R documentation

of all in ‘man’

May 9, 2018

R topics documented:

`bsplineBasis` *Regression using B-spline basis*

Description

This function provides nonparametric regressions using B-splines. The B-splines are defined following the recursive formulas due to de Boor. Only univariate input can be used.

Usage

```
bsplineBasis(x, y, x_test, order = 4, innerknots)
```

Arguments

`x`	The input vector of training dataset.
`y`	The output vector of training dataset.
`x_test`	The input values at which evaluations are required.
`order`	The order of B-spline functions. The default is order=4 for cubic B-splines.
`innerknots`	The internal knots that define the spline.

Value

A list with the following components:

beta	The coefficients of nonparametric regression.
basis	The B-spline basis matrix of dimension c(length(x), df). df = length(innerknots) + order.
f	The evaluated output at x_test.

Examples

```
x<-seq(0, 1, 0.001)
y <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)

innerknots <- c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)
order<-4
x_test<-seq(0, 1, 0.01)

b_fit<-bspline(x,y,x_test,order,innerknots)

plot(x_test,b_fit$f)
lines(x_test,x_test^3 * 3 - x_test^2 * 2 + x_test + exp(1),col="red")

plot(x,rep(0,length(x)),type="l",ylim=c(0,1))
for (i in 1: (j+order)){
lines(x,b_fit$basis[,i])
}
```

CubicPowerBasisSpline *Regression using cubic spline*

Description

This function provides regressions using cubic splines. The cubic splines are defined as h1 = 1,h2 = x,h3 = x^2,h4 = x^3,h5 = (x-k1)^3+,h6 = (x-k2)^3+,..., where k1, k2 and kn are n knots, '+' denotes the positive part.

Usage

```
CubicPowerBasisSpline(x, y, x_test, innerknots)
```

Arguments

x	The input vector of training dataset.
y	The output vector of training dataset.
x_test	The input values at which evaluations are required.
innerknots	The internal knots that define the spline.

Details

Only univariate input can be used.

Value

A list with the following components:

beta	The coefficients of nonparametric regression.
basis	The cubic spline basis matrix of dimension c(length(x), NumKnots+4)
f	The evaluated output at x_test.

Examples

```
n <- 100
t <- seq(0,2*pi,length.out = 100)
a <- 3
b <- 2
c.unif <- runif(n)
amp <- 2
set.seed(1)
y1 <- a*sin(b*t)+c.unif*amp # uniform error
innerknots <- 2*pi*c(1/4,2/4,3/4)
solution <- CubicPowerBasisSpline(t,y1,t,innerknots)
y.hat <- solution$f
plot(t, y1, t="l")
lines(t, y.hat, col=4)
```

natural_cubic_splines *Regression using natural cubic splines*

Description

This function provides regressions using natural cubic splines with truncated power basis functions. Only univariate input can be used.

Usage

```
natural_cubic_splines(x_train, y_train, x_test, df = NULL, knots = NULL)
```

Arguments

x_train	The input vector of training dataset.
y_train	The output vector of training dataset.
x_test	The input values at which evaluations are required.
df	The degree of freedom specified by user, number of knots will be equal to df.
knots	Knots location in terms of quantiles of x_train, optional, default will be evenly spaced quantiles based on number of knots.

Value

y_pred	A vector of dimension length(x) The prediction vector evaluated at x_test values

Examples

```
x_train <-seq(0, 1, 0.001)
y_train <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)
df <- 10
x_test <- seq(0, 1, 0.01)
y_pred <- natural_cubic_splines(x, y, x_test, df)
plot(x_test,y_pred)
lines(x_test,x_test^3 * 3 - x_test^2 * 2 + x_test + exp(1),col="red")
```

---

natural_cubic_splines.eval_basis

Evaluate basis functions as each x and return the evaluated basis matrix N

Description

Evaluate basis functions as each x and return the evaluated basis matrix N

Usage

```
natural_cubic_splines.eval_basis(x, knots, nknots)
```

Arguments

x	Predictor variable vector
knots	Knots location in terms of quantiles of x_train, optional, default will be evenly spaced quantiles based on number of knots
nknots	Number of knots useded in training.

Value

Basis matrix evaluated at each x value

---

natural_cubic_splines.predict

Prediction based on trained regression model

Description

Prediction based on trained regression model

Usage

```
natural_cubic_splines.predict(x_test, betas, knots, nknots)
```

Arguments

`x_test`	The input values at which evaluations are required.
`betas`	Least sqaure fit parameters obtained from training.
`knots`	Knots location in terms of quantiles of x_train, optional, default will be evenly spaced quantiles based on number of knots
`nknots`	Number of knots used in training.

Value

`y_pred`	A vector of dimension length(x)The prediction vector evaluated at x_test values

---

`natural_cubic_splines.train` *Generate an evaluated basis matrix for natural cubic splines*

Description

Generate an evaluated basis matrix for natural cubic splines

Usage

```
natural_cubic_splines.train(x_train, y_train, df = NULL, knots = NULL,
  intercept = FALSE)
```

Arguments

`x_train`	The input vector of training dataset.
`y_train`	The output vector of training dataset.
`df`	The degree of freedom specified by user, number of knots will be equal to df.
`knots`	Knots location in terms of quantiles of x_train, optional, default will be evenly spaced quantiles based on number of knots
`intercept`	Default false, do not change.

Value

A list of following components:

`knots`

`N`

`betas`

Examples

```
x_train <-seq(0, 1, 0.001)
y_train <- x^3 * 3 - x^2 * 2 + x + exp(1)+rnorm(length(x),0,0.1)
plot(x,y)
df <- 10
x_test <- seq(0, 1, 0.01)
train_result <- natural_cubic_splines.train(x, y, df)
print(train_result$betas)
print(train_result$N[1:5,1:5])
```

`place_knots` *Find evenly spaced out knots by quantile*

Description

Find evenly spaced out knots by quantile

Usage

```
place_knots(nknots, x)
```

Arguments

`nknots`	Number of knots to be located.
`x`	Data vector on which knots are placed.

Value

A named vector with knot quantiles and values

Index