

# Package ‘spsi’

August 19, 2015

**Title** Shape-Preserving Uni-Variate and Bi-Variate Spline Interpolation

**Date** 2015-08-12

**Version** 0.1

**Author** Szymon Sacher & Andrew Clausen, The University of Edinburgh. Excerpts adapted from Fortran code Copyright (C) Paolo Costantini

**Maintainer** Szymon Sacher <s1340144@sms.ed.ac.uk>

**Description** Program uses method of polynomial of variable degrees to interpolate gridded data preserving monotonicity and/or convexity or none. Method is implemented for univariate and bivariate cases. If values of derivatives are provided, spline will fix them, if not program will estimate them numerically. Package written purely in R.

**License** GPL (>= 2.0)

**LazyData** TRUE

**Depends** plot3D, R(>= 2.10.0)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-08-19 20:35:02

## R topics documented:

akima . . . . .	2
spsi . . . . .	2
sps_eval . . . . .	3
sps_fun . . . . .	5
sps_prep . . . . .	7

<b>Index</b>	<b>11</b>
--------------	-----------

---

akima                      *Wave distortions data.*

---

**Description**

Data taken from a study of waveform distortion in electronic circuits.

**Usage**

```
data("akima")
```

**Format**

The format is: num [1:11, 1:9] 58.2 37.2 22.4 21.8 16.8 12 7.4 3.2 0 0 ...

**Source**

Akima, H; 'A Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures', Comm. ACM, 17, 1974, pp 26-27

**Examples**

```
data(akima)
```

---

spsi                      *Shape Preserving Spline Interpolation*

---

**Description**

Shape preserving Uni-variate and Bi-variate spline interpolation using method of polynomials of variable degree.

**Details**

Package:    spsi  
Type:        Package  
Version:    0.1  
Date:        2015-08-19  
License:    GPL-2 or greater  
LazyLoad:   yes

**Author(s)**

Szymon Sacher <s1340144@sms.ed.ac.uk> & Andrew Clausen <andrew.clausen@ed.ac.uk>  
 Excerpts adapted from Fortran code Copyright (C) Paolo Costantini

**See Also**

[sps\\_fun](#) [sps\\_prep](#)

---

sps_eval	<i>Evaluates spline on given points</i>
----------	---

---

**Description**

Function which uses output from `sps_prep` to evaluate spline on given set of tabulation points

**Usage**

```
sps_eval(spline, x, der.x = NULL, y = NULL, der.y = NULL, grid = FALSE)
```

**Arguments**

<code>spline</code>	Output of <code>sps_prep</code> function; list containing values of data points, derivatives on the points, degrees of polynomial, etc. see: <a href="#">sps_prep</a>
<code>x</code>	Vector or matrix of x-coordinates of tabulation points.
<code>der.x</code>	vector of requested derivatives of spline with respect to x. By default will return values of spline. If <code>der.x = c(0,1)</code> will return value of spline and its first derivative with respect to x. Only 0 and 1 are supported so far.
<code>y</code>	Vector or matrix of y-coordinates of tabulation points. For bi-variate case only.
<code>der.y</code>	For bi-variate case only. Vector of requested derivatives of spline with respect to y. By default will return values of spline. If <code>der.y = c(0,1)</code> will return value of spline and its first derivative with respect to y. Only 0 and 1 are supported so far. See details.
<code>grid</code>	For bi-variate case only. If TRUE function will return matrix of values of the spline on grid spanned by vectors of tabulation points. If FALSE vector of $f(x[i], y[i]), i = 1, 2, \dots, length(x)$ will be returned. If matrices were given as tabulation points <code>grid</code> is meaningless.

**Details**

`Der.x` and `der.y` need some more attention for bi-variate case. If they are not provided they are both assumed to be 0. If user needs more than 1 derivative then length of `der.x` and `der.y` must be equal. For example if `der.x = c(0,1,0)` and `der.y = c(0,0,1)` than function will return values of spline, values of partial derivative with respect to x and partial derivative with respect to y, respectively, on each tabulation point in a form of list of vectors or matrices.

**Value**

If length of `der.x = 1` function will return vector or matrix of values, depending on parameter `grid`. Otherwise if length of `der.x` is greater than 1, function will return list of vectors or matrices depending on `grid`.

**Author(s)**

Szymon Sacher <s1340144@sms.ed.ac.uk> & Andrew Clausen <andrew.clausen@ed.ac.uk>  
Excerpts adapted from Fortran code Copyright (C) Paolo Costantini

**References**

Costantini, P; Fontanella, F; 'Shape Preserving Bi-variate Interpolation' sSIAM J NUMER. ANAL. Vol. 27, No.2, pp. 488-506, April 1990

**See Also**

[sps\\_fun](#) [sps\\_prep](#)

**Examples**

```
## Univariate example
x <- c( 1, 2, 3, 4, 5, 6)
y <- c(16 ,18, 21, 17, 15, 12)
spline <- sps_prep(x, y, shape = 'monotonicity', smoothness = 2)
plot(seq(1, 6, 0.1), sps_eval(spline, seq(1, 6, 0.1)))

## Bivariate example

fun <- function(x,y) pmax(0, sin(pi*x) * sin(pi*y))

X <- seq(-1, 2, 0.5)
Y <- seq(-1, 1, 0.5)
grid <- mesh(X, Y)

Z <- matrix(fun(grid$x, grid$y), ncol = length(Y))

X_ <- seq(-1, 2, 0.05)
Y_ <- seq(-1, 1, 0.05)

# Prepare spline parameters
spline <- sps_prep(X, Y, Z)

# evaluate spline on grid of tabulation points spanned by X_ and Y_
eval <- sps_eval(spline, x = X_, y = Y_, grid = TRUE)

# Plot resulting data
persp3D(X_, Y_, eval)
```

---

sps\_fun

*Shape Preserving Bi-variate and Uni-variate Interpolation.*


---

### Description

Both functions recognize whether uni-variate or bi-variate interpolation is requested

sps\_fun calls sps\_prep to prepare spline object and according to data provided creates uni-variate or bi-variate executable function. Parameters can be set by specifying them as arguments

sps\_interpolate prepares spline and returns its values on given set of tabulation points

### Usage

```
sps_fun(x, y, z=NULL, der=0, der.x=der, der.y=der, grid = FALSE, ...)
```

```
sps_interpolate(x, y, z=NULL, xt, yt=NULL, grid = TRUE, ...)
```

### Arguments

x	Vector of x-coordinates of data.
y	Vector of y-coordinates of data. For uni-variate case length should be equal to length of x and $y[i] = f(x[i])$ . For bi-variate case length may differ.
z	Bi-variate case only. Matrix of values of function on grid spanned by x and y; $z[i, j] = f(x[i], y[j])$ OR Vector of values of functions on points (x, y); $z[i] = f(x[i], y[i])$ . In this case length of all three vectors should be equal and it should be possible to transform the point to a gridded form.
xt, yt	Vectors OR Matrices containing coordinates of the tabulation point; yt for bi-variate case only
der, der.x, der.y	Indicate which derivative should be returned; der.y used in bi-variate case only. If length of vector is greater than 1 function will return list of values. For bi-variate case vectors should have equal length. At the moment only 1 and 0 are supported. See examples.
grid	For bi-variate interpolation only. If TRUE function will return matrix of values of the spline on grid spanned by vectors of tabulation points. If FALSE vector of $f(xt[i], yt[i])$ , $i = 1, 2, \dots, \text{length}(xt)$ will be returned. If matrices were given as tabulation points grid is meaningless.
...	arguments in <i>tag = value</i> format. The tags must come from the names of parameters of <i>sps_prep</i> and/or <i>sps_eval</i> .

## Details

Following parameters can be specified:

**fx, fy, fxy** Matrices with values of the derivatives: see [sps\\_prep](#)

**maxdeg** Maximum degree of polynomial allowed: see [sps\\_prep](#)

**smoothness** Smoothness required: see [sps\\_prep](#)

**tol** Relative tolerance used by program

**shape** Vector of shape attributes that must be preserved. Must contain only 'monotonicity' and/or 'curvature'

## Value

**sps\_fun** Function of 1 or 2 variables depending on data provided. Function will accept vector(s) or matrix(es) of tabulation point and return object of the same class. If while calling `sps_fun` length `der.x` (and possibly `der.y`) was bigger than 1. Resulting function will be returning list of values of respective derivatives at given tabulation points.

**sps\_interpolate** Vector, matrix or list of vectors/matrices of value of function and/or derivatives on given set of tabulation points

## Author(s)

Szymon Sacher <[s1340144@sms.ed.ac.uk](mailto:s1340144@sms.ed.ac.uk)> & Andrew Clausen <[andrew.clausen@ed.ac.uk](mailto:andrew.clausen@ed.ac.uk)>  
Excerpts adapted from Fortran code Copyright (C) Paolo Costantini

## References

Costantini, P; Fontanella, F; 'Shape Preserving Bi-variate Interpolation' SIAM J NUMER. ANAL. Vol. 27, No.2, pp. 488-506, April 1990

## See Also

[sps\\_eval](#) [sps\\_prep](#)

## Examples

```
## Example 1

# Following example shows usage of sps_fun along with the parameter 'smoothness'.
# As you will see if smoothness = 2 then first derivative of function is differentiable
# everywhere.

x <- c( 1, 2, 3, 4, 5, 6)
y <- c(16 ,18, 21, 17, 15, 12)

evalK1 <- sps_fun(x, y)
derK1 <- sps_fun(x, y, der.x=1)

evalK2 <- sps_fun(x, y, smoothness = 2)
```

```

derK2 <- sps_fun(x, y, smoothness = 2, der.x = 1)

xs <- seq(1, 6, 0.01)
par(mfrow = c(2, 2))
plot(x, y, col = "red", xlim = c(0, 7), ylim = c(10, 22),
     main = "Spline, smoothness = 1")
grid()

lines(xs,evalK1(xs), col="cyan")
par(new = TRUE)
plot(derK1, from = 1,to = 6, col = "magenta", xlim = c(0,7), ylim = c(-6,5),
     xaxt = 'n',yaxt='n',ann = FALSE)
axis(4, -6:5)

plot(x, y, col="red", xlim=c(0,7), ylim=c(10,22),
     main = "Spline, smoothness = 2")
grid()

lines(xs,evalK2(xs), col="cyan")
par(new = TRUE)
plot(derK2, from = 1, to = 6, col = "magenta", xlim = c(0,7), ylim = c(-6,5),
     xaxt = 'n',yaxt = 'n', ann = FALSE)
axis(4, -6:5)

plot(derK1, from = 1.5, to = 2.5)
plot(derK2, from = 1.5, to = 2.5)

## EXAMPLE 2
par(mfrow = c(1,1))
X <- seq(0, 50, 5)
Y <- seq(0, 40, 5)

X_ <- seq(0, 50, 0.5)
Y_ <- seq(0, 40, 0.5)

persp3D(X_, Y_, sps_interpolate(X, Y, akima, X_, Y_,
                                grid = TRUE, shape = 'monotonicity'))

```

---

sps\_prep

*Prepare list of objects for shape preserving interpolation.*


---

## Description

This is primitive function which output is used by `sps_eval`. Function works with both uni- and bi-variate functions and returns list consisting original data provided, values of the derivatives and degrees of polynomial needed to satisfy the shape constraints required

**Usage**

```
sps_prep(x, y, z=NULL,
         fx = NA, fy = NA, fxy = NA,
         shape = c("monotonicity", "curvature"),
         shape.x = shape, shape.y = shape,
         max.deg = 50, smoothness = 1,
         tol = 0.0001)
```

**Arguments**

x	Vector of x-coordinates of points to be interpolated.
y	Vector of y-coordinates of points to be interpolated. For uni-variate case length should be equal to length of x and $y[i] = f(x[i])$ . For bi-variate case length may differ.
z	Bi-variate case only. Matrix of values of function on grid spanned by x and y; $z[i, j] = f(x[i], y[j])$ OR Vector of values of functions on points (x, y); $z[i] = f(x[i], y[i])$ . In this case length of all three vectors should be equal and it should be possible to transform the point to a gridded form.
fx	Matrix of values of the derivative with respect to x of the function; $fx[i, j] = fx(x[i], y[j])$ ; By default it is estimated internally.
fy	Bi-variate case only. Matrix of values of the derivative with respect to y of the function; $fy[i, j] = fy(x[i], y[j])$ ; By default it is estimated internally.
fxy	Bi-variate case only. Matrix of values of mixed partial derivative of the function; $fxy[i, j] = fxy(x[i], y[j])$ ; By default it is estimated internally.
shape	Specifies which attributes should be preserved. Vector should contain 'monotonicity' and/or 'curvature' only. In bi-variate case this can be set separately for both dimensions using shape.x, shape.y.
shape.x	Specifies which attributes should be preserved for x dimension.
shape.y	Specifies which attributes should be preserved for y dimension.
max.deg	Specifies maximum degree of polynomial allowed. In some cases in order to preserve shape, very high degrees are necessary. If maximum degree is reached, it is not guaranteed that resulting spline will preserve all the attributes required.
smoothness	How many times does the spline needs to be differentiable.
tol	Tolerance used within program. Default value is suitable for graphical purposes.

**Details**

If z is not provided function will prepare list needed for uni-variate interpolation. If values of the derivatives are provided, resulting spline will preserve them. For bi-variate case it is possible to set some derivatives and let program estimate the rest.

**Value**

Uni-variate: list with 6 components:

x, y                      Coordinates of the data points



k                   Smoothness (or continuity class) required  
 fx                   Estimated or given values of derivative  
 deg                  Degree of polynomial needed on each of the line segments  
 dim                  Number of variables; numeric equal to 1

Bi-variate: list with 10 components:

x,y                  Coordinates of the data points  
 z                    Matrix of vales of the function  
 k                    Smoothness (or continuity class) required  
 fx, fy, fxy         Estimated or given values of respective derivatives  
 deg.x, deg.y        Degree of polynomial needed on each of the line segments in each dimension  
 dim                  Number of variables; numeric equal to 2

### Author(s)

Szymon Sacher <s1340144@sms.ed.ac.uk> & Andrew Clausen <andrew.clausen@ed.ac.uk>  
 Excerpts adapted from Fortran code Copyright (C) Paolo Costantini

### References

Costantini, P; Fontanella, F; 'Shape Preserving Bi-variate Interpolation' SIAM J NUMER. ANAL.  
 Vol. 27, No.2, pp. 488-506, April 1990

### See Also

[sps\\_eval](#) [sps\\_fun](#)

### Examples

```

## Univariate example
x <- c(1, 2, 3, 4, 5, 6)
y <- c(16, 18, 21, 17, 15, 12)
spline <- sps_prep(x, y, shape = 'monotonicity', smoothness = 2)
plot(seq(1, 6, 0.1), sps_eval(spline, seq(1, 6, 0.1)))

## Bivariate example

tower <- function(x, y)
{
  X <- abs(x)
  Y <- abs(y)
  ifelse((X + Y) <= 1, floor(3*(1 - X - Y)),
        ifelse(pmax(X, Y) >= 1, pmax(X, Y)/2 - 0.5,
              0))
}

X <- Y <- seq(-1.25, 1.25, 2.5/13)

```

```
grid <- mesh(X, Y)
Z <- tower(grid$x, grid$y)

spline <- sps_prep(X, Y, Z)

X_ <- Y_ <- seq(-1.25, 1.25, 2.5/60)

persp3D(X_, Y_, sps_eval(spline, x = X_, y = Y_, grid = TRUE))
```

# Index

\*Topic **Akima**

akima, 2

\*Topic **Costantini**

sps\_eval, 3

sps\_fun, 5

sps\_prep, 7

\*Topic **curvature**

sps\_fun, 5

sps\_prep, 7

\*Topic **datasets**

akima, 2

\*Topic **interpolation**

sps\_eval, 3

spsi, 2

\*Topic **monotonicity**

sps\_fun, 5

sps\_prep, 7

\*Topic **spline**

sps\_eval, 3

sps\_fun, 5

sps\_prep, 7

spsi, 2

akima, 2

sps\_eval, 3, 6, 9

sps\_fun, 3, 4, 5, 9

sps\_interpolate (sps\_fun), 5

sps\_prep, 3, 4, 6, 7

spsi, 2