Package ‘gps’

April 15, 2022

Version 1.1

Date 2022-04-15

Title General P-Splines

Author Zheyuan Li [aut, cre] (<`https://orcid.org/0000-0002-7434-5947`>),
Jiguo Cao [fnd] (<`https://orcid.org/0000-0001-7417-6330`>)

Maintainer Zheyuan Li <`zheyuan.li@bath.edu`>

Depends R (>= 4.0.0)

Imports stats, splines, Matrix, methods, graphics, grDevices

Description Routines to construct general P-splines for non-uniform B-splines on arbitrary knots, proposed by Li and Cao (2022) <arXiv:2201.06808>. This P-spline variant extends the standard P-splines of Eilers and Marx (1996) <doi:10.1214/ss/1038425655> that are tailored for uniform B-splines on equidistant knots. Includes SparseD() for computing general difference matrices, SparseS() for computing derivative penalty matrix or its sparse root using a sandwich formula, etc. Also includes several demos on B-splines and P-splines. Aims to facilitate other packages to implement general P-splines as a smoothing tool in their model estimation framework.

NeedsCompilation yes

URL `https://github.com/ZheyuanLi/gps`

R topics documented:

DemoBS *Demonstrate construction of ordinary B-splines*

Description

Demonstrate construction of 4 ordinary cubic B-splines on 8 knots.

Usage

```
DemoBS(uniform = TRUE, clamped = FALSE)
```

Arguments

`uniform`	TRUE for uniform knots and FALSE for non-uniform knots.
`clamped`	TRUE for clamped boundary knots (for aesthetic reason only boundary knots on the left end are clamped).

Value

This function has no returned values.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

Examples

```
require(gps)

## uniform B-splines
DemoBS(uniform = TRUE)

## non-uniform B-splines
DemoBS(uniform = FALSE, clamped = FALSE)

## non-uniform & clamped B-splines
DemoBS(uniform = FALSE, clamped = TRUE)
```

`DemoKnots` *Demonstrate ordinary cubic B-splines on three types of knots*

Description

Demonstrate ordinary cubic B-splines on three types of knots: (a) uniform knots; (b) non-uniform knots; (c) non-uniform knots with clamped boundary knots.

Usage

```
DemoKnots(aligned = TRUE)
```

Arguments

`aligned`	if TRUE, case (b) and (c) are aligned for nice display, because they have identical interior knots.

Value

This function has no returned values.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

DemoKnots(aligned = TRUE)
```

`DemoNull` *Demonstrate null space of a P-spline*

Description

Demonstrate the limiting behaviour of a P-spline at $\lambda = +\infty$. A cubic P-spline set up with non-uniform B-splines and 2nd order difference penalty is fitted to observations simulated from $y = x$. If this P-spline has the correct null space, its limiting fit will be a straight line regardless of knot locations. In this demo, non-uniform knots from different distributions (primarily Beta distributions with varying shape parameters) are attempted. The result shows that when handling non-uniform B-splines, standard P-spline has an incorrect and unpredictable limiting behavior that is sensitive to knot locations, whereas general P-spline has a correct and consistent limiting behavior.

Usage

```
DemoNull(n, k, gps = FALSE)
```

Arguments

`n`	number of simulated observations from $y = x$.
`k`	number of interior knots to place.
`gps`	TRUE to fit general P-spline; FALSE to fit standard P-spline.

Value

This function has no returned values.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## standard P-spline
DemoNull(n = 100, k = 10, gps = FALSE)

## general P-spline
DemoNull(n = 100, k = 10, gps = TRUE)
```

`DemoPBS` *Demonstrate construction of periodic B-splines*

Description

Demonstrate construction of 6 periodic cubic B-splines on 7 domain knots.

Usage

```
DemoPBS(uniform = TRUE)
```

Arguments

`uniform`	TRUE for uniform knots and FALSE for non-uniform knots.

Value

This function has no returned values.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## uniform periodic cubic B-splines
DemoPBS(uniform = TRUE)

## non-uniform periodic cubic B-splines
DemoPBS(uniform = FALSE)
```

`DemoSpl` *Demonstrate polynomial spline and its B-spline representation*

Description

Demonstrate cubic spline and its B-spline representation.

Usage

```
DemoSpl(uniform = TRUE)
```

Arguments

`uniform`	TRUE for uniform knots and FALSE for non-uniform knots.

Value

A list giving domain knots, B-spline coefficients and piecewise polynomial coefficients.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## cubic spline with uniform knots
DemoSpl(uniform = TRUE)

## cubic spline with non-uniform knots
DemoSpl(uniform = FALSE)
```

`DReigen` *Compute exact Demmler-Reinsch eigenvalues*

Description

`Rho2EDF` maps smoothing parameter value to effective degree of freedom using Demmler-Reinsch eigenvalues

Usage

```
DReigen(E)

Rho2EDF(values, rho)
```

Arguments

`E`	in the returned list of `gps2Grid`.
`values`	a sequence of Demmler-Reinsch eigenvalues.
`rho`	a grid of log smoothing parameters.

Value

A sequence of Demmler-Reinsch eigenvalues.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

x <- rnorm(100)
xt <- PlaceKnots(x, d = 4, k = 10)

## set ng = 0 to skip solving penalized least squares
## y = x
setup <- gps2Grid(x, x, xt = xt, d = 4, m = 2, ng = 0)

## compute exact eigenvalues
ei <- DReigen(setup$E)

## compare approximated eigenvalues with exact eigenvalues on the log scale
log.exact <- log(ei$values)
log.approx <- log(setup$eigen$approx.values)
op <- par(mar = c(2, 2, 1.5, 0.5))
plot(log.exact, pch = 19, col = 2, ann = FALSE)
points(log.approx, pch = 19)
```

```
legend("topright", legend = c("approx", "exact"), col = 1:2, pch = 19)
title("log(eigenvalues)")
par(op)
```

`gps2Grid` *General P-spline estimation via grid search*

Description

Fit a general P-spline to (x, y, w) for an automatically generated grid of log smoothing parameter values.

Usage

```
gps2Grid(x, y, w = NULL, xt, d = 4, m = 2, deriv.pen = FALSE,
         boundary = "none", ng = 20)
```

Arguments

`x, y, w`	a vector of x-values, y-values and weights.
`xt`	full knot sequence for ordinary B-splines (`length(xt)` >= `2 * d` required).
`d`	B-spline order ($d \geq 2$ required).
`m`	penalty order ($1 \leq m \leq d - 1$ required).
`deriv.pen`	TRUE to use derivative penalty; FALSE to use general difference penalty.
`boundary`	type of boundary constraint: "none", "natural" (applicable only to B-splines of even order) or "periodic".
`ng`	number of grid points; can be set to 0 to

Value

to do

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## simulate 100 (x, y) data from g(x) = sin(2 * pi * x) on [0, 1]
## x-values are not equidistant but at quantiles of Beta(2, 2)
## note that g(x) is a periodic function
x <- qbeta(seq.int(0, 1, length.out = 100), 2, 2)
gx <- sin(2 * pi * x)
y <- rnorm(length(x), gx, sd = 0.1)
```

```
## place quantile knots with clamped boundary knots
xt <- PlaceKnots(x, d = 4, k = 10)

## fit a general P-spline with different boundary constraints
ordinary <- gps2Grid(x, y, xt = xt, d = 4, m = 2)
natural <- gps2Grid(x, y, xt = xt, d = 4, m = 2, boundary = "natural")
periodic <- gps2Grid(x, y, xt = xt, d = 4, m = 2, boundary = "periodic")

## identify the optimal fit minimizing GCV score
opt.ordinary <- which.min(ordinary$pls$gcv)
opt.natural <- which.min(natural$pls$gcv)
opt.periodic <- which.min(periodic$pls$gcv)

## inspect grid search result
## column 1: ordinary cubic spline
## column 2: natural cubic spline
## column 3: periodic cubic spline
op <- par(mfcol = c(2, 3), mar = c(2, 2, 1.5, 0.5))
## ordinary spline
with(ordinary$pls, plot(rho, edf, ann = FALSE))
title("edf v.s. log(lambda)")
with(ordinary$pls, plot(rho, gcv, ann = FALSE))
with(ordinary$pls, points(rho[opt.ordinary], gcv[opt.ordinary], pch = 19))
title("GCV v.s. log(lambda)")
## natural spline
with(natural$pls, plot(rho, edf, ann = FALSE))
title("edf v.s. log(lambda)")
with(natural$pls, plot(rho, gcv, ann = FALSE))
with(natural$pls, points(rho[opt.natural], gcv[opt.natural], pch = 19))
title("GCV v.s. log(lambda)")
## periodic spline
with(periodic$pls, plot(rho, edf, ann = FALSE))
title("edf v.s. log(lambda)")
with(periodic$pls, plot(rho, gcv, ann = FALSE))
with(periodic$pls, points(rho[opt.periodic], gcv[opt.periodic], pch = 19))
title("GCV v.s. log(lambda)")
par(op)

## inspect fitted splines
yhat.ordinary <- with(ordinary, eqn$B %*% pls$beta)
yhat.natural <- with(natural, eqn$B %*% pls$beta)
yhat.periodic <- with(periodic, eqn$B %*% pls$beta)
op <- par(mfcol = c(1, 3), mar = c(2, 2, 1.5, 0.5))
## ordinary spline
matplot(x, yhat.ordinary, type = "l", lty = 1, ann = FALSE)
title("ordinary")
## natural spline
matplot(x, yhat.natural, type = "l", lty = 1, ann = FALSE)
title("natural")
## periodic spline
matplot(x, yhat.periodic, type = "l", lty = 1, ann = FALSE)
title("periodic")
par(op)
```

```
## pick and plot the optimal fit minimizing GCV score
best.ordinary <- yhat.ordinary[, opt.ordinary]
best.natural <- yhat.natural[, opt.natural]
best.periodic <- yhat.periodic[, opt.periodic]
op <- par(mfcol = c(1, 3), mar = c(2, 2, 1.5, 0.5))
## ordinary spline
plot(x, y, ann = FALSE)
lines(x, gx, lwd = 2, col = 2)
lines(x, best.ordinary, lwd = 2)
title("ordinary")
## natural spline
plot(x, y, ann = FALSE)
lines(x, gx, lwd = 2, col = 2)
lines(x, best.natural, lwd = 2)
title("natural")
## periodic spline
plot(x, y, ann = FALSE)
lines(x, gx, lwd = 2, col = 2)
lines(x, best.periodic, lwd = 2)
title("periodic")
par(op)
```

`MakeGrid` *Make a grid on the domain*

Description

Place equidistant grid points on each knot span to make a grid on the domain, suitable for evaluating B-splines.

Usage

```
MakeGrid(xd, n, rm.dup = FALSE)
```

Arguments

`xd`	domain knot sequence.
`n`	number of equidistant grid points on each knot span.
`rm.dup`	if FALSE, interior knots will appear twice on the grid; if TRUE, they will appear only once.

Details

Denote the domain knot sequence by $s_0, s_1, s_2, \dots, s_k, s_{k+1}$, where $(s_j)_1^k$ are interior knots and $s_0 = a$, $s_{k+1} = b$ are domain endpoints. A knot span is the interval between two successive knots, i.e., $[s_j, s_{j+1}]$.

To make a suitable grid on $[a, b]$ for evaluating B-splines, we can place n equidistant grid points on each knot span, ending up with $n(k + 1)$ grid points in total. Interior knots will show up twice in this grid. To keep only one instance, set `rm.dup = TRUE`.

Value

A vector of grid points.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

Examples

```
require(gps)

## 4 domain knots: two interior knots 0.5 and 1.5 in domain [0, 3]
xd <- c(0, 0.5, 1.5, 3)

## interior knots will appear twice
MakeGrid(xd, 5, rm.dup = FALSE)

## interior knots only appear once
MakeGrid(xd, 5, rm.dup = TRUE)
```

Ospline *Build derivative penalty matrix or its sparse "root"*

Description

Compute order-m derivative penalty matrix $\boldsymbol{S}$ or its sparse "root" $\boldsymbol{K}$ such that $\boldsymbol{K'K} = \boldsymbol{S}$.

Evaluate derivative penalty

Evaluate derivative penalty $\int_a^b f^{(m)}(x)^2 \mathrm{d}x = \boldsymbol{\beta'S\beta}$, where $\boldsymbol{S}$ is the order-m derivative penalty matrix and $\boldsymbol{\beta}$ is the vector of B-spline coefficients for $f(x)$.

Usage

```
SandBar(xt, d, m)

SparseS(xt, d, m, root = FALSE)

btSb(b, xt, d, m)
```

Arguments

`xt`	full knot sequence for ordinary B-splines (`length(xt) >= 2 * d` required).
`d`	B-spline order ($d \geq 2$ required).
`m`	order of the derivative penalty ($0 \leq m \leq d - 1$ required).
`root`	if TRUE, return the sparse "root" of the derivative penalty matrix.
`b`	B-spline coefficients (`length(b) == length(xt) -d` required).

Details

Build $\bar{\boldsymbol{S}}$ in the sandwich formula $\boldsymbol{S} = \boldsymbol{D}'\bar{\boldsymbol{S}}\boldsymbol{D}$.

Given that $\boldsymbol{S}$ can be obtained from SparseS, it is straightforward to compute the quadratic form $\boldsymbol{\beta}'\boldsymbol{S}\boldsymbol{\beta}$. However, the function takes an alternative approach.

Denote the domain knot sequence by $s_0, s_1, s_2, \ldots, s_k, s_{k+1}$, where $(s_j)_1^k$ are interior knots and $s_0 = a$, $s_{k+1} = b$ are domain endpoints. The derivative penalty aggregates roughness penalties on all knot spans:

$$\int_a^b f^{(m)}(x)^2 \mathrm{d}x = \sum_{j=0}^{k} \int_{s_j}^{s_{j+1}} f^{(m)}(x)^2 \mathrm{d}x.$$

The function calculates and returns those additive components in a vector.

Value

`SandBar` returns a sparse matrice of "dsCMatrix" (when $m < d - 1$) or "ddiMatrix" (when $m = d - 1$) class.

`SparseS` returns a sparse matrice giving the order-m derivative penalty matrix (of "dsCMatrix" class) or its sparse "root" (of "dgCMatrix" class).

`btSb` returns a vector of evaluated derivative penalties at all knot spans.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## example with ordinary B-splines
xt <- c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4)

SandBar(xt, d = 4, m = 2)

SparseS(xt, d = 4, m = 2)

b <- rnorm(6)
btSb(b, xt, d = 4, m = 2)
```

`periodic` *Design matrix and general difference matrices for periodic B-splines*

Description

For order-d periodic B-splines, `pbsDesign` evaluates B-splines or their derivatives at given x-values, and `SparsePD` computes general difference matrices of order 1 to $d - 1$.

Usage

```
pbsDesign(x, xd, d, nDeriv = 0, sparse = FALSE, wrap = TRUE)

SparsePD(xd, d, wrap = TRUE)
```

Arguments

`x`	x-values where periodic B-splines are to be evaluated.
`xd`	domain knot sequence for periodic B-splines (`length(xd)` >= `d + 1` required).
`d`	B-spline order ($d \geq 2$ required).
`nDeriv`	derivative order.
`sparse`	if TRUE, create a sparse design matrix of "dgCMatrix" class.
`wrap`	TRUE to

Details

These functions perform type-2 construction, by transforming design matrix and general difference matrices for ordinary B-splines to satisfy periodic boundary constraints (see Details). By contrast, `pbsDesign` and `SparsePD` in **gps** perform type-1 construction by basis wrapping.

A spline $f(x)$ on domain $[a, b]$ can be constructed to satisfy periodic boundary constraints, that is, $f^{(q)}(a) = f^{(q)}(b)$, q = 0, 1, ..., degree - 1. These are actually linear equality constraints

Unlike ordinary B-splines, period B-splines do not require explicit auxiliary boundary knots for their construction. The magic is that auxiliary boundary knots will be automatically positioned by periodic extension of interior knots.

Denote the domain knot sequence by $s_0, s_1, s_2, \ldots, s_k, s_{k+1}$, where $(s_j)_1^k$ are interior knots and $s_0 = a$, $s_{k+1} = b$ are domain endpoints. For order-d B-splines, we replicate the first $d - 1$ interior knots (after adding $b - a$) to the right of $[a, b]$ for an augmented set of $K = k + d + 1$ knots, which spawns $p = K - d = k + 1$ ordinary B-splines. It turns out that periodic B-splines can be obtained by wrapping segments of those ordinary B-splines that stretch beyond $[a, b]$ to the start of the domain (a demo is offered by `DemoPBS`).

Note that we must have at least $d - 1$ interior knots to do such periodic extension. This means that $d + 1$ domain knots are required at a minimum for construction of periodic B-splines.

Value

`pbsDesign` returns a design matrix with `length(x)` rows and `length(xd) -1` columns. `SparsePD` returns a list of sparse matrices (of "dgCMatrix" class), giving general difference matrices of order 1 to $d - 1$.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## 5 domain knots: three interior knots 0.5, 1.5 and 1.8 in domain [0, 3]
xd <- c(0, 0.5, 1.5, 1.8, 3)

## make a grid
x <- MakeGrid(xd, n = 10)

## construct periodic cubic B-splines
PB1 <- pbsDesign(x, xd, d = 4, wrap = TRUE)
PB2 <- pbsDesign(x, xd, d = 4, wrap = FALSE)

## construct general difference matrices of order 1 to 3
SparsePD(xd, d = 4, wrap = TRUE)
SparsePD(xd, d = 4, wrap = FALSE)
```

PlaceKnots *Automatically place knots according to data*

Description

Place knots for ordinary B-splines or periodic B-splines using automatic strategies.

Usage

```
PlaceKnots(x, d, k, domain = NULL, uniform = FALSE, periodic = FALSE)
```

Arguments

`x`	observed x-values.
`d`	B-spline order.
`k`	number of interior knots.
`domain`	a vector of two values giving domain interval $[a, b]$. Will use `min(x)` and `max(x)` if not specified.
`uniform`	TRUE to place equidistant knots; FALSE to place quantile knots with clamped boundary knots.
`periodic`	if TRUE, return domain knot sequence that is sufficient for constructing periodic B-splines (see `pbsDesign`); if FALSE, return full knot sequence that is required for constructing ordinary B-splines.

Value

A vector of $K = k + 2d$ knots for ordinary B-splines, or $k + 2$ knots for periodic B-splines.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

Examples

```
require(gps)

x <- rnorm(50)

## uniform knots for uniform cubic B-splines
xt1 <- PlaceKnots(x, d = 4, k = 5, uniform = TRUE)
B1 <- splines::splineDesign(xt1, x, ord = 4)

## clamped quantile knots for clamped non-uniform cubic B-splines
xt2 <- PlaceKnots(x, d = 4, k = 5, uniform = FALSE)
B2 <- splines::splineDesign(xt2, x, ord = 4)

## uniform knots for uniform periodic cubic B-splines
xd1 <- PlaceKnots(x, d = 4, k = 5, uniform = TRUE, periodic = TRUE)
PB1 <- pbsDesign(x, xd1, d = 4)

## quantile knots for non-uniform periodic cubic B-splines
xd2 <- PlaceKnots(x, d = 4, k = 5, uniform = FALSE, periodic = TRUE)
PB2 <- pbsDesign(x, xd2, d = 4)
```

`prior` *Sample B-spline coefficients from their prior distribution*

Description

In the Bayesian view, the L_2 penalty $\|\boldsymbol{D}\boldsymbol{\beta}\|^2$ for B-spline coefficients $\boldsymbol{\beta}$ is an improper Gaussian prior: $\boldsymbol{\beta} \sim \mathrm{N}(\boldsymbol{0},\ (\boldsymbol{D'}\boldsymbol{D})^-)$, where the inverse denotes the Moore-Penrose generalized inverse.

Usage

```
PriorCoef(n, D)

MPinverse(D, diag.only = FALSE)
```

Arguments

`n`	number of samples to draw.
`D`	the matrix
`diag.only`	if TRUE, only diagonal elements

Value

A vector of coefficients when `n = 1`, or a coefficient matrix with `n` columns when `n > 1`.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## 11 domain knots at equal quantiles of Beta(3, 3) distribution
xd <- qbeta(seq.int(0, 1, by = 0.1), 3, 3)
## full knots (with clamped boundary knots) for constructing cubic B-splines
xt <- c(0, 0, 0, xd, 1, 1, 1)

## 2nd order general difference matrix
D <- SparseD(xt, d = 4)[[2]]
## sparse "root" of the 2nd order derivative penalty matrix
K <- SparseS(xt, d = 4, m = 2, root = TRUE)

## sample B-spline coefficients from their prior distribution
b1 <- PriorCoef(n = 5, D)
b2 <- PriorCoef(n = 5, K)
op <- par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5), oma = c(0, 0, 1, 0))
## prior B-spline coefficients for general difference penalty
matplot(b1, type = "l", lty = 1, ann = FALSE)
title("general difference penalty")
## prior B-spline coefficients for derivative penalty
matplot(b2, type = "l", lty = 1, ann = FALSE)
title("derivative penalty")
title("random B-spline coefficients from their prior", outer = TRUE)
par(op)

## plot the corresponding cubic splines for these B-spline coefficients
x <- MakeGrid(xd, n = 11)
B <- splines::splineDesign(xt, x, ord = 4, sparse = TRUE)
y1 <- B %*% b1
y2 <- B %*% b2
op <- par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5), oma = c(0, 0, 1, 0))
matplot(x, y1, type = "l", lty = 1, ann = FALSE)
title("general difference penalty")
matplot(x, y2, type = "l", lty = 1, ann = FALSE)
title("derivative penalty")
title("random cubic splines with prior B-spline coefficients", outer = TRUE)
par(op)
```

`SparseD` *General difference matrices for ordinary B-splines*

Description

Compute general difference matrices of order 1 to $d-1$ for ordinary B-splines of order d. Compute general differences of $f(x)$'s B-spline coefficients, which are $f^{(m)}(x)$'s coefficients. Compute general differences of B-spline coefficients. Find null space of a general difference matrix. Compute a basis matrix for the null space of a general difference matrix.

Usage

```
SparseD(xt, d)

DiffCoef(b, xt, d, m)

NullD(xt, d, m)
```

Arguments

`xt`	full knot sequence for ordinary B-splines (`length(xt) >= 2 * d` required).
`d`	B-spline order ($d \geq 2$ required).
`m`	order of the general difference matrix ($1 \leq m \leq d - 1$ required).
`b`	B-spline coefficients (`length(b) == length(xt) -d` required).

Value

`SparseD` returns a list of sparse matrices (of "dgCMatrix" class), giving general difference matrices of order 1 to $d - 1$.

`DiffCoef` returns a vector of coefficients.

`NullD` returns a matrix of m columns.

Author(s)

Zheyuan Li <`zheyuan.li@bath.edu`>

Examples

```
require(gps)

## example with ordinary B-splines
xt <- c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4)

SparseD(xt, d = 4)

b <- rnorm(6)
DiffCoef(b, xt, d = 4, m = 2)

NullD(xt, d = 4, m = 2)
```

Index