Package 'gps.mgcv'

April 13, 2022

Version 1.0

Date 2022-04-13

Title General P-Splines for Package 'mgcv'

Author Zheyuan Li [aut, cre] (<https://orcid.org/0000-0002-7434-5947>),
Jiguo Cao [fnd] (<https://orcid.org/0000-0001-7417-6330>),
Ahmed Elhakeem [dtc] (<https://orcid.org/0000-0001-7637-6360>)

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

Depends R (>= 4.0.0)

Imports gps (>= 1.1), mgcv (>= 1.8-40), stats, Matrix, splines,
graphics, parallel

Description The package constructs and predicts the general P-splines of Li and Cao (2022) <arXiv:2201.06808> in mgcv by defining a new 'gps' smooth class (see ?gps.smooth). A general P-spline f(x) is specified as s(x, bs = 'gps', ...) in a formula and estimated using mgcv's model fitting functions, namely gam (generalized additive models, or GAMs), bam (GAMs for big data) and gamm (GAMs as mixed-effect models). General P-splines are state-of-the-art penalized B-splines. Unlike the standard P-splines of Eilers and Marx (1996) <doi:10.1214/ss/1038425655> that only make sense for uniform B-splines on equidistant knots, they are properly defined for non-uniform B-splines on irregularly spaced knots, thanks to their powerful general difference penalty that accounts for uneven knot spacing. The package also contains functions for fitting and benchmarking different penalized B-splines (see ?Fit4BS and ?SimStudy) and a case study of smoothing Bone Mineral Content longitudinal data (see ?BMC).

License GPL-3

NeedsCompilation no

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

R topics documented:

BMC *A case study of smoothing Bone Mineral Content longitudinal data*

Description

The synthetic dataset contains BMC measured at growing ages for white ethnic subjects, with 932 data from 112 males and 1113 data from 127 females. An additive mixed model based on general P-splines is ideal for analyzing this dataset, able to produce smooth estimations of population and subject BMC trajectories.

Usage

```
synBMC(gender)

FitBMC(gender, p, gps = TRUE, subset = NULL)

cvBMC(gender, gps = TRUE, n.sim = 100, nc = 1)
```

Arguments

gender	the gender group ("male" or "female") to study.
p	number of B-splines to represent $f(x)$ and $f_j(x)$ (see Details).
gps	TRUE to construct $f(x)$ and $f_j(x)$ as general P-splines; FALSE to construct them as standard P-splines.
subset	an optional vector of integers that specifies a data subset to be used for model fitting (for example, it is used when cvBMC calls FitBMC on a training set during cross-validation).
n.sim	number of simulations in cross-validation.
nc	number of CPU cores to use for parallel simulations.

Details

Data Description:

For either gender group, the synthetic BMC dataset is a dataframe with four variables:

- id, subject's ID ("M001" to "M112" for males and "F001" to "F127" for females);
- age, subject's age when a measurement was taken;
- bmc, BMC measurements (in grams);
- log.bmc, log-transformed BMC measurements.

It is a synthetic dataset that highly resembles the real one from The Saskatchewan Pediatric Bone Mineral Accrual Study (PBMAS).

Additive Mixed Model:

Denote age, BMC and log(BMC) by x, y and z, respectively. The model for either gender group is:

$$z_{j,x} = f(x) + f_j(x) + e_{j,x},$$

where

- $f(x)$ (a fixed-effect spline) is the population log(BMC) trajectory;
- $f_j(x)$ (a random-effect spline) is subject j's log(BMC) deviation from $f(x)$;
- $e_{j,x}$ is an i.i.d. Gaussian measurement error.

`FitBMC` sets up $f(x)$ and $f_j(x)$ as standard or general P-splines then fits this model with `gamm`.

How Synthetic Data are Simulated:

Let n be the number of subjects in either gender group. Let vectors $\boldsymbol{x_j}$, $\boldsymbol{y_j}$ and $\boldsymbol{z_j}$ be the data for subject j. Let $\delta = \min\{\Delta\boldsymbol{x_1}, \Delta\boldsymbol{x_2}, \ldots, \Delta\boldsymbol{x_n}\}$ be the minimum step size in age values. The synthetic data are simulated as follows.

1. Fit the additive mixed model above to the real data $(\boldsymbol{z_j}, \boldsymbol{x_j}, j)_1^n$;
2. Randomize the association between $\boldsymbol{x_j}$ and j to obtain $(\boldsymbol{x_j}, r_j)_1^n$, where $r_1, r_2, \ldots, r_n$ is a random shuffle of $1, 2, \ldots, n$;
3. Add a small amount of noise ("jitter") to obtain $\tilde{\boldsymbol{x}}_{\boldsymbol{j}} = \boldsymbol{x_j} + \boldsymbol{\delta_j}$, where $\boldsymbol{\delta_j}$ is vector of samples from uniform distribution on $[-\delta/5, -\delta/10] \cup [\delta/10, \delta/5]$;
4. Predict $\hat{\boldsymbol{z}}_{\boldsymbol{j}}$ on $(\tilde{\boldsymbol{x}}_{\boldsymbol{j}}, r_j)_1^n$ using the fitted model and draw $\tilde{\boldsymbol{z}}_{\boldsymbol{j}} \sim \mathrm{N}(\hat{\boldsymbol{z}}_{\boldsymbol{j}}, \boldsymbol{I}\hat{\sigma}_e^2)$, where $\hat{\sigma}_e^2$ is the estimated variance of measurement error and $\boldsymbol{I}$ is an identity matrix;
5. Back transform $\tilde{\boldsymbol{y}}_{\boldsymbol{j}} = \exp(\tilde{\boldsymbol{z}}_{\boldsymbol{j}})$ and compose the synthetic dataset as $(\tilde{\boldsymbol{y}}_{\boldsymbol{j}}, \tilde{\boldsymbol{x}}_{\boldsymbol{j}}, r_j)_1^n$.

The synthetic dataset preserves two aspects of the real dataset.

- Distribution of age values is almost unchanged;
- The number of data per subject is unchanged.

The fitted model has an R-squared as high as 99.97% for males and 99.92% for females. Thus, statistical modeling on two datasets does not end up with much difference.

Value

`synBMC` returns a data frame of four variables.

`FitBMC` returns a fitted "gamm" model, which is a large list with an "lme" component and a (much more useful) "gam" component.

`cvBMC` returns a matrix of 3 columns. Column 1 gives the mean cross-validation error (averaged over `n.sim` simulations) for $p = 5, 6, \ldots, 20$. Columns 2 and 3 give two-standard-error lower and upper bounds for this mean.

Note

We do not hold authentic BMC data. The synthetic version was generated by Ahmed Elhakeem (using our idea and code, though), who holds a copy of the authentic version for his research (see References) collaborated with PBMAS's program director, Professor Adam DG Baxter-Jones. The synthetic dataset is also shared by Elhakeem at https://raw.githubusercontent.com/aelhak/nltmr/main/synth_cohort.csv. Researches interested in the authentic BMC data should contact Professor Baxter-Jones (baxter.jones@usask.ca) in the first place.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

References

Bailey, D. A. (1997). The Saskatchewan Pediatric Bone Mineral Accrual Study: Bone mineral acquisition during the growing years. *International Journal of Sports Medicine*, 18:s191-s194.

Baxter-Jones, A. D., Faulkner, R. A., Forwood, M. R., Mirwald, R. L., and Bailey, D. A. (2011). Bone mineral accrual from 8 to 30 years of age: An estimation of peak bone mass. *Journal of Bone and Mineral Research*, 26(8):1729-1739.

Elhakeem, A., Hughes, R., Tilling, K., Cousminer, D., Jackowski, S., Cole, T., Kwong, A., Li, Z., Grant, S., Baxter-Jones, A., Zemel, B., and Lawlor, D. (2022). Using linear and natural cubic splines, sitar, and latent trajectory models to characterise nonlinear longitudinal growth trajectories in cohort studies. *BMC Medical Research Methodology*, 22(68).

Li, Z. and Cao, J. (2022). General P-Splines for Non-Uniform B-Splines.

Examples

```
require(mgcv)
require(gps.mgcv)

## get synthetic BMC data
MBMC <- synBMC(gender = "male")
FBMC <- synBMC(gender = "female")

## number of subjects
NM <- nlevels(MBMC$id)
NF <- nlevels(FBMC$id)

## split data by subject
MBMC.id <- split(MBMC[-1], MBMC$id)
FBMC.id <- split(FBMC[-1], FBMC$id)

## visualize data
op <- par(mfrow = c(2, 3), mar = c(2, 2, 1.5, 0.5))
## male: bmc ~ age
with(MBMC, plot(age, bmc, type = "n", ann = FALSE))
for (i in 1:NM) with(MBMC.id[[i]], lines(age, bmc, col = i))
title("male: bmc ~ age")
## male: log(bmc) ~ age
with(MBMC, plot(age, log.bmc, type = "n", ann = FALSE))
for (i in 1:NM) with(MBMC.id[[i]], lines(age, log.bmc, col = i))
title("male: log(bmc) ~ age")
## male: hist(age)
hist(MBMC$age, breaks = 15, ann = FALSE)
title("male: age distribution")
## female: bmc ~ age
with(FBMC, plot(age, bmc, type = "n", ann = FALSE))
for (i in 1:NF) with(FBMC.id[[i]], lines(age, bmc, col = i))
title("female: bmc ~ age")
```

```
## female: log(bmc) ~ age
with(FBMC, plot(age, log.bmc, type = "n", ann = FALSE))
for (i in 1:NF) with(FBMC.id[[i]], lines(age, log.bmc, col = i))
title("female: log(bmc) ~ age")
## female: hist(age)
hist(FBMC$age, breaks = 15, ann = FALSE)
title("female: age distribution")
par(op)

## Not run:

## fit models for male group
gps.10 <- FitBMC(gender = "male", p = 10, gps = TRUE)$gam
sps.10 <- FitBMC(gender = "male", p = 10, gps = FALSE)$gam

## newdata
age <- seq.int(min(MBMC$age), max(MBMC$age), length.out = 100)
newdat1 <- data.frame(age = rep.int(age, NM),
                      id = rep(levels(MBMC$id), each = length(age)))
newdat2 <- data.frame(age = age, id = levels(MBMC$id)[1])

## predict all subject BMC trajectories
sub.gps.10 <- matrix(exp(predict(gps.10, newdat1)), nrow = length(age))
sub.sps.10 <- matrix(exp(predict(sps.10, newdat1)), nrow = length(age))

## predict population BMC trajectory
pop.gps.10 <- predict(gps.10, newdat2, type = "terms", terms = "s(age)")
pop.sps.10 <- predict(sps.10, newdat2, type = "terms", terms = "s(age)")
pop.gps.10 <- exp(as.numeric(pop.gps.10) + attr(pop.gps.10, "constant"))
pop.sps.10 <- exp(as.numeric(pop.sps.10) + attr(pop.sps.10, "constant"))

## extract knots
knots.gps.10 <- gps.10$smooth[[1]]$knots
knots.sps.10 <- sps.10$smooth[[1]]$knots

## plot data and fitted trajectories
op <- par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
## a common ylim for the next 2 plots
ylim <- range(sub.gps.10[, c(41, 66)], sub.sps.10[, c(41, 66)])
## standard P-spline, population trajectory and subject 41, 66
plot(age, pop.sps.10, type = "l", lwd = 2, ylim = ylim, ann = FALSE)
with(MBMC.id[[41]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
with(MBMC.id[[66]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
abline(v = knots.sps.10, lty = 2, col = 8)
lines(age, sub.sps.10[, 41], lwd = 2, lty = 2)
lines(age, sub.sps.10[, 66], lwd = 2, lty = 2)
title("standard P-spline")
## general P-spline, population trajectory and subject 41, 66
plot(age, pop.gps.10, type = "l", lwd = 2, ylim = ylim, ann = FALSE)
with(MBMC.id[[41]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
with(MBMC.id[[66]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
abline(v = knots.gps.10, lty = 2, col = 8)
lines(age, sub.gps.10[, 41], lwd = 2, lty = 2)
```

```
lines(age, sub.gps.10[, 66], lwd = 2, lty = 2)
title("general P-spline")
par(op)

## End(Not run)
```

`Fit4BS` *Experiment 4 variants of penalized B-splines*

Description

Fit 4 variants of penalized B-splines to noisy (x, y) data simulated from $g(x)$ and compare their MSE performance under repeated simulations.

Usage

```
Fit4BS(x, y, k, knots = NULL, domain = NULL, periodic = FALSE,
       select = c("os", "sps", "nps", "gps"))

MSE4BS(x, g, k, knots = NULL, domain = NULL, periodic = FALSE,
       n2s.ratio = 0.2, n.sim = 100, nc = 1,
       select = c("os", "sps", "nps", "gps"))
```

Arguments

`x, y`	(x, y) data.
`g`	the true generative function $g(x)$ from which data are simulated.
`k`	number of interior knots to place. This gives $k + 4$ ordinary cubic B-splines or $k + 1$ periodic cubic B-splines.
`knots`	a sequence of interior knots on which non-uniform B-splines are constructed for "os", "nps" and "gps" (see Details). Will be automatically placed at equal quantiles of x-values if not specified. If specified, must match `k`.
`domain`	a vector of two values giving domain interval $[a, b]$. Will use `min(x)` and `max(x)` if not specified.
`periodic`	TRUE to fit a periodic spline; FALSE to fit an ordinary spline.
`n2s.ratio`	a noise-to-signal ratio that is used when simulating noisy y-values from $g(x)$.
`n.sim`	number of simulations.
`nc`	number of CPU cores to use for parallel simulations.
`select`	a character vector giving names (see Details) of the variants to fit.

Details

The 4 variants of penalized B-splines are:

- O-spline ("os"): non-uniform B-splines + derivative penalty;
- standard P-spline ("sps"): uniform B-splines + standard difference penalty;
- naive P-spline ("nps"): non-uniform B-splines + standard difference penalty;
- general P-spline ("gps"): non-uniform B-splines + general difference penalty.

They are constructed with cubic B-splines on k interior knots, penalized with a 2nd order derivative or difference penalty. By default, quantile knots are placed automatically for "os", "nps" and "gps". But this can be overridden by user-specified interior knots through argument `knots`.

Value

A list of 4 fitted "gam" models, with components "os", "sps", "nps" and "gps". A component may be empty (with value `NULL`) if it is excluded from `select`.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

References

Li, Z. and Cao, J. (2022). General P-Splines for Non-Uniform B-Splines.

Examples

```
## Not run:

require(mgcv)
require(gps.mgcv)

## a U-shaped curve
g <- function (x) (1 / 8) * abs(x) ^ 3

## number of data
n <- 400
## noise-to-signal ratio
n2s.ratio <- 0.4
## number of interior knots
k <- 40

## (x, y) data
x <- qnorm(seq.int(pnorm(-3), pnorm(3), length.out = n))
gx <- g(x)
y <- rnorm(length(gx), gx, n2s.ratio * sd(gx))

## fit 4 variants of penalized B-splines
fit <- Fit4BS(x, y, k)

## plot a fitted penalized B-spline against observed (x, y) data and true function g(x)
```

```
## gamfit: a fitted GAM model, should only have a single s(x) term of B-spline class
## x, y: observed (x, y) data
## g: the true function g(x)
PlotXYFit <- function (gamfit, x, y, g) {
  ## g must be a function
  if (!is.function(g)) stop("g is not a function!")
  ## plot (x, y) data
  plot(x, y, col = 8, ann = FALSE)
  ## observed x-values may not be evenly spaced
  ## to plot g(x), we'd better use evenly spaced x-values
  nx <- length(x)
  xg <- seq.int(min(x), max(x), length.out = nx)
  lines(xg, g(xg), col = 2, lwd = 2)
  ## show knot location
  abline(v = gamfit$smooth[[1]]$knots, lty = 2, col = 8)
  ## plot estimated f(x) at xg
  yh <- predict(gamfit, newdata = data.frame(x = xg))
  lines(xg, yh, lwd = 2)
}

## compare 4 variants of penalized B-splines
op <- par(no.readonly = TRUE)
layout(matrix(c(1, 2, 3, 4, 5, 5), nrow = 2))
par(mar = c(2, 2, 1.5, 0.5))
## O-spline
PlotXYFit(fit$os, x, y, g)
title("O-spline")
## standard P-spline
PlotXYFit(fit$sps, x, y, g)
title("standard P-spline")
## naive P-spline
PlotXYFit(fit$nps, x, y, g)
title("naive P-spline")
## general P-spline
PlotXYFit(fit$gps, x, y, g)
title("general P-spline")
## boxplot of MSE
mse <- MSE4BS(x, g, k = k, n2s.ratio = n2s.ratio, n.sim = 100, nc = 1)
boxplot(mse, main = "MSE (100 simulations)")
par(op)

## End(Not run)
```

---

gps.smooth *General P-splines for package 'mgcv'*

Description

Construct and predict a general P-spline in the framework of generalized additive models (GAMs) using **mgcv**.

Usage

```
## S3 method for class 'gps.smooth.spec'
smooth.construct(object, data, knots)

## S3 method for class 'gps.smooth'
Predict.matrix(object, data)
```

Arguments

`object`	a smooth term specification list generated by `s(x,bs = "gps",...)`.
`data`	a named list containing the data (including any "by" variable) required by this term, whose names should match `object$term` (and `object$by`).
`knots`	a named list containing any **interior knots** supplied for constructing B-splines, whose names should match `object$term`. Can be `NULL` for automatic knot placement.

Details

Simple Specification:

A general P-spline $f(x)$ with p order-d B-splines and an order-m general difference matrix is specified as `s(x,bs = 'gps',k = p,m = c(d -1,m))` in a formula, fed to **mgcv**'s model fitting functions, namely `gam` (generalized additive models), `bam` (generalized additive models for large datasets) and `gamm` (generalized additive mixed models). A simpler specification as `s(x,bs = 'gps',k = p)` assumes $d = 4$ and $m = 2$, i.e., a cubic spline with a 2nd order general difference penalty.

Note that the math notations p, d, k and m should not be confused with function arguments `k` and `m`. The latter are set by **mgcv**'s standard. Also, argument `m` expects B-splines to be specified by its degree $d - 1$, not its order d.

Multiple Penalties:

Usually there is only one wiggliness penalty. However, we can set up multiple penalties (each with an independent smoothing parameter). For example, a general P-spline with 20 cubic B-splines and three difference penalties of order 3 to 1 is specified as `s(x,bs = 'gps',k = 20,m = c(3,3,2,1))`. This is as same as saying that its 1st to 3rd derivatives are penalized.

Additional Arguments:

By **mgcv**'s standard, `s()` has an argument `xt` that accepts a list of extra arguments. You can use this argument to gain more control over the construction of a "gps" smooth.

1. By default, the domain of the spline is chosen to be `[min(x),max(x)]`. You can specify a wider interval $[a, b]$ by adding `xt = list(domain = [a,b])` into `s()`;
2. To use derivative penalty instead of general difference penalty, i.e., to specify an O-spline, append `derivative = TRUE` to the `xt` list;
3. To construct a general P-spline with periodic boundary constraints, i.e., $f^{(q)}(a) = f^{(q)}(b)$, q = 0, 1, ..., degree - 1, append `periodic = TRUE` to the `xt` list.

So for example, an O-spline with 20 cubic B-splines on domain $[-5, 5]$ and a 1st order penalty is specified as `s(x,bs = 'gps',k = 20,m = c(3,1),xt = list(domain = c(-5,5),derivative = TRUE))`.

Knot Placement:

By default, $K = p + d$ quantile knots with clamped boundary knots are placed on $[a, b]$. To override such automatic knot placement, add `knots = list(x = xs)` into `gam()`, `bam()` or `gamm()` (not into `s()`!), where `xs` is an ascending sequence of $k = p - d$ interior knots in (a, b).

Tensor Product Smooth:

A "gps" smooth can be a margin of a tensor product smooth, constructed with `te()`, `ti()` or `t2()`. For example, a bivariate tensor product spline $f(x_1, x_2) = f_1(x_1) \otimes f_2(x_2)$, where $f_1(x_1)$ is a "gps" with 10 cubic B-splines and a 1st order difference penalty, and $f_2(x_2)$ is a "gps" with 20 cubic B-splines and a 2nd order difference penalty, subject to periodic boundary constraints, can be specified as `te(x1,x2,d = c(1,1),bs = c('gps','gps'),k = c(10,20),m = list(x1 = c(3,1),x2 = c(3,2)),xt = list(x1 = NULL,x2 = list(periodic = TRUE)))`.

For `te()` and `ti()`, a "gps" margin often results in a warning about "unstable reparametrization". To get rid of such warning, append `np = FALSE` to `te()` or `ti()`. The "t2" smooth does not suffer from such issue.

Note that it is not possible to use multiple penalties for a margin.

"By" Variable:

Both `s()` and tensor product smooth (`te()`, `ti()` or `t2()`) have a `by` argument, that can accept a numerical variable or a factor variable.

Factor-Smooth Interaction (efficient for `gamm`):

A "gps" smooth can also be used in an "fs" smooth. For example, let `fac` be a factor variable, then `s(x,fac,bs = 'fs',k = 10,xt = list(bs = 'gps'),m = c(3,1))` sets up a general P-spline with 10 cubic B-splines and a 1st order difference penalty for each level of `fac`, which is as same as specifying a random-effect trajectory for each level. All trajectories share a common smoothing parameter. With a 1st order penalty, this model term will be shrunk toward a random intercept as smoothing parameters goes to infinity. If a 2nd order penalty is used, a random slope will also be included, which is often not desirable.

Note that `xt` argument in this case is processed by "fs" smooth, not "gps" smooth. At the moment "fs" smooth ignores other variables provided through the `xt` list, so that it is not possible to successfully pass in additional arguments for the "gps" smooth.

Also note that multiple penalties are not allowed by a "fs" smooth.

Value

An object of class `"gps.smooth"`. It augments the smooth term specification list, by adding for example, B-spline design matrix, list of penalty matrices, and information like domain and knots. See examples below.

Note

Mathematically speaking, general P-spline comprises standard P-spline as a special case. However, a "gps" smooth always constructs B-splines with clamped boundary knots, it is thus impossible to produce a standard P-spline with "gps" smooth class (even if all interior knots are equidistant). To specify a standard P-spline, you should use the "ps" class from **mgcv**.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

References

Li, Z. and Cao, J. (2022). General P-Splines for Non-Uniform B-Splines.

Examples

```
require(mgcv)
require(gps.mgcv)

######################################################################
## use general P-spline for smoothing (in the framework of GAMs) ##
######################################################################

## set this to TRUE for a periodic spline demo
cyclic <- FALSE

## 250 data from a random cubic spline on [0, 1]
spl <- RandomSpl(n = 250, degree = 3, periodic = cyclic, plot = FALSE)
## noisy (x, y) observations
x <- spl$x
y <- rnorm(250, spl$y, sd = 0.2 * sd(spl$y))
## fit a general P-spline using 10 cubic B-splines and a 2nd order penalty
## increase k if the fit is not adequate (it can happen for some random example)
fit <- gam(y ~ s(x, bs = "gps", k = 10, xt = list(periodic = cyclic)))
## visualize fit
plot(x, y, col = 8)
lines(spl$xg, spl$yg, col = 2, lwd = 2)
yh <- predict(fit, newdata = data.frame(x = spl$xg))
lines(spl$xg, yh, lwd = 2)

#####################################################
## direct use of smooth.construct.gps.smooth.spec ##
#####################################################

## this is useful when you want to use the constructor outside mgcv for your own purpose

## provide data in a list (or data frame)
xdat <- list(x = rnorm(50))

## simplest specification, 10 cubic B-splines with a 2nd order penalty
sterm <- s(x, bs = "gps", k = 10)
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = NULL)

## use argument 'm' to specify, say 10 quadratic B-splines with a 1st order penalty
sterm <- s(x, bs = "gps", k = 10, m = c(2, 1))
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = NULL)

## multiple penalties, say 10 cubic B-splines with penalties of order 1 to 3
sterm <- s(x, bs = "gps", k = 10, m = c(3, 1, 2, 3))
```

```
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = NULL)

## use argument 'xt' to specify a wider domain than [min(x), max(x)]
ab <- extendrange(xdat$x, f = 0.05)
sterm <- s(x, bs = "gps", k = 10, xt = list(domain = ab))
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = NULL)

## use argument 'xt' to specify an O-spline
sterm <- s(x, bs = "gps", k = 10, xt = list(derivative = TRUE))
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = NULL)

## use argument 'xt' to specify a general P-spline with periodic boundary constraints
sterm <- s(x, bs = "gps", k = 10, xt = list(periodic = TRUE))
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = NULL)

## specify your own knots
## a reminder: provide interior knots!!
## interior knots are strictly ascending **inside** the range of x-values
## number of interior knots = argument.k - degree - 1
## the following code constructs 10 cubic B-splines on equidistant interior knots
## (note that this is not a standard P-spline because of clamped boundary knots)
sterm <- s(x, bs = "gps", k = 10)
xk <- seq.int(min(xdat$x), max(xdat$x), length.out = 8)[-c(1, 8)]
gps.object <- smooth.construct.gps.smooth.spec(sterm, data = xdat, knots = list(x = xk))

## in all examples above, you can inspect the construction result using str()
str(gps.object)
## but you are probably more interested in the following quantities
## B-spline design matrix
gps.object$X
## list of penalty matrices
gps.object$S
## spline domain
gps.object$xt$domain
## interior knots
gps.object$interior.knots
## full knot sequence (boundary knots are always clamped)
gps.object$knots
## O-spline instead of general P-spline?
gps.object$xt$derivative
## with periodic boundary constraints?
gps.object$xt$periodic
## note that in the periodic case, number of B-splines = argument.k - degree
## that is, for cubic splines, with k = 10 you see 7 not 10 columns in X
## the reason is that the 3 constraints are absorbed into X
## similarly, penalty matrix is 7 x 7 not 10 x 10
```

RandomSpl *A random spline on [0, 1].*

Description

Generate a random spline $g(x)$ on [0, 1] and sample its values at n unevenly spaced x-values.

Usage

```
RandomSpl(n, degree, periodic = FALSE, plot = TRUE)
```

Arguments

n	number of unevenly spaced x-values.
degree	polynomial degree of the spline.
periodic	TRUE to generate a periodic spline; FALSE to generate an ordinary spline.
plot	TRUE to plot the generated random spline.

Details

Let d be the polynomial order (degree + 1). A random spline is generated as a linear combination of $3d$ uniform B-splines, whose coefficients are a realization of a first order random walk. In case $g(x)$ needs be periodic, these B-spline coefficients are further transformed to satisfy the periodic constraints, i.e., $g^{(m)}(0) = g^{(m)}(1)$, m = 0, 1, ..., degree - 1.

Value

A list of the following components:

- x, n unevenly spaced x-values drawn from a "tent" distribution, whose density peaks at $g(x)$'s local extrema;
- y, $g(x)$ evaluated at x;
- xg, 101 equidistant x-values (i.e., stepsize = 0.01);
- yg, $g(x)$ evaluated at xg;

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

References

Li, Z. and Cao, J. (2022). General P-Splines for Non-Uniform B-Splines.

Examples

```
require(gps.mgcv)

op <- par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
spl.ordinary <- RandomSpl(400, degree = 3)
title("a random cubic spline")
spl.periodic <- RandomSpl(400, degree = 3, periodic = TRUE)
title("a random periodic cubic spline")
par(op)
```

SimStudy *A simulation study on penalized B-splines' MSE performance*

Description

Compare MSE performance of O-spline ("os"), standard P-spline ("sps") and general P-spline ("gps") via a simulation study with random curves generated by RandomSpl.

Usage

```
SimStudy(n, degree, periodic = FALSE, n2s.ratio = 0.2, n.sim = 100, nc = 1)
```

Arguments

n	number of unevenly spaced x-values. This argument is passed to RandomSpl.
degree	polynomial degree of the spline. This argument is passed to RandomSpl.
periodic	TRUE to generate a periodic spline; FALSE to generate an ordinary spline. This argument is passed to RandomSpl.
n2s.ratio	a noise-to-signal ratio that is used when simulating noisy y-values.
n.sim	number of simulations.
nc	number of CPU cores to use for parallel simulations.

Details

See Section 3 of the reference.

Value

A matrix of MSE values. It has n.sim rows and (3 * degree) named columns. A name is like "spl.m", where "spl" is an abbreviation ("os", "sps" or "gps") and "m" is a penalty order from 1 to degree.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

References

Li, Z. and Cao, J. (2022). General P-Splines for Non-Uniform B-Splines.

Examples

```
## Not run:

require(gps.mgcv)

SimStudy(n = 400, degree = 3, n2s.ratio = 0.2, n.sim = 5)

## End(Not run)
```

Index