Package ‘MBA’

October 12, 2022

Version 0.0-9
Date 2017-02-20
Encoding UTF-8
Title Multilevel B-Spline Approximation
Depends R (> 2.10.0)
Suggests sp
LinkingTo BH

Description Functions to interpolate irregularly and regularly spaced data using Multilevel B-spline Approximation (MBA). Functions call portions of the SINTEF Multilevel B-spline Library written by Øyvind Hjelle which implements methods developed by Lee, Woldberg and Shin (1997; <doi:10.1109/2945.620490>).

License GPL (>= 2)
NeedsCompilation yes
Author Andrew Finley [aut, cre],
Sudipto Banerjee [aut],
Øyvind Hjelle [aut],
Roger Bivand [ctb]
Maintainer Andrew Finley <finleya@msu.edu>
Repository CRAN
Date/Publication 2017-03-08 08:22:04

R topics documented:

LIDAR ................................................................. 2
mba.points ......................................................... 2
mba.surf ............................................................. 4

Index 7
LIDAR

**Canopy LIDAR data**

**Description**
This is a small portion of Light Detection and Ranging (LIDAR) data taken over a forested landscape in Wisconsin, USA.

**Usage**
data(LIDAR)

**Format**
A data frame containing 10123 rows and 3 columns corresponding to longitude, latitude, and elevation.

**Source**
Data provided by: Dr. Paul V. Bolstad, Department of Forest Resources, University of Minnesota, <pbolstad@umn.edu>

---

**mba.points**

**Point approximation from bivariate scattered data using multilevel B-splines**

**Description**
The function `mba.points` returns points on a surface approximated from a bivariate scatter of points using multilevel B-splines.

**Usage**

```r
mba.points(xyz, xy.est, n = 1, m = 1, h = 8, extend = TRUE, 
verbose = TRUE, ...)```

**Arguments**

- `xyz` a \( n \times 3 \) matrix or data frame, where \( n \) is the number of observed points. The three columns correspond to point x, y, and z coordinates. The z value is the response at the given x, y coordinates.

- `xy.est` a \( p \times 2 \) matrix or data frame, where \( p \) is the number of points for which to estimate a z. The two columns correspond to x, y point coordinates where a z estimate is required.
mba.points

n
initial size of the spline space in the hierarchical construction along the x axis. If the rectangular domain is a square, n = m = 1 is recommended. If the x axis is k times the length of the y axis, n = 1, m = k is recommended. The default is n = 1.

m
initial size of the spline space in the hierarchical construction along the y axis. If the y axis is k times the length of the x axis, m = 1, n = k is recommended. The default is m = 1.

h
Number of levels in the hierarchical construction. If, e.g., n = m = 1 and h = 8, the resulting spline surface has a coefficient grid of size \(2^h + 3 = 259\) in each direction of the spline surface. See references for additional information.

extend
if FALSE, points in \(xy\).est that fall outside of the domain defined by \(xyz\) are set to NA with a warning; otherwise, the domain is extended to accommodate points in \(xy\).est with a warning.

verbose
if TRUE, warning messages are printed to the screen.

Value
List with 1 component:

\(xyz\).est
a \(p \times 3\) matrix. The first two columns are \(xy\).est and the third column is the corresponding z estimates.

Note
The function mba.points relies on the Multilevel B-spline Approximation (MBA) algorithm. The underlying code was developed at SINTEF Applied Mathematics by Dr. Øyvind Hjelle. Dr. Øyvind Hjelle based the algorithm on the paper by the originators of Multilevel B-splines:


For additional documentation and references see:

See Also
mba.surf

Examples

data(LIDAR)

# split the LIDAR dataset into training and validation sets
tr <- sample(1:nrow(LIDAR), trunc(0.5*nrow(LIDAR)))

# look at how smoothing changes z-approximation,
# careful the number of B-spline surface coefficients
# increases at \(2^h\) in each direction
for(i in 1:10){
```r
mba.pts <- mba.points(LIDAR[tr,], LIDAR[-tr,c("x","y")], h=i)$xyz.est
print(sum(abs(LIDAR[-tr,"z"]-mba.pts[,"z"]))/nrow(mba.pts))
}

## Not run:
## rgl or scatterplot3d libraries can be fun
library(rgl)

## exaggerate z a bit for effect and take a smaller subset of LIDAR
LIDAR[,"z"] <- 10*LIDAR[,"z"]
tr <- sample(1:nrow(LIDAR), trunc(0.99*nrow(LIDAR)))

## get the "true" surface
mba.int <- mba.surf(LIDAR[tr,], 100, 100, extend=TRUE)$xyz.est
open3d()
surface3d(mba.int$x, mba.int$y, mba.int$z)

## now the point estimates
mba.pts <- mba.points(LIDAR[tr,], LIDAR[-tr,c("x","y")])$xyz.est
spheres3d(mba.pts[,"x"], mba.pts[,"y"], mba.pts[,"z"],
   radius=5, color="red")

## End(Not run)
```

---

**mba.surf**

Surface approximation from bivariate scattered data using multilevel B-splines

---

**Description**

The function `mba.surf` returns a surface approximated from a bivariate scatter of data points using multilevel B-splines.

**Usage**

```r
mba.surf(xyz, no.X, no.Y, n = 1, m = 1, h = 8, extend=FALSE, sp=FALSE, ...)
```

**Arguments**

- **xyz**
  
a $n \times 3$ matrix or data frame, where $n$ is the number of observed points. The three columns correspond to point x, y, and z coordinates. The z value is the response at the given x, y coordinates.

- **no.X**
  
  resolution of the approximated surface along the x axis.

- **no.Y**
  
  resolution of the approximated surface along the y axis.
The function `mba.surf` relies on the Multilevel B-spline Approximation (MBA) algorithm. The underlying code was developed at SINTEF Applied Mathematics by Dr. Øyvind Hjelle. Dr. Øyvind Hjelle based the algorithm on the paper by the originators of Multilevel B-splines:


For additional documentation and references see:

See Also

mba.points

Examples

## Not run:
data(LIDAR)

mba.int <- mba.surf(LIDAR, 300, 300, extend=TRUE)$xyz.est

##Image plot
image(mba.int, xaxs="r", yaxs="r")

##Perspective plot
persp(mba.int, theta = 135, phi = 30, col = "green3", scale = FALSE,
     ltheta = -120, shade = 0.75, expand = 10, border = NA, box = FALSE)

##For a good time I recommend using rgl
library(rgl)

ex <- 10
x <- mba.int[[1]]
y <- mba.int[[2]]
z <- ex*mba.int[[3]]
zlim <- range(z)
colorlut <- heat.colors(as.integer(zlen))
col <- colorlut[ z-zlim[1]+1 ]

open3d()
surface3d(x, y, z, color=col, back="lines")

## End(Not run)
Index

* datasets
  LIDAR, 2
* dplot
  mba.points, 2
  mba.surf, 4
* smooth
  mba.points, 2
  mba.surf, 4

LIDAR, 2

mba.points, 2, 6
mba.surf, 3, 4