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

mass.FEM

---

tricoefCal *Compute the coefficient matrix required to test of a point is inside a triangle.*

Description

Compute the coefficient matrix tricoef required to test if a point is inside a triangle.

Usage

```
tricoefCal(pts, tri)
```

Arguments

pts	A two-column matrix of the locations of the vertices of the mesh.
tri	A three-column matrix of the indices in pts of the triangle vertices.

Value

A matrix `tricoef`.

See Also

`insideIndex`

`triDensity` *Compute the probabilities that a random location will be within one of the triangles of an FEM mesh.*

Description

Given a function defining a log-density surface for an FEM density, compute the probabilities that a random location will be within each of the triangles of an FEM mesh. This involves numerical quadrature over each rectangle, and `nquad` is the order of approximation.

Usage

```
triDensity(pts, tri, logIntensFn, nquad=5)
```

Arguments

`pts`	A two-column matrix with each row corresponding to a location within a two-dimensional region bounded by line segments and containing a triangular mesh.
`tri`	A three-column matrix indexing for each triangle its vertices.
`logIntensFn`	A function of `x` and `y` giving the surface height for an unnormalized log density function. It must return a square matrix of order equal to `nquad`. This implies that any multiplications must be pointwise.
`nquad`	The order of the quadrature over a triangle. The default order of five is good for most applications.

Value

A vector of length equal to the number of triangles in a mesh containing the probabilities that a random observation will fall within the triangles.

Author(s)

Jim Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, Journal of the Royal Statistical Society, Series B, 75, 681-703.

See Also

smooth.FEM.density

Examples

```
#  ---------  right triangle mesh  with zero log density --------------
#  Define the locations of the vertices of the right triangle
pts  <- matrix(c(0,0, 1,0, 0,1), 3, 2, byrow=TRUE)
npts <- dim(pts)[1]
#  Specify the single triangle defined by these vertices.
#  The vertices are listed in counter-clockwise order
tri  <- matrix(c(1,2,3),1,3)
ntri <- dim(tri)[1]
#  Set up the FEM basis object for this region
edg <- NULL
RtTriBasis <- create.FEM.basis(pts, edg, tri, 1)
#  set up function to provide log intensity zero
ZeroFn <- function(x,y) {
  xdim <- dim(x)
  ZeroIntens <- matrix(0,xdim[1],xdim[1])
  return(ZeroIntens)
}
#  Define the true log density, which is flat with height 0
intDensityVec <- triDensity(pts, tri, ZeroFn)
#  ---------  square mesh with sin*cos log density  --------------
nsquare <- 3
result <- squareMesh(nsquare)
pts <- result$p
tri <- result$t
pts <- pts/nsquare
#  Set up and plot an FEM basis object
edge <- NULL
SquareBasis3 <- create.FEM.basis(pts, edg, tri)
plotFEM.mesh(pts, tri)
#  set up function to provide log intensity sine*cosine
SinCosIntensFn <- function(x,y) {
  scale <- 1
  SinCosIntens <- scale*sin(2*pi*x)*cos(2*pi*y)
  return(SinCosIntens)
}
#  Computation of probabilities for each triangle of having a
#  location within it.
intDensityVec <- triDensity(pts, tri, SinCosIntensFn)
#  Display triangle numbers with probabilities
print(cbind(matrix(1:ntri,ntri,1), round(intDensityVec,3)))
```

---

triquad *Set up Gaussian quadrature points and weights for a triangular domain.*

Description

The integral of a function over a triangle is approximated by a weighted sum of the values of the function at a set of points.

Usage

```
triquad(nquad, v)
```

Arguments

nquad	The number of quadrature points and weights is N^2.
v	A matrix with three rows and two columns containing the locations of the vertices of the triangle.

Details

Gaussian quadrature approximates an integral of a function f over a triangle by a weighted sum of N^2 values of f at specified points within the triangle. The larger N, the more accurate the approximation, but course the longer it takes to compute. For many purposes, including most uses of finite element methods, the accuracy does not have to be great, and N=5 may well suffice

Value

A list object containing these named fields:

X	The X-coordinates of the quadrature points.
Y	The Y-coordinates of the quadrature points.
Wx	The weights for the X-coordinates of the quadrature points.
Wy	The weights for the Y-coordinates of the quadrature points.

Author(s)

J. O. Ramsay

References

Sangalli, Laura M., Ramsay, James O., Ramsay, Timothy O. (2013), Spatial spline regression models, *Journal of the Royal Statistical Society, Series B*, 75, 681-703.

Index