Package ‘schumaker’

January 23, 2026

Type Package

Title Schumaker Shape-Preserving Spline

Version 1.2.2

Date 2026-01-23

Encoding UTF-8

Description This is a shape preserving spline <doi:10.1137/0720057>
which is guaranteed to be monotonic and concave or convex if the
data is monotonic and concave or convex. It does not use any
optimisation and is therefore quick and smoothly converges to a
fixed point in economic dynamics problems including value function
iteration. It also automatically gives the first two derivatives
of the spline and options for determining behaviour when evaluated
outside the interpolation domain.

License MIT + file LICENSE

Suggests testthat, knitr, numDeriv

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Stuart Baumann [aut, cre],
Margaryta Klymak [aut]

Maintainer Stuart Baumann <stuart@stuartbaumann.com>

Repository CRAN

Date/Publication 2026-01-23 09:30:02 UTC

Contents

`impute_gradients` *impute gradients from two vectors with x and y coordinates.*

Description

impute gradients from two vectors with x and y coordinates.

Usage

```
impute_gradients(x, y, edgeGradients = c(NA, NA))
```

Arguments

`x`	A vector of x coordinates
`y`	A corresponding vector of y coordinates
`edgeGradients`	This gives the options of specifing the gradients at either edge of the domain. By default this is c(NA,NA) meaning that the defaults from the original paper are used. If this is set to c(0,NA) for instance this will mean that the left edge gradient is zero and the right edge gradient is as recommended in the original paper. This setting has no impact if a full set of gradients is input.

References

Schumaker, L.L. 1983. On shape-preserving quadratic spline interpolation. SIAM Journal of Numerical Analysis 20: 854-64.

Judd (1998). Numerical Methods in Economics. MIT Press

Examples

```
x = seq(1,6)
y = log(x)
grads = impute_gradients(x, y)
```

`make_approx_functions_from_dataframe`
make_approx_functions_from_dataframe

Description

make_approx_functions_from_dataframe

Usage

```
make_approx_functions_from_dataframe(
  dataframe,
  group_vars,
  x_var,
  y_var,
  approx_func
)
```

Arguments

`dataframe`	A data.frame with your data.
`group_vars`	The variable names in the dataframe that subset the data into the various groups.
`x_var`	The name of the x variable in the dataframe
`y_var`	The name of the y variable in the dataframe.
`approx_func`	A function that takes in two arguments, an x vector and a y vector. Make sure it can handle vectors of length 0 (if that can happen in your data).

Value

A function of the form function(groupvar1, groupvar2, ..., x).

Examples

```
# Generating example data.
# Consider we have equity prices for several days and times.
RICs = c("BARC.L", "VOD.L", "IBM.L")
Dates = as.Date(c("11-11-2019", "12-11-2019", "13-11-2019",
           "14-11-2019", "15-11-2019"), format="%d-%m-%Y")
times = seq(0,28800, length.out = 10) # The number of seconds into the trading day.
dd = expand.grid(TIME = times, Date = Dates, RIC = RICs)
dd = merge(dd, data.frame(RIC = RICs, PRICE = c(160.00, 162.24, 137.24)))
randomness = rlnorm(dim(dd)[1])
dd$PRICE = dd$PRICE * cumprod(randomness)

approx_func = function(x,y){approxfun(x, y)}
dispatched_approxfun = make_approx_functions_from_dataframe(dd,
                                group_vars = c("RIC", "Date"),
                                 x_var = "TIME", y_var = "PRICE",
                                                     approx_func)
dispatched_approxfun("BARC.L", Dates[2], c(100, 156, 6045))
```

ppmak *ppmak*

Description

Create a spline with given intervals and quadratic coefficients. This is an internal function that is called from the Schumaker function. It roughly works like ppmak in matlab.

Usage

```
ppmak(IntStarts, SpCoefs, Vectorised = TRUE)
```

Arguments

`IntStarts`	This is a vector with the start of each interval.
`SpCoefs`	This is a matrix with three columns. The first is the coefficient of the squared term followed by linear term coefficients and constants.
`Vectorised`	This is a boolean parameter. Set to TRUE if you want to be able to input vectors to the created spline. If you will only input single values set this to FALSE as it is a bit faster.

Value

A spline function for the given intervals and quadratic curves. Each function takes an x value (or vector if Vectorised = TRUE) and outputs the interpolated y value (or relevent derivative).

ppmak2Deriv *ppmak2Deriv*

Description

Create the second derivative of the spline defined by given intervals and quadratic coefficients. This is an internal function that is called from the Schumaker function.

Usage

```
ppmak2Deriv(IntStarts, SpCoefs, Vectorised = TRUE)
```

Arguments

`IntStarts`	This is a vector with the start of each interval.
`SpCoefs`	This is a matrix with three columns. The first is the coefficient of the squared term followed by linear term coefficients and constants.
`Vectorised`	This is a boolean parameter. Set to TRUE if you want to be able to input vectors to the created spline. If you will only input single values set this to FALSE as it is a bit faster.

Value

A spline function for the given intervals and quadratic curves. Each function takes an x value (or vector if Vectorised = TRUE) and outputs the interpolated y value (or relevent derivative).

`ppmakDeriv` *ppmakDeriv*

Description

Create the derivative of the spline defined by given intervals and quadratic coefficients. This is an internal function that is called from the Schumaker function.

Usage

```
ppmakDeriv(IntStarts, SpCoefs, Vectorised = TRUE)
```

Arguments

`IntStarts`	This is a vector with the start of each interval.
`SpCoefs`	This is a matrix with three columns. The first is the coefficient of the squared term followed by linear term coefficients and constants.
`Vectorised`	This is a boolean parameter. Set to TRUE if you want to be able to input vectors to the created spline. If you will only input single values set this to FALSE as it is a bit faster.

Value

A spline function for the given intervals and quadratic curves. Each function takes an x value (or vector if Vectorised = TRUE) and outputs the interpolated y value (or relevent derivative).

`Schumaker` *Create a Schumaker spline*

Description

Create a Schumaker spline

Usage

```
Schumaker(
  x,
  y,
  gradients = NA,
  Vectorised = TRUE,
  Extrapolation = c("Curve", "Constant", "Linear"),
  edgeGradients = c(NA, NA)
)
```

Arguments

`x`	A vector of x coordinates
`y`	A corresponding vector of y coordinates
`gradients`	(Optional) A corresponding vector of gradiants at the data points. If this is NA then it will be estimated.
`Vectorised`	This is a boolean parameter. Set to TRUE if you want to be able to input vectors to the created spline. If you will only input single values set this to FALSE as it is a bit faster.
`Extrapolation`	This determines how the spline function responds when an input is recieved outside the domain of x. The options are "Curve" which outputs the result of the point on the quadratic curve at the nearest interval, "Constant" which outputs the y value at the end of the x domain and "Linear" which extends the spline using the gradiant at the edge of x.
`edgeGradients`	This gives the options of specifing the gradients at either edge of the domain. By default this is c(NA,NA) meaning that the defaults from the original paper are used. If this is set to c(0,NA) for instance this will mean that the left edge gradient is zero and the right edge gradient is as recommended in the original paper. This setting has no impact if a full set of gradients is input.

Value

A list with 3 spline functions and a table with spline intervals and coefficients. The first spline is the schumaker spline, the second spline is the first derivative of the schumaker spline, the third spline is the second derivative of the schumaker spline. Each function takes an x value (or vector if Vectorised = TRUE) and outputs the interpolated y value (or relevant derivative).

References

Schumaker, L.L. 1983. On shape-preserving quadratic spline interpolation. SIAM Journal of Numerical Analysis 20: 854-64.

Judd (1998). Numerical Methods in Economics. MIT Press

Examples

```
x = seq(1,6)
y = log(x)
SSS = schumaker::Schumaker(x,y, Vectorised = TRUE)
xarray = seq(1,6,0.01)
Result = SSS$Spline(xarray)
Result2 = SSS$DerivativeSpline(xarray)
Result3 = SSS$SecondDerivativeSpline(xarray)
plot(xarray, Result, ylim=c(-0.5,2))
lines(xarray, Result2, col = 2)
lines(xarray, Result3, col = 3)
```

`SchumakerIndInterval` *SchumakerIndInterval*

Description

This creates quadratic coefficients for one interval of a domain. This is an internal function that is called from the Schumaker function.

Usage

```
SchumakerIndInterval(z, s, Smallt)
```

Arguments

`z`	This is the y value at edges of an interval.
`s`	This is the slope at edges of an interval.
`Smallt`	This is x values at the edge of an interval.

Value

The location of the knot and quadratic coefficients for an interval.

Index