Package 'splines2'

July 14, 2020

Title Regression Spline Functions and Classes

Version 0.3.1

Date 2020-07-13

Description Constructs B-splines and its integral, M-splines
and its integral (I-splines), convex splines (C-splines),
generalized Bernstein polynomials, and their derivatives.
It also contains a C++ head-only library integrated with Rcpp.
See De Boor (1978) <doi:10.1002/zamm.19800600129>,
Ramsay (1988) <doi:10.1214/ss/1177012761>, and
Meyer (2008) <doi:10.1214/08-AOAS167>
for more information about the spline basis.

Imports Rcpp, stats

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, tinytest

Depends R (>= 3.2.3)

VignetteBuilder knitr

License GPL (>= 3)

URL 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 <wjwang.stat@gmail.com>

Repository CRAN

Date/Publication 2020-07-14 13:30:02 UTC

R topics documented:

bernsteinPoly 2
bSpline 4
cSpline 5
dbs 7
deriv 9
ibs 11
iSpline 13
mSpline 14
predict 16
splines2 18

Index **19**

`bernsteinPoly` *Generalized Bernstein Polynomial Basis*

Description

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

Usage

```
bernsteinPoly(
  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 non-negative 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 non-negative integer specifying the order of derivatives. The default value is `0L` for Bernstein polynomial bases.

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)

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")

## 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 and degree, evaluated at the values of x.

Usage

```
bSpline(
  x,
  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.
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 -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 spline. 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 non-negative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piece-wise constant bases.
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 spline basis. By default, they are the range of the non-NA data. 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 extends the bs() function in splines package for B-spline basis by allowing piecewise constant (left-closed and right-open except on the right boundary) spline basis with zero degree.

Value

A numeric matrix with `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 for usage of other functions in this package.

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)

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-spline Bases")
abline(v = knots, lty = 2, col = "gray")

## 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 *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,
  ...
)
```

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 -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 spline. 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 spline bases are included. 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 bases in the model.
Boundary.knots	Boundary points at which to anchor the spline basis. By default, they are the range of the non-NA data. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots.
derivs	A non-negative integer specifying the order of derivatives of C-splines. The default value is 0L for C-spline bases.
scale	Logical value (TRUE by default) indicating whether scaling on C-spline basis is required. If TRUE, C-spline basis is scaled to have unit height at right boundary knot; the corresponding I-spline and M-spline basis matrices shipped in attributes are also scaled to the same extent.
...	Optional arguments that are not used.

Details

It is an implementation of the close form C-spline basis derived from the recursion formula of I-splines and M-splines.

Value

A numeric matrix with 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 for usage of other functions in this package.

References

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

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)

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")

### 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-Spline Basis*

Description

Produces the derivatives of given order of B-splines.

Usage

```
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. By default, it is 1L for the first derivative.
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 -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 spline. 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 non-negative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piece-wise constant bases.
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 spline basis. By default, they are the range of the non-NA data. 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 close form recursion formula. At knots, the derivative is defined to be the right derivative.

Value

A numeric matrix with 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 for usage of other functions in this package.

References

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

See Also

`bSpline` for B-splines; `ibs` for integrals of B-splines.

Examples

```
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))
```

---

deriv *Derivatives of Spline Bases*

Description

Returns derivatives of given order for the spline bases.

Usage

```
## 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, ...)
```

Arguments

expr	Objects of class bSpline2, ibs, dbs, mSpline, iSpline, or cSpline, etc.
derivs	A positive integer specifying the order of derivatives. By default, it is 1L for the first derivative.
...	Other arguments that are not used.

Details

At knots, the derivative is defined to be the right derivative. By default, the function returns the first derivatives. For derivatives of order greater than one, the nested call such as deriv(deriv(expr)) is 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 attributions 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)

## 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(all.equal(bsMat, d1Mat, check.attributes = FALSE))

## the second derivative
d2Mat1 <- deriv(bsMat)
d2Mat2 <- deriv(ibsMat, derivs = 2L)
## nested calls are supported but not recommended
d2Mat3 <- deriv(deriv(ibsMat))
stopifnot(all.equal(d2Mat1, d2Mat2, check.attributes = FALSE))
stopifnot(all.equal(d2Mat2, d2Mat3, check.attributes = FALSE))

## 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(all.equal(isMat, deriv(csMat), check.attributes = FALSE))

msMat <- mSpline(x, knots = knots, intercept = TRUE)
stopifnot(all.equal(msMat, deriv(isMat), check.attributes = FALSE))
stopifnot(all.equal(msMat, deriv(csMat, 2), check.attributes = FALSE))
stopifnot(all.equal(msMat, deriv(deriv(csMat)), check.attributes = FALSE))

dmsMat <- mSpline(x, knots = knots, intercept = TRUE, derivs = 1)
stopifnot(all.equal(dmsMat, deriv(msMat), check.attributes = FALSE))
stopifnot(all.equal(dmsMat, deriv(isMat, 2), check.attributes = FALSE))
stopifnot(all.equal(dmsMat, deriv(deriv(isMat)), check.attributes = FALSE))
stopifnot(all.equal(dmsMat, deriv(csMat, 3), check.attributes = FALSE))
stopifnot(all.equal(dmsMat, deriv(deriv(deriv(csMat))), check.attributes = FALSE))
```

---

ibs *Integrals of B-Spline Basis*

Description

Generates the integrals of B-spline basis matrix.

Usage

```
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 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 spline. 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 non-negative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piece-wise constant bases.

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 spline basis. By default, they are the range of the non-NA data. 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 close form recursion formula.

Value

A numeric matrix with 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 for usage of other functions in this package.

References

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

See Also

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

Examples

```
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)

## the B-spline bases integrated by function bSpline (same arguments)
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
library(graphics)
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")
par(mfrow = c(1, 1))
```

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,
  ...
)
```

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 -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 spline. 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 spline bases are included. 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 bases in the model.
Boundary.knots	Boundary points at which to anchor the spline basis. By default, they are the range of the non-NA data. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots.
derivs	A non-negative integer specifying the order of derivatives of I-splines.
...	Optional arguments that are not used.

Details

It is an implementation of the close form I-spline basis based on the recursion formula given by Ramsay (1988).

Value

A numeric matrix with `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 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)

## 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)

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")

## 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))
```

`mSpline` *M-Spline Basis for Polynomial Splines*

Description

Generates the basis matrix of the regression spline called M-spline or the corresponding derivatives of given order. For monotone regression, `iSpline` should be used instead of M-splines.

Usage

```
mSpline(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  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 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 spline. 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 non-negative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piece-wise constant bases.
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 spline basis. By default, they are the range of the non-NA data. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots.
derivs	A non-negative integer specifying the order of derivatives of M-splines. The default value is 0L for M-spline bases.
...	Optional arguments that are not used.

Details

It is an implementation of the close form M-spline basis based on the recursion formula given by Ramsay (1988).

Value

A numeric matrix with 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 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

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)

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))
```

---

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 'ibs'
predict(object, newx, ...)

## S3 method for class 'dbs'
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, ...)
```

Arguments

object	Objects of class bSpline2, ibs, mSpline, iSpline, cSpline, or bernsteinPoly with attributes describing knots, degree, etc.
newx	The x values at which evaluations are required.
...	Optional argument that are not used.

Details

These are methods for the generic function predict for objects inheriting from class bSpline2, ibs, mSpline, iSpline, cSpline, 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.

See Also

bSpline for B-splines; ibs for integrals of B-splines; dbs for derivatives of B-splines; mSpline for M-splines; iSpline for I-splines; cSpline for C-splines.

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)

## 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)
```

---

splines2 *splines2: Regression Spline Functions and Classes*

Description

This package provides functions to construct basis matrix of

- B-splines
- M-splines
- I-splines
- convex splines (C-splines)
- generalized Bernstein polynomials
- their integrals (except C-splines) and derivatives of given order by close-form recursive formulas

Details

In addition to the R interface, it also provides a C++ header-only library integrated with **Rcpp**, which allows construction of spline basis matrix directly in C++ with the help of **Rcpp** and **RcppArmadillo**. So 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>.

It is named after the **splines** package: "Regression Spline Functions and Classes". The tailing number two is simply "too" (and by no means for the generation two).

Index

bernsteinPoly, 2
bSpline, 4, *9*, *12*, *16, 17*

cSpline, 5, *14*, *16, 17*

dbs, *5*, 7, *12*, *17*
deriv, 9

ibs, *5*, *9*, 11, *17*
iSpline, *7*, 13, *14*, *16, 17*

mSpline, *7*, *14*, 14, *17*

predict, 16

splines2, 18