The akima Package
February 3, 2006

Version 0.5-1
Date 2006-02-01
Title Interpolation of irregularly spaced data
Author Fortran code by H. Akima R port by Albrecht Gebhardt <albrecht.gebhardt@uni-klu.ac.at>
aspline function by Thomas Petzoldt <petzoldt@rcs.urz.tu-dresden.de> enhancements and
corrections by Martin Maechler <maechler@stat.math.ethz.ch>
Maintainer Albrecht Gebhardt <albrecht.gebhardt@uni-klu.ac.at>
Description Linear or cubic spline interpolation for irregular gridded data
License Fortran code: ACM, free for non-commercial use, R functions GPL

R topics documented:

| akima | Waveform Distortion Data for Bivariate Interpolation |

Description

Hiroshi Akima, "A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly
Distributed Data Points", ACM Transactions on Mathematical Software, Vol. 4, No. 2, June 1978,
The data was taken from a study of waveform distortion in electronic circuits, described in: Hi-
roshi Akima, "A Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local

Arguments

\[x, y, z\]

represents a smooth surface of \(z\) values at selected points irregularly distributed
in the \(x-y\) plane.
Examples

```r
## Not run:
library(rgl)
data(akima)
# data
rgl.spheres(akima$x, akima$z, akima$y, 0.5, color="red")
rgl.bbox()
# bivariate linear interpolation
# interp:
akima.li <- interp(akima$x, akima$y, akima$z,
                   xo=seq(min(akima$x), max(akima$x), length = 100),
                   yo=seq(min(akima$y), max(akima$y), length = 100))
# interp surface:
rgl.surface(akima.li$x, akima.li$y, akima.li$z, color="green", alpha=c(0.5))
# interpp:
akima.p <- interpp(akima$x, akima$y, akima$z,
                    runif(200, min(akima$x), max(akima$x)),
                    runif(200, min(akima$y), max(akima$y)))
# interpp points:
rgl.points(akima.p$x, akima.p$z, akima.p$y, size=4, color="yellow")

# bivariate spline interpolation
# data
rgl.spheres(akima$x, akima$z, akima$y, 0.5, color="red")
rgl.bbox()
# bivariate cubic spline interpolation
# interp:
akima.si <- interp(akima$x, akima$y, akima$z,
                   xo=seq(min(akima$x), max(akima$x), length = 100),
                   yo=seq(min(akima$y), max(akima$y), length = 100),
                   linear = FALSE, extrap = TRUE)
# interp surface:
rgl.surface(akima.si$x, akima.si$y, akima.si$z, color="green", alpha=c(0.5))
# interpp:
akima.sp <- interpp(akima$x, akima$y, akima$z,
                     runif(200, min(akima$x), max(akima$x)),
                     runif(200, min(akima$y), max(akima$y)),
                     linear = FALSE, extrap = TRUE)
# interpp points:
rgl.points(akima.sp$x, akima.sp$z, akima.sp$y, size=4, color="yellow")
## End(Not run)
```

### aspline

**Univariate Akima interpolation**

**Description**

The function returns a list of points which smoothly interpolate given data points, similar to a curve drawn by hand.

**Usage**

```r
aspline(x, y=NULL, xout, n = 50, ties = mean, method="original", degree=3)
```
Arguments

x, y vectors giving the coordinates of the points to be interpolated. Alternatively a single plotting structure can be specified: see `xy.coords`.

xout an optional set of values specifying where interpolation is to take place.

n If xout is not specified, interpolation takes place at n equally spaced points spanning the interval [min(x), max(x)].

ties Handling of tied x values. Either a function with a single vector argument returning a single number result or the string "ordered".

method either "original" method after Akima (1970) or "improved" method after Akima (1991)

degree if improved algorithm is selected: degree of the polynomials for the interpolating function

Details

The original algorithm is based on a piecewise function composed of a set of polynomials, each of degree three, at most, and applicable to successive interval of the given points. In this method, the slope of the curve is determined at each given point locally, and each polynomial representing a portion of the curve between a pair of given points is determined by the coordinates of and the slopes at the points.

Value

A list with components x and y, containing n coordinates which interpolate the given data points.

References


See Also

approx, spline

Examples

### regular spaced data

```r
x <- 1:10
y <- c(rnorm(5), c(1,1,1,1,3))
xnew <- seq(-1, 11, 0.1)
plot(x, y, ylim=c(-3, 3), xlim=range(xnew))
lines(spline(x, y, xmin=min(xnew), xmax=max(xnew), n=200), col="blue")
lines(aspline(x, y, xnew), col="red")
lines(aspline(x, y, xnew, method="improved"), col="black", lty="dotted")
lines(aspline(x, y, xnew, method="improved", degree=10), col="green", lty="dashed")
```

### irregular spaced data

```r
x <- sort(runif(10, max=10))
```
y <- c(rnorm(5), c(1,1,1,1,3))
xnew <- seq(-1, 11, 0.1)
plot(x, y, ylim=c(-3, 3), xlim=range(xnew))
lines(spline(x, y, xmin=min(xnew), xmax=max(xnew), n=200), col="blue")
lines(aspline(x, y, xnew, col="red")
lines(aspline(x, y, xnew, method="improved"), col="black", lty="dotted")
lines(aspline(x, y, xnew, method="improved", degree=10), col="green", lty="dashed")

## an example of Akima, 1991
x <- c(-3, -2, -1, 0, 1, 2, 2.5, 3)
y <- c(0, 0, 0, 0, -1, -1, 0, 2)
plot(x, y, ylim=c(-3, 3))
lines(spline(x, y, n=200), col="blue")
lines(aspline(x, y, n=200), col="red")
lines(aspline(x, y, n=200, method="improved"), col="black", lty="dotted")
lines(aspline(x, y, n=200, method="improved", degree=10), col="green", lty="dashed")

---

**interp**

**Gridded Bivariate Interpolation for Irregular Data**

**Description**

These functions implement bivariate interpolation onto a grid for irregularly spaced input data. Bilinear or bicubic spline interpolation is applied using different versions of algorithms from Akima.

**Usage**

```r
interp(x, y, z, xo=seq(min(x), max(x), length = 40),
      yo=seq(min(y), max(y), length = 40),
      linear = TRUE, extrap=FALSE, duplicate = "error", dupfun = NULL, ncp = NULL)
interp.old(x, y, z, xo=seq(min(x), max(x), length = 40),
           yo=seq(min(y), max(y), length = 40),
           ncp = 0, extrap=FALSE, duplicate = "error", dupfun = NULL)
interp.new(x, y, z, xo=seq(min(x), max(x), length = 40),
           yo=seq(min(y), max(y), length = 40),
           ncp = NULL, extrap=FALSE, duplicate = "error", dupfun = NULL)
```

**Arguments**

- **x**: vector of x-coordinates of data points. Missing values are not accepted.
- **y**: vector of y-coordinates of data points. Missing values are not accepted.
- **z**: vector of z-coordinates of data points. Missing values are not accepted.
- **xo** and **yo** must be the same length and may contain no fewer than four points. The points of **x** and **y** cannot be collinear, i.e., they cannot fall on the same line (two vectors **x** and **y** such that **y** = **a**x + **b** for some **a**, **b** will not be accepted). **interp** is meant for cases in which you have **x**, **y** values scattered over a plane and a **z** value for each. If, instead, you are trying to evaluate a mathematical function, or get a graphical interpretation of relationships that can be described by a polynomial, try **outer**().
**interp**

- **xo**: vector of x-coordinates of output grid. The default is 40 points evenly spaced over the range of x. If extrapolation is not being used (extrap=FALSE, the default), xo should have a range that is close to or inside of the range of x for the results to be meaningful.

- **yo**: vector of y-coordinates of output grid; analogous to xo, see above.

- **linear**: logical – indicating whether linear or spline interpolation should be used. supersedes old ncp parameter

- **ncp**: deprecated, use parameter linear. Now only used by interp.old(). meaning was: number of additional points to be used in computing partial derivatives at each data point. ncp must be either 0 (partial derivatives are not used), or at least 2 but smaller than the number of data points (and smaller than 25).

- **extrap**: logical flag: should extrapolation be used outside of the convex hull determined by the data points?

- **duplicate**: character string indicating how to handle duplicate data points. Possible values are
  - "error" produces an error message,
  - "strip" remove duplicate z values,
  - "mean", "median", "user" calculate mean, median or user defined function (dupfun) of duplicate z values.

- **dupfun**: a function, applied to duplicate points if duplicate= "user".

**Details**

If linear is TRUE (default), linear interpolation is used in the triangles bounded by data points. Cubic interpolation is done if linear is set to FALSE. If extrap is FALSE, z-values for points outside the convex hull are returned as NA. No extrapolation can be performed for the linear case.

The interp function handles duplicate (x,y) points in different ways. As default it will stop with an error message. But it can give duplicate points an unique z value according to the parameter duplicate (mean,median or any other user defined function).

The triangulation scheme used by interp works well if x and y have similar scales but will appear stretched if they have very different scales. The spreads of x and y must be within four orders of magnitude of each other for interp to work.

**Value**

- list with 3 components:
  - x,y: vectors of x- and y- coordinates of output grid, the same as the input argument xo, or yo, if present. Otherwise, their default, a vector 40 points evenly spaced over the range of the input x.
  - z: matrix of fitted z-values. The value z[i,j] is computed at the x,y point xo[i], yo[j]. z has dimensions length(xo) times length(yo).

**Note**

interp is a wrapper for the two versions interp.old (it uses (almost) the same Fortran code from Akima 1978 as the S-Plus version) and interp.new (it is based on new Fortran code from Akima 1996). For linear interpolation the old version is chosen, but spline interpolation is done by the new version.
Earlier versions (pre 0.5-1) of `interp` used the parameter `ncp` to choose between linear and cubic interpolation, this is now done by setting the logical parameter `linear`. Use of `ncp` is still possible, but is deprecated.

The resulting structure is suitable for input to the functions `contour` and `image`. Check the requirements of these functions when choosing values for `xo` and `yo`.

**References**


**See Also**

`contour`, `image`, `approx`, `spline`, `aspline`, `outer`, `expand.grid`.

**Examples**

```r
data(akima)
plot(y ~ x, data = akima, main = "akima example data")
with(akima, text(x, y, formatC(z,dig=2), adj = -0.1))

## linear interpolation
akima.li <- interp(akima$x, akima$y, akima$z)
image (akima.li, add=TRUE)
contour(akima.li, add=TRUE)
points (akima, pch = 3)

## increase smoothness (using finer grid):
akima.smooth <- with(akima, interp(x, y, z, xo=seq(0,25, length=100),
                                  yo=seq(0,20, length=100)))
image (akima.smooth, main = "interp(<akima data>, *) on finer grid")
contour(akima.smooth, add = TRUE, col = "thistle")
points(akima, pch = 3, cex = 2, col = "blue")
# use triangulation package to show underlying triangulation:
if(library(tripack, logical.return=TRUE))
  plot(tri.mesh(akima), add=TRUE, lty="dashed")

## use only 15 points (interpolation only within convex hull!)
akima.part <- with(akima, interp(x[1:15], y[1:15], z[1:15]))
image(akima.part)
title("interp() on subset of only 15 points")
contour(akima.part, add=TRUE)
points(akima$x[1:15],akima$y[1:15], col = "blue")

## spline interpolation, two variants (AMS 526 "Old", AMS 761 "New")
## -----------------------------------------------------------------
## "Old": use 5 points to calculate derivatives -> many NAs
akima.so <- interp.old(akima$x, akima$y, akima$z,
                       xo=seq(0,25, length=100), yo=seq(0,20, length=100), ncp=5)
table(is.na(akima.so)) ## 3990 NA's; ~ 40 %
akima.so <- with(akima,
                 interp.old(x,y,z, xo=seq(0,25, length=100), yo=seq(0,20, len=100), ncp = 4))
table(is.na(akima.so)) ## still 3429
```
interpp

Pointwise Bivariate Interpolation for Irregular Data

Description

If `ncp` is zero, linear interpolation is used in the triangles bounded by data points. Cubic interpolation is done if partial derivatives are used. If `extrap` is `FALSE`, z-values for points outside the convex hull are returned as `NA`. No extrapolation can be performed if `ncp` is zero.

The `interpp` function handles duplicate `(x,y)` points in different ways. As default it will stop with an error message. But it can give duplicate points an unique z value according to the parameter `duplicate` (mean, median or any other user defined function).

The triangulation scheme used by `interp` works well if x and y have similar scales but will appear stretched if they have very different scales. The spreads of x and y must be within four orders of magnitude of each other for `interpp` to work.
Usage

interpp(x, y, z, xo, yo, linear=TRUE, extrap=FALSE, duplicate = "error", dupfun = NULL, ncp)

Arguments

x
vector of x-coordinates of data points. Missing values are not accepted.

y
vector of y-coordinates of data points. Missing values are not accepted.

z
vector of z-coordinates of data points. Missing values are not accepted.

x, y, and z must be the same length and may contain no fewer than four points. The points of x and y cannot be collinear, i.e., they cannot fall on the same line (two vectors x and y such that y = ax + b for some a, b will not be accepted).

xo
vector of x-coordinates of points at which to evaluate the interpolating function.

yo
vector of y-coordinates of points at which to evaluate the interpolating function.

linear
logical – indicating wether linear or spline interpolation should be used. supercedes old ncp parameter

ncp
deprecated, use parameter linear. Now only used by interpp.old().

meaning was: number of additional points to be used in computing partial derivatives at each data point. ncp must be either 0 (partial derivatives are not used, = linear interpolation), or at least 2 but smaller than the number of data points (and smaller than 25).

extrap
logical flag: should extrapolation be used outside of the convex hull determined by the data points?

duplicate
indicates how to handle duplicate data points. Possible values are "error" - produces an error message, "strip" - remove duplicate z values, "mean", "median", "user" - calculate mean, median or user defined function of duplicate z values.

dupfun
this function is applied to duplicate points if duplicate="user"

Value

list with 3 components:

x
vector of x-coordinates of output points, the same as the input argument xo.

y
vector of y-coordinates of output points, the same as the input argument yo.

z
fitted z-values. The value z[i] is computed at the x,y point x[i], y[i].

NOTE

Use interp if interpolation on a regular grid is wanted.

The two versions interpp.old and interpp.new refer to Akimas Fortran code from 1978 and 1996 resp. The call wrapper interpp chooses interpp.old for linear and interpp.new for cubic spline interpolation.

Earlier versions (pre 0.5-1) of interpp used the parameter ncp to choose between linear and cubic interpolation, this is now done by setting the logical parameter linear. Use of ncp is still possible, but is deprecated.
References


See Also

contour, image, approx, spline, outer, expand.grid, interp, aspline.

Examples

data(akima)
# linear interpolation at points (1,2), (5,6) and (10,12)
akima.lip<-interpp(akima$x, akima$y, akima$z,c(1,5,10),c(2,6,12))
akima.lip$z
# spline interpolation
akima.sip<-interpp(akima$x, akima$y, akima$z,c(1,5,10),c(2,6,12),
  linear=FALSE)
akima.sip$z
Index

* Topic **arith**
  aspline, 2

* Topic **datasets**
  akima, 1

* Topic **dplot**
  aspline, 2
  interp, 4
  interpp, 7

akima, 1
approx, 3, 6, 9
aspline, 2, 6, 9

contour, 6, 9
expand.grid, 6, 9
image, 6, 9
interp, 4, 9
interpp, 7

outer, 6, 9
spline, 3, 6, 9
xy.coords, 3