

Package ‘interpolation’

July 22, 2025

Type Package

Title Interpolation of Bivariate Functions

Version 0.1.1

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Provides two different methods, linear and nonlinear, to interpolate a bivariate function, scalar-valued or vector-valued. The interpolated data are not necessarily gridded. The algorithms are performed by the 'C++' library 'CGAL' (<<https://www.cgal.org/>>).

License GPL-3

URL <https://github.com/stla/interpolation>

BugReports <https://github.com/stla/interpolation/issues>

Imports Rcpp (>= 1.0.10)

LinkingTo Rcpp, RcppCGAL, BH

Encoding UTF-8

RoxxygenNote 7.2.3

SystemRequirements C++ 17, gmp

NeedsCompilation yes

Author Stéphane Laurent [aut, cre]

Repository CRAN

Date/Publication 2023-12-20 09:20:02 UTC

Contents

interpfun	2
Index	4

interpfun	<i>Interpolation function</i>
-----------	-------------------------------

Description

Generates a function $f(x,y)$ that interpolates the known function values at some given (x,y) -coordinates.

Usage

```
interpfun(x, y, z, method = "linear")
```

Arguments

<code>x, y</code>	two numeric vectors of the same size
<code>z</code>	a numeric vector or matrix of the same size as <code>x</code> and <code>y</code> , with two or three columns if it is a matrix
<code>method</code>	method of interpolation, either "linear" or "sibson"; the "sibson" method is not available for vector-valued functions, i.e. if <code>z</code> is a matrix

Details

The new pairs of coordinates must be in the convex hull of the points (x,y) . If a new pair is outside the convex hull, the interpolating function returns NA for this pair. The linear method is exact for a function of the form $f(x,y) = a + bx*x + by*y$. The Sibson method is exact for a function of the form $f(x,y) = a + bx*x + by*y + c*(x^2 + y^2)$. This method estimates the gradient of the function and this can fail if the data are insufficient, in which case NA is returned.

Value

A function whose graph interpolates the data $((x,y),z)$.

Examples

```
library(interpolation)
a <- 0.2; bx <- 0.3; by <- -0.4
x0 <- y0 <- seq(1, 10, by = 1)
Grid <- expand.grid(X = x0, Y = y0)
x <- Grid$X; y <- Grid$Y
z <- a + bx*x + by*y
xnew <- ynew <- seq(2.5, 8.5, by = 1)
fun <- interpfun(x, y, z, "linear")
# computed values:
( znew <- fun(xnew, ynew) )
# true values:
a + bx*xnew + by*ynew

# a vector-valued example ####
```

```

x <- y <- c(-5, -4, -3, -2, 2, 3, 4, 5)
From <- as.matrix(expand.grid(x0 = x, y0 = y))
f <- function(x0y0) {
  d <- c(-10, -5) - x0y0
  x0y0 + 0.8 * d / sqrt(c(crossprod(d)))
}
To <- t(apply(From, 1L, f))
x0 <- From[, "x0"]; y0 <- From[, "y0"]
x1 <- To[, 1L]; y1 <- To[, 2L]
# plot data
plot(
  x0, y0, asp = 1, pch = 19, xlab = "x", ylab = "y"
)
arrows(x0, y0, x1, y1, length = 0.1)
# interpolate
library(interpolation)
fun <- interpfun(x0, y0, To, method = "linear")
From_new <- rbind(
  as.matrix(expand.grid(x0 = c(-1, 0, 1), y0 = (-5):5)),
  as.matrix(expand.grid(x0 = c(-5, -4, -3, -2), y0 = c(-1, 0, 1))),
  as.matrix(expand.grid(x0 = c(2, 3, 4, 5), y0 = c(-1, 0, 1)))
)
To_new <- fun(From_new)
x0 <- From_new[, "x0"]; y0 <- From_new[, "y0"]
x1 <- To_new[, 1L]; y1 <- To_new[, 2L]
points(x0, y0, pch = 19, col = "red")
arrows(x0, y0, x1, y1, length = 0.1, col = "red")

```

Index

interpfun, [2](#)