Package ‘splines2’

September 19, 2021

Title  Regression Spline Functions and Classes
Version  0.4.5
Description  Constructs basis matrix of B-splines, M-splines, I-splines, convex splines (C-splines), periodic M-splines, natural cubic splines, generalized Bernstein polynomials, and their integrals (except C-splines) and derivatives of given order by close-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  Rcpp, stats
LinkingTo  Rcpp, RcppArmadillo
Suggests  knitr, rmarkdown, tinytest
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  7.1.1
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  CRAN
Date/Publication  2021-09-19 13:20:02 UTC
\textbf{bernsteinPoly}

R topics documented:

bernsteinPoly ............................................................... 2
bSpline ................................................................. 4
cSpline ................................................................. 6
dbs ................................................................. 8
deriv ................................................................. 10
ibs ................................................................. 12
iSpline ................................................................. 13
knots ................................................................. 15
mSpline ................................................................. 16
naturalSpline ............................................................. 20
predict ............................................................... 22
splines2 ................................................................. 23

Index 25

\begin{center}
\begin{tabular}{ll}
bernsteinPoly & \textit{Generalized Bernstein Polynomial Basis} \\
\end{tabular}
\end{center}

\textbf{Description}

Returns a generalized Bernstein polynomial basis matrix of given degree over a specified range.

\textbf{Usage}

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

\textbf{Arguments}

\begin{itemize}
  \item \textbf{x} \hspace{1cm} The predictor variable taking values inside of the specified boundary. Missing values are allowed and will be returned as they are.
  \item \textbf{degree} \hspace{1cm} A nonnegative integer representing the degree of the polynomials.
  \item \textbf{intercept} \hspace{1cm} If TRUE, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
  \item \textbf{Boundary.knots} \hspace{1cm} Boundary points at which to anchor the Bernstein polynomial basis. The default value is NULL and the boundary knots is set internally to be \texttt{range(x, na.rm = TRUE)}.
\end{itemize}
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.

Value
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), mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
matplot(x1, bMat1, type = "l", ylab = "y")
matplot(x2, bMat2, type = "l", ylab = "y")

## 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))
matplot(x1, d1Mat1, type = "l", ylab = "y")
matplot(x2, d1Mat2, type = "l", ylab = "y")
matplot(x1, d2Mat1, type = "l", ylab = "y")
matplot(x2, d2Mat2, type = "l", ylab = "y")

## 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 B-spline basis matrix representing the family of piecewise polynomials with the specified interior knots, degree, and boundary knots, evaluated at the values of \( x \).

**Usage**

```r
bSpline(
  x,  
  df = NULL,  
  knots = NULL,  
  degree = 3L,  
  intercept = FALSE,  
  Boundary.knots = NULL,  
  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 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. 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.

- **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`.

- **derivs**  
A nonnegative integer specifying the order of derivatives of B-splines. The default value is `0L` for B-spline basis functions.
integral  A logical value. If TRUE, the corresponding integrals of spline basis functions will be returned. The default value is FALSE.

... Optional arguments that are not used.

Details

This function extends the bs() function in the splines package for B-spline basis by allowing piecewise constant (left-closed and right-open except on the right boundary) spline basis of degree zero.

Value

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

References


See Also

dbs for derivatives of B-splines; ibs for integrals of B-splines;

Examples

library(splines2)

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

## cubic B-splines
bsMat <- bSpline(x, knots = knots, degree = 3, intercept = TRUE)

op <- par(mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
matplot(x, bsMat, type = "l", ylab = "Cubic B-splines")
abline(v = knots, lty = 2, col = "gray")

## reset to previous plotting settings
par(op)

## the first derivaitves
d1Mat <- deriv(bsMat)

## the second derivaitves
d2Mat <- deriv(bsMat, 2)

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

---

**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**

```r
cSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = TRUE,
  Boundary.knots = NULL,
  derivs = 0L,
  scale = 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. 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.
- **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.
- **derivs**  A nonnegative integer specifying the order of derivatives of C-splines. The default value is 0L for C-spline basis functions.
cSpline

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.

...  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.

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

References

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)

op <- par(mar = c(2.5, 2.5, 0.2, 0.1), mgp = c(1.5, 0.5, 0))
matplot(x, csMat, type = "l", ylab = "C-spline basis")
abline(v = knots, lty = 2, col = "gray")
isMat <- deriv(csMat)
msMat <- deriv(csMat, derivs = 2)
matplot(x, isMat, type = "l", ylab = "scaled I-spline basis")
matplot(x, msMat, type = "l", ylab = "scaled M-spline basis")

## reset to previous plotting settings
par(op)

### 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))

---

**dbs**

**Derivatives of B-Splines**

**Description**

Produces the derivatives of given order of B-splines.

**Usage**

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

**Arguments**

- **x**: The predictor variable. Missing values are allowed and will be returned as they are.
- **derivs**: A positive integer specifying the order of derivative. The default value is 1L for the first derivative.
- **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. 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.
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.
...  Optional arguments that are not used.

Details
This function provides a more user-friendly interface and a more consistent handling for NA’s than
splines::splineDesign() for derivatives of B-splines. The implementation is based on the
closed-form recursion formula. At knots, the derivative is defined to be the right derivative except
at the right boundary knot.

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

References

See Also
bSpline for B-splines; ibs for integrals of B-splines.

Examples
```r
library(splines2)
x <- seq.int(0, 1, 0.01)
knots <- c(0.2, 0.4, 0.7)
## the second derivative of cubic B-splines with three internal knots
dMat <- dbs(x, derivs = 2L, knots = knots, intercept = TRUE)

## compare with the results from splineDesign
ord <- attr(dMat, "degree") + 1L
bKnots <- attr(dMat, "Boundary.knots")
aKnots <- c(rep(bKnots[1L], ord), knots, rep(bKnots[2L], ord))
res <- splines::splineDesign(aKnots, x = x, derivs = 2L)
stopifnot(all.equal(res, dMat, check.attributes = FALSE))
```
**Derivatives of Spline Basis Functions**

**Description**

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

**Usage**

```r
## S3 method for class 'bSpline2'
deriv(expr, derivs = 1L, ...)
## S3 method for class 'dbs'
deriv(expr, derivs = 1L, ...)
## S3 method for class 'ibs'
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, ...)
```

**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**

```r
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
stophifnot_equivalent <- function(...) {
  stopifnot(all.equal(..., check.attributes = FALSE))
}

## integral 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)
stophifnot_equivalent(bsMat, d1Mat)

d2Mat1 <- deriv(bsMat)
d2Mat2 <- deriv(ibsMat, derivs = 2L)
stophifnot_equivalent(d2Mat1, d2Mat2)

## nested calls are supported
d2Mat3 <- deriv(deriv(ibsMat))
stophifnot_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)
msMat <- mSpline(x, knots = knots, intercept = TRUE)

msMat <- mSpline(x, knots = knots, intercept = TRUE, derivs = 1)
stophifnot_equivalent(msMat, deriv(deriv(isMat)))

stopifnot_equivalent(msMat, deriv(deriv(isMat)))
stopifnot_equivalent(msMat, deriv(deriv(deriv(csMat)))))
```
ibs

Integrals of B-Splines

Description

Generates basis matrix for integrals of B-splines.

Usage

```r
ibs(
x, 
df = NULL, 
knots = NULL, 
degree = 3, 
intercept = FALSE, 
Boundary.knots = NULL, 
...
)
```

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. 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.
- **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`.
- **...**: Optional arguments that are not used.

Details

The implementation is based on the closed-form recursion formula.
**Value**

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

**References**


**See Also**

`bSpline` for B-splines; `dbs` for derivatives of B-splines;

**Examples**

```r
library(splines2)

x <- seq.int(0, 1, 0.01)
knots <- c(0.2, 0.4, 0.7, 0.9)
ibsMat <- ibs(x, knots = knots, degree = 1, intercept = TRUE)

## get the corresponding B-splines by bSpline()
bsMat0 <- bSpline(x, knots = knots, degree = 1, intercept = TRUE)
## or by the deriv() method
bsMat <- deriv(ibsMat)
stopifnot(all.equal(bsMat0, bsMat, check.attributes = FALSE))

## plot B-spline basis with their corresponding integrals
op <- par(mfrow = c(1, 2))
matplot(x, bsMat, type = "l", ylab = "B-spline basis")
abline(v = knots, lty = 2, col = "gray")
matplot(x, ibsMat, type = "l", ylab = "Integral of B-spline basis")
abline(v = knots, lty = 2, col = "gray")

## reset to previous plotting settings
par(op)
```

---

**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**

```r
iSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = TRUE,
  Boundary.knots = NULL,
  derivs = 0L,
  ...
)
```

**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. 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.
- **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`.
- **derivs**: A nonnegative integer specifying the order of derivatives of I-splines.
- **...**: 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).

**Value**

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

References


See Also

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

Examples

```r
library(splines2)

## Example given in the reference paper by 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))
matplot(x, isMat, type = "l", ylab = "I-spline basis")
abline(v = knots, lty = 2, col = "gray")

## reset to previous plotting settings
par(op)

## 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

```r
## S3 method for class 'bSpline2'
knots(Fn, type = c("internal", "boundary"), ...)

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

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

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

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

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

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

## S3 method for class 'naturalSpline'
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

```r
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**
mSpline

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,
  ...
)

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. For M-splines, 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 M-spline (periodic = TRUE), 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 (periodic = TRUE), the number of knots must be greater or equal to the specified degree -1.

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 (periodic = TRUE), the specified boundary knots define the cyclic interval.
periodic  A logical value. If TRUE, the periodic splines will be returned instead of regular M-splines. The default value is FALSE.
derivs    A nonnegative integer specifying the order of derivatives of M-splines. The default value is 0L for M-spline basis functions.
derivs    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.
...      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.

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


See Also

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

Examples

```r
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))
matplot(x, msMat, type = "l", ylab = "y")
abline(v = knots, lty = 2, col = "gray")

## 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))
matplot(x, pMat, type = "l", ylab = "Periodic Basis")
abline(v = seq.int(0, max(x)), lty = 2, col = "grey")
matplot(x, iMat, type = "l", ylab = "Integrals from 0")
abline(v = seq.int(0, max(x)), h = seq.int(0, max(x)), lty = 2, col = "grey")
matplot(x, dMat1, type = "l", ylab = "1st derivatives by derivs=1")
abline(v = seq.int(0, max(x)), lty = 2, col = "grey")
matplot(x, dMat2, type = "l", ylab = "1st derivatives by deriv()")
abline(v = seq.int(0, max(x)), lty = 2, col = "grey")

## reset to previous plotting settings
par(op)

### 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 = splines2::mSpline(x, degree = degree, df = df,
  periodic = TRUE, intercept = intercept)

## 2. specify knots
spline_basis2 = splines2::mSpline(x, degree = degree, knots = internal_knots,
  periodic = TRUE, intercept = intercept)

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

Generates the nonnegative natural cubic spline basis matrix, the corresponding integrals (from the left boundary knot), or derivatives of given order. Each basis is assumed to follow a linear trend for x outside of boundary.

Usage

naturalSpline(
  x,
  df = NULL,
  knots = NULL,
  intercept = FALSE,
  Boundary.knots = NULL,
  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.
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.
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.
naturalSpline

Details

It is an implementation of the natural spline basis based on B-spline basis, which utilizes the close-form null space that can be derived from the recursive formula for the second derivatives of B-splines. The constructed spline basis functions are intended to be nonnegative within boundary with second derivatives being zeros at boundary knots.

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. However, there is no guarantee that the resulting basis functions are nonnegative within boundary.

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

```r
library(splines2)
x <- seq.int(0, 1, 0.01)
knots <- c(0.3, 0.5, 0.6)

## natural spline basis
nsMat0 <- naturalSpline(x, knots = knots, intercept = TRUE)

## integrals
nsMat1 <- naturalSpline(x, knots = knots, intercept = TRUE, integral = TRUE)

## first derivatives
nsMat2 <- naturalSpline(x, knots = knots, intercept = TRUE, derivs = 1)

## second derivatives
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))
matplot(x, nsMat0, type = "l", ylab = "basis")
matplot(x, nsMat1, type = "l", ylab = "integral")
matplot(x, nsMat2, type = "l", ylab = "1st derivative")
matplot(x, nsMat3, type = "l", ylab = "2nd derivative")
par(op) # reset to previous plotting settings

## 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))
```
predict

Evaluate a Spline Basis at specified points

Description

This function evaluates a predefined spline basis at a (new) given \( x \).

Usage

## S3 method for class 'bSpline2'
predict(object, newx, ...)

## S3 method for class 'dbs'
predict(object, newx, ...)

## S3 method for class 'ibs'
predict(object, newx, ...)

## S3 method for class 'mSpline'
predict(object, newx, ...)

## S3 method for class 'iSpline'
predict(object, newx, ...)

## S3 method for class 'cSpline'
predict(object, newx, ...)

## S3 method for class 'bernsteinPoly'
predict(object, newx, ...)

## S3 method for class 'naturalSpline'
predict(object, newx, ...)

Arguments

- **object**: Objects of class bSpline2, ibs, mSpline, iSpline, cSpline, bernsteinPoly or naturalSpline with attributes describing knots, degree, etc.
- **newx**: The \( x \) values at which evaluations are required.
- **...**: Optional arguments at which evaluations are not used.

Details

These are methods for the generic function predict for objects inheriting from class bSpline2, ibs, mSpline, iSpline, cSpline, naturalSpline, or bernsteinPoly. If newx is not given, the function returns the input object.
Value

An object just like the object input, except evaluated at the new values of x.

Examples

```r
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)

## for B-splines
bsMat <- bSpline(x, knots = knots, degree = 2)
predict(bsMat, newX)

## for integral of B-splines
ibsMat <- ibs(x, knots = knots, degree = 2)
predict(ibsMat, newX)

## for derivative of B-splines
dbsMat <- dbs(x, knots = knots, degree = 2)
predict(dbsMat, newX)

## for M-spline
msMat <- mSpline(x, knots = knots, degree = 2)
predict(msMat, newX)

## for I-spline
isMat <- iSpline(x, knots = knots, degree = 2)
predict(isMat, newX)

## for C-spline
csMat <- cSpline(x, knots = knots, degree = 2)
predict(csMat, newX)
```

Description

This package provides functions to construct basis matrices of

- B-splines
- M-splines
- I-splines
- convex splines (C-splines)
- periodic M-splines
- natural cubic splines
• generalized Bernstein polynomials
• along with their integrals (except C-splines) and derivatives of given order by closed-form recursive formulas

Details

In addition to the R interface, it also provides a C++ header-only library integrated with Rcpp, which allows the construction of spline basis functions directly in C++ with the help of Rcpp and RcppArmadillo. Thus, it can also be treated as one of the Rcpp* packages. A toy example package that uses the C++ interface is available at <https://github.com/wenjie2wang/example-pkg-Rcpp-splines2>.

The package splines2 is intended to be a user-friendly supplement to the base package splines. The trailing number two in the package name means "too" (and by no means refers to the generation two). See Wang and Yan (2021) for details and illustrations of how the package can be applied to shape-restricted regression.

References

Index

bernsteinPoly, 2
bSpline, 4, 9, 13, 18, 21

cSpline, 6, 15, 18

dbs, 5, 8, 13
deriv, 10

ibs, 5, 9, 12
iSpline, 7, 13, 18, 21

knots, 15

mSpline, 7, 15, 16, 21

naturalSpline, 20

predict, 22

splines2, 23