Package: splines2 (via r-universe)

October 28, 2026

Title Regression Spline Functions and Classes

Version 0.5.4

Description Constructs basis functions of B-splines, M-splines, I-splines, convex splines (C-splines), periodic splines, natural cubic splines, generalized Bernstein polynomials, their derivatives, and integrals (except C-splines) by closed-form recursive formulas. It also contains a C++ head-only library integrated with Rcpp. See Wang and Yan (2021) <doi:10.6339/21-JDS1020> for details.

Imports stats, graphics, Rcpp

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, tinytest, RcppArmadillo

Depends R (>= 3.2.3)

VignetteBuilder knitr

License GPL (>= 3)

URL https://wwenjie.org/splines2, https://github.com/wenjie2wang/splines2

BugReports https://github.com/wenjie2wang/splines2/issues

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

Author Wenjie Wang [aut, cre] (<https://orcid.org/0000-0003-0363-3180>), Jun Yan [aut] (<https://orcid.org/0000-0003-4401-7296>)

Maintainer Wenjie Wang <wang@wwenjie.org>

Repository https://cran.r-universe.dev

Date/Publication 2025-02-28 05:40:02 UTC

RemoteUrl https://github.com/cran/splines2

RemoteRef HEAD

RemoteSha 2321fc1788c387ad06e531d408c31b5d840b2286

Contents

bernsteinPoly 2
bSpline 4
cSpline 7
deriv 10
iSpline 12
knots 14
mSpline 15
naturalSpline 19
plot.splines2 22
predict 23
update 25

Index **27**

`bernsteinPoly` *Generalized Bernstein Polynomial Basis Functions*

Description

Returns generalized Bernstein polynomial basis functions of the given degree over the specified range.

Usage

```
bernsteinPoly(
  x,
  degree = 3,
  intercept = FALSE,
  Boundary.knots = NULL,
  derivs = 0L,
  integral = FALSE,
  ...
)

bpoly(
  x,
  degree = 3,
  intercept = FALSE,
  Boundary.knots = NULL,
  derivs = 0L,
  integral = FALSE,
  ...
)
```

Arguments

x	The predictor variable taking values inside of the specified boundary. Missing values are allowed and will be returned as they are.
degree	A nonnegative integer representing the degree of the polynomials.
intercept	If TRUE, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
Boundary.knots	Boundary points at which to anchor the Bernstein polynomial basis. The default value is NULL and the boundary knots is set internally to be range(x, na.rm = TRUE).
derivs	A nonnegative integer specifying the order of derivatives. The default value is 0L for Bernstein polynomial basis functions.
integral	A logical value. If TRUE, the integrals of the Bernstein polynomials will be returned. The default value is FALSE.
...	Optional arguments that are not used.

Details

The Bernstein polynomial basis functions are defined over the support from 0 to 1. The generalized Bernstein polynomial basis functions extend the support to any finite interval in the real line.

The function bpoly() is an alias to encourage the use in a model formula.

Value

A BernsteinPoly object that is essentially a numeric matrix of dimension length(x) by degree + as.integer(intercept).

Examples

```
library(splines2)

x1 <- seq.int(0, 1, 0.01)
x2 <- seq.int(- 2, 2, 0.01)

## Bernstein polynomial basis matrix over [0, 1]
bMat1 <- bernsteinPoly(x1, degree = 4, intercept = TRUE)

## generalized Bernstein polynomials basis over [- 2, 2]
bMat2 <- bernsteinPoly(x2, degree = 4, intercept = TRUE)

op <- par(mfrow = c(1, 2))
plot(bMat1)
plot(bMat2)

## the first and second derivative matrix
d1Mat1 <- bernsteinPoly(x1, degree = 4, derivs = 1, intercept = TRUE)
d2Mat1 <- bernsteinPoly(x1, degree = 4, derivs = 2, intercept = TRUE)
d1Mat2 <- bernsteinPoly(x2, degree = 4, derivs = 1, intercept = TRUE)
d2Mat2 <- bernsteinPoly(x2, degree = 4, derivs = 2, intercept = TRUE)
```

```
par(mfrow = c(2, 2))
plot(d1Mat1)
plot(d1Mat2)
plot(d2Mat1)
plot(d2Mat2)

## reset to previous plotting settings
par(op)

## or use the deriv method
all.equal(d1Mat1, deriv(bMat1))
all.equal(d2Mat1, deriv(bMat1, 2))

## the integrals
iMat1 <- bernsteinPoly(x1, degree = 4, integral = TRUE, intercept = TRUE)
iMat2 <- bernsteinPoly(x2, degree = 4, integral = TRUE, intercept = TRUE)
all.equal(deriv(iMat1), bMat1, check.attributes = FALSE)
all.equal(deriv(iMat2), bMat2, check.attributes = FALSE)
```

---

bSpline *B-Spline Basis for Polynomial Splines*

Description

Generates the spline basis matrix for B-splines representing the family of piecewise polynomials with the specified interior knots, degree, and boundary knots, evaluated at the values of x.

Usage

```
bSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  Boundary.knots = NULL,
  periodic = FALSE,
  derivs = 0L,
  integral = FALSE,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)

ibs(
  x,
  df = NULL,
  knots = NULL,
```

```
  degree = 3,
  intercept = FALSE,
  Boundary.knots = NULL,
  ...
)

dbs(
  x,
  derivs = 1L,
  df = NULL,
  knots = NULL,
  degree = 3,
  intercept = FALSE,
  Boundary.knots = NULL,
  ...
)

bsp(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  Boundary.knots = NULL,
  periodic = FALSE,
  derivs = 0L,
  integral = FALSE,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)
```

Arguments

x	The predictor variable. Missing values are allowed and will be returned as they are.
df	Degree of freedom that equals to the column number of the returned matrix. One can specify df rather than knots, then the function chooses df - degree - as.integer(intercept) internal knots at suitable quantiles of x ignoring missing values and those x outside of the boundary. For periodic splines, df - as.integer(intercept) internal knots will be chosen at suitable quantiles of x relative to the beginning of the cyclic intervals they belong to (see Examples) and the number of internal knots must be greater or equal to the specified degree - 1. If internal knots are specified via knots, the specified df will be ignored.
knots	The internal breakpoints that define the splines. The default is NULL, which results in a basis for ordinary polynomial regression. Typical values are the mean or median for one knot, quantiles for more knots. For periodic splines, the number of knots must be greater or equal to the specified degree - 1. Duplicated internal knots are not allowed.

degree	A nonnegative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piecewise constant basis functions.
intercept	If TRUE, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
Boundary.knots	Boundary points at which to anchor the splines. By default, they are the range of x excluding NA. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots. For periodic splines, the specified bounary knots define the cyclic interval.
periodic	A logical value. If TRUE, the periodic splines will be returned. The default value is FALSE.
derivs	A nonnegative integer specifying the order of derivatives of splines basis function. The default value is 0.
integral	A logical value. If TRUE, the corresponding integrals of spline basis functions will be returned. The default value is FALSE. For periodic splines, the integral of each basis is integrated from the left boundary knot.
warn.outside	A logical value indicating if a warning should be thrown out when any x is outside the boundary. This option can also be set through options("splines2.warn.outside") after the package is loaded.
...	Optional arguments that are not used.

Details

This function extends the bs() function in the splines package for B-spline basis functions by allowing piecewise constant (left-closed and right-open except on the right boundary) spline basis of degree zero. In addition, the function provides derivatives or integrals of the B-spline basis functions when one specifies the arguments derivs or integral appropriately. The function constructs periodic B-splines when periodic is TRUE. All the implementations are based on the closed-form recursion formula following De Boor (1978) and Wang and Yan (2021).

The functions ibs() and dbs() are provided for convenience. The former provides the integrals of B-splines and is equivalent to bSpline() with integral = TRUE. The latter produces the derivatives of given order of B-splines and is equivalent to bSpline() with default derivs = 1. The function bsp() is an alias of to encourage the use in a model formula.

Value

A numeric matrix of length(x) rows and df columns if df is specified. If knots are specified instead, the output matrix will consist of length(knots) + degree + as.integer(intercept) columns if periodic = FALSE, or length(knots) + as.integer(intercept) columns if periodic = TRUE. Attributes that correspond to the arguments specified are returned for usage of other functions in this package.

References

De Boor, Carl. (1978). *A practical guide to splines*. Vol. 27. New York: Springer-Verlag.

Wang, W., & Yan, J. (2021). *Shape-restricted regression splines with R package splines2*. Journal of Data Science, 19(3),498–517.

See Also

knots for extracting internal and boundary knots.

Examples

```
library(splines2)

set.seed(1)
x <- runif(100)
knots <- c(0.3, 0.5, 0.6) # internal knots

## cubic B-splines
bsMat <- bSpline(x, knots = knots, degree = 3, intercept = TRUE)
ibsMat <- update(bsMat, integral = TRUE) # the integrals
d1Mat <- deriv(bsMat)                    # the 1st derivaitves
d2Mat <- deriv(bsMat, 2)                 # the 2nd derivaitves

op <- par(mfrow = c(2, 2), mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
plot(bsMat, ylab = "Cubic B-splines")
plot(ibsMat, ylab = "The integrals")
plot(d1Mat, ylab = "The 1st derivatives")
plot(d2Mat, ylab = "The 2nd derivatives")

## evaluate at new values
predict(bsMat, c(0.125, 0.801))

## periodic B-splines
px <- seq(0, 3, 0.01)
pbsMat <- bSpline(px, knots = knots, Boundary.knots = c(0, 1),
                  intercept = TRUE, periodic = TRUE)
ipMat <- update(pbsMat, integral = TRUE)
dpMat <- deriv(pbsMat)
dp2Mat <- deriv(pbsMat, 2)

plot(pbsMat, ylab = "Periodic B-splines", mark_knots = "b")
plot(ipMat, ylab = "The integrals", mark_knots = "b")
plot(dpMat, ylab = "The 1st derivatives", mark_knots = "b")
plot(dp2Mat, ylab = "The 2nd derivatives", mark_knots = "b")
par(op) # reset to previous plotting settings
```

---

cSpline *C-Spline Basis for Polynomial Splines*

Description

Generates the convex regression spline (called C-spline) basis matrix by integrating I-spline basis for a polynomial spline or the corresponding derivatives.

Usage

```
cSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = TRUE,
  Boundary.knots = NULL,
  derivs = 0L,
  scale = TRUE,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)

csp(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = TRUE,
  Boundary.knots = NULL,
  derivs = 0L,
  scale = TRUE,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)
```

Arguments

x	The predictor variable. Missing values are allowed and will be returned as they are.
df	Degree of freedom that equals to the column number of the returned matrix. One can specify df rather than knots, then the function chooses df - degree - as.integer(intercept) internal knots at suitable quantiles of x ignoring missing values and those x outside of the boundary. For periodic splines, df - as.integer(intercept) internal knots will be chosen at suitable quantiles of x relative to the beginning of the cyclic intervals they belong to (see Examples) and the number of internal knots must be greater or equal to the specified degree - 1. If internal knots are specified via knots, the specified df will be ignored.
knots	The internal breakpoints that define the splines. The default is NULL, which results in a basis for ordinary polynomial regression. Typical values are the mean or median for one knot, quantiles for more knots. For periodic splines, the number of knots must be greater or equal to the specified degree - 1. Duplicated internal knots are not allowed.
degree	The degree of C-spline defined to be the degree of the associated M-spline instead of actual polynomial degree. For example, C-spline basis of degree 2 is defined as the scaled double integral of associated M-spline basis of degree 2.

intercept	If TRUE by default, all of the spline basis functions are returned. Notice that when using C-Spline for shape-restricted regression, intercept = TRUE should be set even when an intercept term is considered additional to the spline basis in the model.
Boundary.knots	Boundary points at which to anchor the splines. By default, they are the range of x excluding NA. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots. For periodic splines, the specified bounary knots define the cyclic interval.
derivs	A nonnegative integer specifying the order of derivatives of C-splines. The default value is 0L for C-spline basis functions.
scale	A logical value indicating if scaling C-splines is required. If TRUE by default, each C-spline basis is scaled to have unit height at right boundary knot. The corresponding I-spline and M-spline produced by deriv methods will be scaled to the same extent.
warn.outside	A logical value indicating if a warning should be thrown out when any x is outside the boundary. This option can also be set through options("splines2.warn.outside") after the package is loaded.
...	Optional arguments that are not used.

Details

It is an implementation of the closed-form C-spline basis derived from the recursion formula of I-splines and M-splines. The function csp() is an alias of to encourage the use in a model formula.

Value

A numeric matrix of length(x) rows and df columns if df is specified. If knots are specified instead, the output matrix will consist of length(knots) + degree + as.integer(intercept) columns. Attributes that correspond to the arguments specified are returned for usage of other functions in this package.

References

Meyer, M. C. (2008). Inference using shape-restricted regression splines. *The Annals of Applied Statistics*, 2(3), 1013–1033.

See Also

iSpline for I-splines; mSpline for M-splines.

Examples

```
library(splines2)

x <- seq.int(0, 1, 0.01)
knots <- c(0.3, 0.5, 0.6)

### when 'scale = TRUE' (by default)
csMat <- cSpline(x, knots = knots, degree = 2)
```

```
plot(csMat, ylab = "C-spline basis", mark_knots = "internal")
isMat <- deriv(csMat)
msMat <- deriv(csMat, derivs = 2)
plot(isMat, ylab = "scaled I-spline basis")
plot(msMat, ylab = "scaled M-spline basis")

### when 'scale = FALSE'
csMat <- cSpline(x, knots = knots, degree = 2, scale = FALSE)

## the corresponding I-splines and M-splines (with same arguments)
isMat <- iSpline(x, knots = knots, degree = 2)
msMat <- mSpline(x, knots = knots, degree = 2, intercept = TRUE)

## or using deriv methods (more efficient)
isMat1 <- deriv(csMat)
msMat1 <- deriv(csMat, derivs = 2)

## equivalent
stopifnot(all.equal(isMat, isMat1, check.attributes = FALSE))
stopifnot(all.equal(msMat, msMat1, check.attributes = FALSE))
```

---

deriv *Derivatives of Spline Basis Functions*

Description

Returns derivatives of given order for the given spline basis functions.

Usage

```
## S3 method for class 'BSpline'
deriv(expr, derivs = 1L, ...)

## S3 method for class 'MSpline'
deriv(expr, derivs = 1L, ...)

## S3 method for class 'ISpline'
deriv(expr, derivs = 1L, ...)

## S3 method for class 'CSpline'
deriv(expr, derivs = 1L, ...)

## S3 method for class 'BernsteinPoly'
deriv(expr, derivs = 1L, ...)

## S3 method for class 'NaturalSpline'
deriv(expr, derivs = 1L, ...)
```

```
## S3 method for class 'NaturalSplineK'
deriv(expr, derivs = 1L, ...)
```

Arguments

expr	Objects of class bSpline2, ibs, mSpline, iSpline, cSpline, bernsteinPoly or naturalSpline with attributes describing knots, degree, etc.
derivs	A positive integer specifying the order of derivatives. By default, it is 1L for the first derivatives.
...	Optional arguments that are not used.

Details

At knots, the derivative is defined to be the right derivative except at the right boundary knot. By default, the function returns the first derivatives. For derivatives of order greater than one, nested function calls such as deriv(deriv(expr)) are supported but not recommended. For a better performance, argument derivs should be specified instead.

This function is designed for objects produced by this package. It internally extracts necessary specification about the spline/polynomial basis matrix from its attributes. Therefore, the function will not work if the key attributes are not available after some operations.

Value

A numeric matrix of the same dimension with the input expr.

Examples

```
library(splines2)

x <- c(seq.int(0, 1, 0.1), NA)  # NA's will be kept.
knots <- c(0.3, 0.5, 0.6)

## helper function
stopifnot_equivalent <- function(...) {
    stopifnot(all.equal(..., check.attributes = FALSE))
}

## integal of B-splines and the corresponding B-splines integrated
ibsMat <- ibs(x, knots = knots)
bsMat <- bSpline(x, knots = knots)

## the first derivative
d1Mat <- deriv(ibsMat)
stopifnot_equivalent(bsMat, d1Mat)

## the second derivative
d2Mat1 <- deriv(bsMat)
d2Mat2 <- deriv(ibsMat, derivs = 2L)
stopifnot_equivalent(d2Mat1, d2Mat2)
```

```
## nested calls are supported
d2Mat3 <- deriv(deriv(ibsMat))
stopifnot_equivalent(d2Mat2, d2Mat3)

## C-splines, I-splines, M-splines and the derivatives
csMat <- cSpline(x, knots = knots, intercept = TRUE, scale = FALSE)
isMat <- iSpline(x, knots = knots, intercept = TRUE)
stopifnot_equivalent(isMat, deriv(csMat))

msMat <- mSpline(x, knots = knots, intercept = TRUE)
stopifnot_equivalent(msMat, deriv(isMat))
stopifnot_equivalent(msMat, deriv(csMat, 2))
stopifnot_equivalent(msMat, deriv(deriv(csMat)))

dmsMat <- mSpline(x, knots = knots, intercept = TRUE, derivs = 1)
stopifnot_equivalent(dmsMat, deriv(msMat))
stopifnot_equivalent(dmsMat, deriv(isMat, 2))
stopifnot_equivalent(dmsMat, deriv(deriv(isMat)))
stopifnot_equivalent(dmsMat, deriv(csMat, 3))
stopifnot_equivalent(dmsMat, deriv(deriv(deriv(csMat))))
```

`iSpline` *I-Spline Basis for Polynomial Splines*

Description

Generates the I-spline (integral of M-spline) basis matrix for a polynomial spline or the corresponding derivatives of given order.

Usage

```
iSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = TRUE,
  Boundary.knots = NULL,
  derivs = 0L,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)

isp(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = TRUE,
```

```
  Boundary.knots = NULL,
  derivs = 0L,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)
```

Arguments

x	The predictor variable. Missing values are allowed and will be returned as they are.
df	Degree of freedom that equals to the column number of the returned matrix. One can specify df rather than knots, then the function chooses df - degree - as.integer(intercept) internal knots at suitable quantiles of x ignoring missing values and those x outside of the boundary. For periodic splines, df - as.integer(intercept) internal knots will be chosen at suitable quantiles of x relative to the beginning of the cyclic intervals they belong to (see Examples) and the number of internal knots must be greater or equal to the specified degree - 1. If internal knots are specified via knots, the specified df will be ignored.
knots	The internal breakpoints that define the splines. The default is NULL, which results in a basis for ordinary polynomial regression. Typical values are the mean or median for one knot, quantiles for more knots. For periodic splines, the number of knots must be greater or equal to the specified degree - 1. Duplicated internal knots are not allowed.
degree	The degree of I-spline defined to be the degree of the associated M-spline instead of actual polynomial degree. For example, I-spline basis of degree 2 is defined as the integral of associated M-spline basis of degree 2.
intercept	If TRUE by default, all of the spline basis functions are returned. Notice that when using I-Spline for monotonic regression, intercept = TRUE should be set even when an intercept term is considered additional to the spline basis functions.
Boundary.knots	Boundary points at which to anchor the splines. By default, they are the range of x excluding NA. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots. For periodic splines, the specified bounary knots define the cyclic interval.
derivs	A nonnegative integer specifying the order of derivatives of I-splines.
warn.outside	A logical value indicating if a warning should be thrown out when any x is outside the boundary. This option can also be set through options("splines2.warn.outside") after the package is loaded.
...	Optional arguments that are not used.

Details

It is an implementation of the closed-form I-spline basis based on the recursion formula given by Ramsay (1988). The function isp() is an alias of to encourage the use in a model formula.

Value

A numeric matrix of `length(x)` rows and `df` columns if `df` is specified. If `knots` are specified instead, the output matrix will consist of `length(knots)` + `degree` + `as.integer(intercept)` columns. Attributes that correspond to the arguments specified are returned for usage of other functions in this package.

References

Ramsay, J. O. (1988). Monotone regression splines in action. *Statistical Science*, 3(4), 425–441.

See Also

`mSpline` for M-splines; `cSpline` for C-splines;

Examples

```
library(splines2)

## an example given in Ramsay (1988)
x <- seq.int(0, 1, by = 0.01)
knots <- c(0.3, 0.5, 0.6)
isMat <- iSpline(x, knots = knots, degree = 2)

op <- par(mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
plot(isMat, ylab = "I-spline basis", mark_knots = "internal")
par(op) # reset to previous plotting settings

## the derivative of I-splines is M-spline
msMat1 <- iSpline(x, knots = knots, degree = 2, derivs = 1)
msMat2 <- mSpline(x, knots = knots, degree = 2, intercept = TRUE)
stopifnot(all.equal(msMat1, msMat2))
```

`knots` *Extract Knots from the Given Object*

Description

Methods for the generic function `knots` from the **stats** package to obtain internal or boundary knots from the objects produced by this package.

Usage

```
## S3 method for class 'splines2'
knots(Fn, type = c("internal", "boundary"), ...)
```

Arguments

Fn	An splines2 object produced by this package.
type	A character vector of length one indicating the type of knots to return. The available choices are "internal" for internal knots and "Boundary" for boundary knots.
...	Optional arguments that are not used now.

Value

A numerical vector.

Examples

```
library(splines2)

set.seed(123)
x <- rnorm(100)

## B-spline basis
bsMat <- bSpline(x, df = 8, degree = 3)

## extract internal knots placed based on the quantile of x
(internal_knots <- knots(bsMat))

## extract boundary knots placed based on the range of x
boundary_knots <- knots(bsMat, type = "boundary")
all.equal(boundary_knots, range(x))
```

---

mSpline *M-Spline Basis for Polynomial Splines*

Description

Generates the basis matrix of regular M-spline, periodic M-spline, and the corresponding integrals and derivatives.

Usage

```
mSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  Boundary.knots = NULL,
  periodic = FALSE,
  derivs = 0L,
```

```
  integral = FALSE,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)

msp(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  Boundary.knots = NULL,
  periodic = FALSE,
  derivs = 0L,
  integral = FALSE,
  warn.outside = getOption("splines2.warn.outside", TRUE),
  ...
)
```

Arguments

x	The predictor variable. Missing values are allowed and will be returned as they are.
df	Degree of freedom that equals to the column number of the returned matrix. One can specify df rather than knots, then the function chooses df - degree - as.integer(intercept) internal knots at suitable quantiles of x ignoring missing values and those x outside of the boundary. For periodic splines, df - as.integer(intercept) internal knots will be chosen at suitable quantiles of x relative to the beginning of the cyclic intervals they belong to (see Examples) and the number of internal knots must be greater or equal to the specified degree - 1. If internal knots are specified via knots, the specified df will be ignored.
knots	The internal breakpoints that define the splines. The default is NULL, which results in a basis for ordinary polynomial regression. Typical values are the mean or median for one knot, quantiles for more knots. For periodic splines, the number of knots must be greater or equal to the specified degree - 1. Duplicated internal knots are not allowed.
degree	A nonnegative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piecewise constant basis functions.
intercept	If TRUE, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
Boundary.knots	Boundary points at which to anchor the splines. By default, they are the range of x excluding NA. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots. For periodic splines, the specified bounary knots define the cyclic interval.
periodic	A logical value. If TRUE, the periodic splines will be returned. The default value is FALSE.

derivs	A nonnegative integer specifying the order of derivatives of splines basis function. The default value is 0.
integral	A logical value. If TRUE, the corresponding integrals of spline basis functions will be returned. The default value is FALSE. For periodic splines, the integral of each basis is integrated from the left boundary knot.
warn.outside	A logical value indicating if a warning should be thrown out when any x is outside the boundary. This option can also be set through options("splines2.warn.outside") after the package is loaded.
...	Optional arguments that are not used.

Details

This function contains an implementation of the closed-form M-spline basis based on the recursion formula given by Ramsay (1988) or periodic M-spline basis following the procedure producing periodic B-splines given in Piegl and Tiller (1997). For monotone regression, one can use I-splines (see iSpline) instead of M-splines. For shape-restricted regression, see Wang and Yan (2021) for examples.

The function msp() is an alias of to encourage the use in a model formula.

Value

A numeric matrix of length(x) rows and df columns if df is specified. If knots are specified instead, the output matrix will consist of length(knots) + degree + as.integer(intercept) columns if periodic = FALSE, or length(knots) + as.integer(intercept) columns if periodic = TRUE. Attributes that correspond to the arguments specified are returned for usage of other functions in this package.

References

Ramsay, J. O. (1988). Monotone regression splines in action. *Statistical science*, 3(4), 425–441.

Piegl, L., & Tiller, W. (1997). *The NURBS book*. Springer Science & Business Media.

Wang, W., & Yan, J. (2021). *Shape-restricted regression splines with R package splines2*. Journal of Data Science, 19(3),498–517.

See Also

bSpline for B-splines; iSpline for I-splines; cSpline for C-splines.

Examples

```
library(splines2)

### example given in the reference paper by Ramsay (1988)
x <- seq.int(0, 1, 0.01)
knots <- c(0.3, 0.5, 0.6)
msMat <- mSpline(x, knots = knots, degree = 2, intercept = TRUE)

op <- par(mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
```

```
plot(msMat, mark_knots = "internal")

## derivatives of M-splines
dmsMat <- mSpline(x, knots = knots, degree = 2,
                  intercept = TRUE, derivs = 1)

## or using the deriv method
dmsMat1 <- deriv(msMat)
stopifnot(all.equal(dmsMat, dmsMat1, check.attributes = FALSE))

### periodic M-splines
x <- seq.int(0, 3, 0.01)
bknots <- c(0, 1)
pMat <- mSpline(x, knots = knots, degree = 3, intercept = TRUE,
                Boundary.knots = bknots, periodic = TRUE)
## integrals
iMat <- mSpline(x, knots = knots, degree = 3, intercept = TRUE,
                Boundary.knots = bknots, periodic = TRUE, integral = TRUE)
## first derivatives by "derivs = 1"
dMat1 <- mSpline(x, knots = knots, degree = 3, intercept = TRUE,
                 Boundary.knots = bknots, periodic = TRUE, derivs = 1)
## first derivatives by using the deriv() method
dMat2 <- deriv(pMat)

par(mfrow = c(2, 2))
plot(pMat, ylab = "Periodic Basis", mark_knots = "boundary")
plot(iMat, ylab = "Integrals from 0")
abline(v = seq.int(0, max(x)), h = seq.int(0, max(x)), lty = 2, col = "grey")
plot(dMat1, ylab = "1st derivatives by 'derivs=1'", mark_knots = "boundary")
plot(dMat2, ylab = "1st derivatives by 'deriv()'", mark_knots = "boundary")
par(op) # reset to previous plotting settings

### default placement of internal knots for periodic splines
default_knots <- function(x, df, intercept = FALSE,
                          Boundary.knots = range(x, na.rm = TRUE)) {
    ## get x in the cyclic interval [0, 1)
    x2 <- (x - Boundary.knots[1]) %% (Boundary.knots[2] - Boundary.knots[1])
    knots <- quantile(x2, probs = seq(0, 1, length.out = df + 2 - intercept))
    unname(knots[- c(1, length(knots))])
}

df <- 8
degree <- 3
intercept <- TRUE
internal_knots <- default_knots(x, df, intercept)

## 1. specify df
spline_basis1 <- mSpline(x, degree = degree, df = df,
                         periodic = TRUE, intercept = intercept)
## 2. specify knots
spline_basis2 <- mSpline(x, degree = degree, knots = internal_knots,
                         periodic = TRUE, intercept = intercept)
```

```
all.equal(internal_knots, knots(spline_basis1))
all.equal(spline_basis1, spline_basis2)
```

---

naturalSpline *Natural Cubic Spline Basis for Polynomial Splines*

Description

Functions naturalSpline() and nsk() generate the natural cubic spline basis functions, the corresponding derivatives or integrals (from the left boundary knot). Both of them are different from splines::ns(). However, for a given model fitting procedure, using different variants of spline basis functions should result in identical prediction values. The coefficient estimates of the spline basis functions returned by nsk() are more interpretable compared to naturalSpline() or splines::ns() .

Usage

```
naturalSpline(
  x,
  df = NULL,
  knots = NULL,
  intercept = FALSE,
  Boundary.knots = NULL,
  trim = 0,
  derivs = 0L,
  integral = FALSE,
  ...
)

nsp(
  x,
  df = NULL,
  knots = NULL,
  intercept = FALSE,
  Boundary.knots = NULL,
  trim = 0,
  derivs = 0L,
  integral = FALSE,
  ...
)

nsk(
  x,
  df = NULL,
  knots = NULL,
  intercept = FALSE,
  Boundary.knots = NULL,
```

```
  trim = 0,
  derivs = 0L,
  integral = FALSE,
  ...
)
```

Arguments

x	The predictor variable. Missing values are allowed and will be returned as they are.
df	Degree of freedom that equals to the column number of returned matrix. One can specify df rather than knots, then the function chooses df - 1 - as.integer(intercept) internal knots at suitable quantiles of x ignoring missing values and those x outside of the boundary. Thus, df must be greater than or equal to 2. If internal knots are specified via knots, the specified df will be ignored.
knots	The internal breakpoints that define the splines. The default is NULL, which results in a basis for ordinary polynomial regression. Typical values are the mean or median for one knot, quantiles for more knots. For periodic splines, the number of knots must be greater or equal to the specified degree - 1. Duplicated internal knots are not allowed.
intercept	If TRUE, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
Boundary.knots	Boundary points at which to anchor the splines. By default, they are the range of x excluding NA. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots. For periodic splines, the specified bounary knots define the cyclic interval.
trim	The fraction (0 to 0.5) of observations to be trimmed from each end of x before placing the default internal and boundary knots. This argument will be ignored if Boundary.knots is specified. The default value is 0 for backward compatibility, which sets the boundary knots as the range of x. If a positive fraction is specified, the default boundary knots will be equivalent to quantile(x, probs = c(trim, 1 - trim), na.rm = TRUE), which can be a more sensible choice in practice due to the existence of outliers. The default internal knots are placed within the boundary afterwards.
derivs	A nonnegative integer specifying the order of derivatives of natural splines. The default value is 0L for the spline basis functions.
integral	A logical value. The default value is FALSE. If TRUE, this function will return the integrated natural splines from the left boundary knot.
...	Optional arguments that are not used.

Details

The constructed spline basis functions from naturalSpline() are nonnegative within boundary with the second derivatives being zeros at boundary knots. The implementation utilizes the close-form null space that can be derived from the recursive formula for the second derivatives of B-splines. The function nsp() is an alias of naturalSpline() to encourage the use in a model formula.

The function nsk() produces another variant of natural cubic spline matrix such that only one of the basis functions is nonzero and takes a value of one at every boundary and internal knot. As a result, the coefficients of the resulting fit are the values of the spline function at the knots, which makes it easy to interpret the coefficient estimates. In other words, the coefficients of a linear model will be the heights of the function at the knots if intercept = TRUE. If intercept = FALSE, the coefficients will be the change in function value between each knot. This implementation closely follows the function nsk() of the **survival** package (version 3.2-8). The idea corresponds directly to the physical implementation of a spline by passing a flexible strip of wood or metal through a set of fixed points, which is a traditional way to create smooth shapes for things like a ship hull.

The returned basis matrix can be obtained by transforming the corresponding B-spline basis matrix with the matrix H provided in the attribute of the returned object. Each basis is assumed to follow a linear trend for x outside of boundary. A similar implementation is provided by splines::ns, which uses QR decomposition to find the null space of the second derivatives of B-spline basis at boundary knots. See Supplementray Materials of Wang and Yan (2021) for a more detailed introduction.

Value

A numeric matrix of length(x) rows and df columns if df is specified or length(knots) + 1 + as.integer(intercept) columns if knots are specified instead. Attributes that correspond to the arguments specified are returned for usage of other functions in this package.

See Also

bSpline for B-splines; mSpline for M-splines; iSpline for I-splines.

Examples

```
library(splines2)

x <- seq.int(0, 1, 0.01)
knots <- c(0.3, 0.5, 0.6)

## naturalSpline()
nsMat0 <- naturalSpline(x, knots = knots, intercept = TRUE)
nsMat1 <- naturalSpline(x, knots = knots, intercept = TRUE, integral = TRUE)
nsMat2 <- naturalSpline(x, knots = knots, intercept = TRUE, derivs = 1)
nsMat3 <- naturalSpline(x, knots = knots, intercept = TRUE, derivs = 2)

op <- par(mfrow = c(2, 2), mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
plot(nsMat0, ylab = "basis")
plot(nsMat1, ylab = "integral")
plot(nsMat2, ylab = "1st derivative")
plot(nsMat3, ylab = "2nd derivative")
par(op) # reset to previous plotting settings

## nsk()
nskMat <- nsk(x, knots = knots, intercept = TRUE)
plot(nskMat, ylab = "nsk()", mark_knots = "all")
abline(h = 1, col = "red", lty = 3)
```

```
## use the deriv method
all.equal(nsMat0, deriv(nsMat1), check.attributes = FALSE)
all.equal(nsMat2, deriv(nsMat0))
all.equal(nsMat3, deriv(nsMat2))
all.equal(nsMat3, deriv(nsMat0, 2))

## a linear model example
fit1 <- lm(weight ~ -1 + nsk(height, df = 4, intercept = TRUE), data = women)
fit2 <- lm(weight ~ nsk(height, df = 3), data = women)

## the knots (same for both fits)
knots <- unlist(attributes(fit1$model[[2]])[c("Boundary.knots", "knots")])

## predictions at the knot points
predict(fit1, data.frame(height = sort(unname(knots))))
unname(coef(fit1)) # equal to the coefficient estimates

## different interpretation when "intercept = FALSE"
unname(coef(fit1)[-1] - coef(fit1)[1]) # differences: yhat[2:4] - yhat[1]
unname(coef(fit2))[-1]                 # ditto
```

`plot.splines2` *Visualize Spline Basis Functions*

Description

Plot spline basis functions by lines in different colors.

Usage

```
## S3 method for class 'splines2'
plot(
  x,
  y,
  from = NULL,
  to = NULL,
  n = 101,
  coef = NULL,
  mark_knots = c("none", "internal", "boundary", "all"),
  ...
)
```

Arguments

`x`	A `splines2` object.
`y`	An argument that is not used.
`from`, `to`	Two numbers representing the start and end point for the plot, respectively.
`n`	An integer, the number of x values at which to evaluate.

coef	A numeric vector specifying the coefficients of the spline basis functions. If it is NULL (by default), the spline basis functions will be plotted. Otherwise, the resulting spline function will be plotted.
mark_knots	A character vector specifying if knot placement should be indicated by vertical lines.
...	Additional arguments (other than x and y) that would be passed to matplot().

Details

This function is intended to quickly visualize the spline basis functions.

---

predict *Compute Spline Function for Given Coefficients*

Description

Returns the spline function (with the specified coefficients) or evaluate the basis functions at the specified x if the coefficients are not specified.

Usage

```
## S3 method for class 'BSpline'
predict(object, newx = NULL, coef = NULL, ...)

## S3 method for class 'MSpline'
predict(object, newx = NULL, coef = NULL, ...)

## S3 method for class 'ISpline'
predict(object, newx = NULL, coef = NULL, ...)

## S3 method for class 'CSpline'
predict(object, newx = NULL, coef = NULL, ...)

## S3 method for class 'BernsteinPoly'
predict(object, newx = NULL, coef = NULL, ...)

## S3 method for class 'NaturalSpline'
predict(object, newx = NULL, coef = NULL, ...)

## S3 method for class 'NaturalSplineK'
predict(object, newx = NULL, coef = NULL, ...)
```

Arguments

object	Spline objects produced by the splines2 package.
newx	The x values at which evaluations are required. If it is NULL (by default), the original x used to create the spline object will be used.

coef	A numeric vector specifying the coefficients of the spline basis functions. If it is NULL (by default), the spline basis functions will be returned. Otherwise, the resulting spline function will be returned.
...	Other options passed to the corresponding function that constructs the input object. For example, the additional options will be passed to bSpline() for a BSpline object.

Value

The function returns the spline basis functions with the new values of x if coef is not specified. Otherwise, the function returns the resulting spline function (or its derivative if derivs is specified as a positive integer through ...).

Examples

```
library(splines2)

x <- seq.int(0, 1, 0.2)
knots <- c(0.3, 0.5, 0.6)
newx <- seq.int(0.1, 0.9, 0.2)

## Cubic B-spline basis functions
bs_mat <- bSpline(x, knots = knots)

## compute the B-spline basis functions at new x
predict(bs_mat, newx)

## compute the B-spline function for the specified coefficients
beta <- runif(ncol(bs_mat))
predict(bs_mat, coef = beta)

## compute the first derivative of the B-spline function
predict(bs_mat, coef = beta, derivs = 1)
## or equivalently
predict(deriv(bs_mat), coef = beta)

## compute the second derivative
predict(bs_mat, coef = beta, derivs = 2)
## or equivalently
predict(deriv(bs_mat, derivs = 2), coef = beta)

## compute the integral
predict(bs_mat, coef = beta, integral = TRUE)
## or equivalently
predict(update(bs_mat, integral = TRUE), coef = beta)

## visualize
op <- par(mfrow = c(2, 2), mar = c(2.5, 2.5, 0.5, 0.1), mgp = c(1.5, 0.5, 0))
plot(bs_mat, coef = beta, ylab = "B-Spline Function", mark_knots = "all")
plot(deriv(bs_mat), coef = beta, ylab = "1st Derivative", mark_knots = "all")
plot(deriv(bs_mat, derivs = 2), coef = beta,
     ylab = "2nd Derivative", mark_knots = "all")
```

```
plot(update(bs_mat, integral = TRUE), coef = beta,
     ylab = "Integral", mark_knots = "all")
par(op)
```

---

update *Update Spline Basis Functions*

Description

Update the knot placement, polynomial degree, and any other options available when constructing the given spline object.

Usage

```
## S3 method for class 'BSpline'
update(object, ...)

## S3 method for class 'MSpline'
update(object, ...)

## S3 method for class 'ISpline'
update(object, ...)

## S3 method for class 'CSpline'
update(object, ...)

## S3 method for class 'BernsteinPoly'
update(object, ...)

## S3 method for class 'NaturalSpline'
update(object, ...)

## S3 method for class 'NaturalSplineK'
update(object, ...)
```

Arguments

object	Spline objects produced by the splines2 package.
...	Other arguments passed to the corresponing constructor function.

Value

An updated object of the same class as the input object with the specified updates.

Examples

```
library(splines2)

x <- seq.int(0, 1, 0.01)
knots <- c(0.3, 0.5, 0.6)

## quadratic B-splines
bsMat2 <- bSpline(x, knots = knots, degree = 2, intercept = TRUE)

## cubic B-splines
bsMat3 <- update(bsMat2, degree = 3)
```

Index

bernsteinPoly, 2
bpoly *(*bernsteinPoly*)*, 2
bsp *(*bSpline*)*, 4
bSpline, 4, *17*, *21*

csp *(*cSpline*)*, 7
cSpline, 7, *14*, *17*

dbs *(*bSpline*)*, 4
deriv, 10

ibs *(*bSpline*)*, 4
isp *(*iSpline*)*, 12
iSpline, *9*, 12, *17*, *21*

knots, *7*, 14

msp *(*mSpline*)*, 15
mSpline, *9*, *14*, 15, *21*

naturalSpline, 19
nsk *(*naturalSpline*)*, 19
nsp *(*naturalSpline*)*, 19

plot.splines2, 22
predict, 23

update, 25