General P-Splines for Non-Uniform B-Splines (R code)

Zheyuan Li (zheyuan.li@bath.edu) and Jiguo Cao (jiguo_cao@sfu.ca)

Follow this document to reproduce all demos, computational results, simulation outcomes and real data analysis in the paper. You need to first install **gps** and **gps.mgcv** either from CRAN or GitHub.

```r
## installation from CRAN
install.packages("gps")
install.packages("gps.mgcv")

## installation from GitHub
## you may need to first install "devtools" from CRAN
devtools::install_github("ZheyuanLi/gps")
devtools::install_github("ZheyuanLi/gps.mgcv")
```

Load required packages and define a utility function that is used throughout this document.

```r
library(splines)
library(mgcv)
library(gps)
library(gps.mgcv)

## 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)
  ## show knot location
  abline(v = gamfit$smooth[[1]]$knots, lty = 2, col = 8)
  ## overlay g(x)
  nx <- length(x)
  xg <- seq(min(x), max(x), length = nx)
  lines(xg, g(xg), col = 2, lwd = 2)
  ## plot estimated f(x) at xg
  yh <- predict(gamfit, newdata = data.frame(x = xg))
  lines(xg, yh, lwd = 2)
}
```

1 Introduction

The following code produces Figure 1. (Figure 2 is produced in 4.1 BMC Longitudinal Data.)

```
pdf("fig1.pdf", width = 6, height = 4)
gps::DemoKnots(aligned = TRUE)
dev.off()
```

2 The New General P-Spline

2.1 Standard P-Spline

The following code produces Figure 3.

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

## number of data
n <- 500
## noise-to-signal ratio
n2s.ratio <- 0.4
## x-values
x <- qnorm(seq(pnorm(-3), pnorm(3), length = n))
## noisy y-values
gx <- g(x)
set.seed(2021); y <- rnorm(length(gx), gx, n2s.ratio * sd(gx))

## number of interior knots
k <- 50

## fit 4 types of penalized B-splines
fit <- gps.mgcv::Fit4BS(x, y, k, select = c("sps", "nps"))

pdf("fig3.pdf", width = 7.5, height = 2.5)
par(mfrow = c(1, 3), mar = c(2, 2, 1.5, 0.5))
## uniform B-splines
PlotXYFit(fit$sps, x, y, g)
title("uniform B-splines")
## non-uniform B-splines
```

```
PlotXYFit(fit$nps, x, y, g)
title("non-uniform B-splines")
## boxplot of MSE
mse <- gps.mgcv::MSE4BS(x, g, k, n2s.ratio = n2s.ratio, select = c("sps", "nps"))
colnames(mse) <- c("uniform", "non-uniform")
boxplot(mse, main = "MSE (100 simulations)")
dev.off()
```

The following code produces Figure 4.

```
pdf("fig4.pdf", width = 6, height = 4)
gps::DemoNull(n = 100, k = 10, gps = FALSE)
dev.off()
```

2.2 Introducing General P-Spline

The following code computes general difference matrices of order 1 to 3 on the example knot sequence.

```
## general difference matrices of order 1 to 3
D <- gps::SparseD(xt = c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4), d = 4)
## 1st order
D[[1]]
## 2nd order
D[[2]]
## 3rd order
D[[3]]
```

2.3 Justifying General P-Spline

The following code produces Figure 5.

```
pdf("fig5.pdf", width = 7, height = 5)
gps::DemoBS(uniform = FALSE, clamped = FALSE)
dev.off()
```

2.4 Connection with O-Spline

The following code computes matrix “S.bar” of order 1 to 3 on the example knot sequence.

```
gps::SandBar(xt = c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4), d = 4, m = 1)
gps::SandBar(xt = c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4), d = 4, m = 2)
gps::SandBar(xt = c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4), d = 4, m = 3)
```

The following code produces Figure 6 and 7.

```r
SamplePrior <- function (n, d, m, shape1 = 3, shape2 = 3) {
  ## place knots from Beta(shape1, shape2) for 50 B-splines
  p <- 50
  nd <- p - d + 2
  if (shape1 < 1 && shape2 < 1) {
    xd <- seq(-1/8, 1/8, length = nd)
    xd <- sign(xd) * abs(xd)^(1/3) + 0.5
    distr <- "U-quadratic(0, 1)"
  } else {
    xd <- qbeta(seq(0, 1, length = nd), shape1, shape2)
    distr <- sprintf("Beta(%d, %d)", shape1, shape2)
  }
  xt <- c(numeric(d - 1), xd, rep.int(1, d - 1))
  ## D and K
  D <- gps::SparseD(xt, d)[[m]]
  K <- gps::SparseS(xt, d, m, root = TRUE)
  ## Moore-Penrose generalized inverse of D'D and K'K
  SigmaD <- gps::MPinverse(D)
  SigmaK <- gps::MPinverse(K)
  ## marginal standard deviation
  sd1 <- sqrt(diag(SigmaD))
  sd2 <- sqrt(diag(SigmaK))
  ## prior B-spline coefficients
  ## use a common random seed so that they are constructed from the same innovation
  set.seed(0); b1 <- gps::PriorCoef(n, D)
  set.seed(0); b2 <- gps::PriorCoef(n, K)
  ## put them on the same scale
  b1 <- b1 / max(sd1)
  b2 <- b2 / max(sd2)
  ## return
  list(gps = b1, os = b2, distr = distr)
}

ComparePrior1 <- function (d, m, shape1, shape2) {
  ## draw 5 samples
  b <- SamplePrior(n = 5, d, m, shape1, shape2)
  ## plot these samples side by side
  ylim <- range(b$gps, b$os)
  par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
  matplot(b$gps, type = "l", lty = 1, lwd = 2, ann = FALSE, ylim = ylim)
  title("general difference penalty")
  matplot(b$os, type = "l", lty = 1, lwd = 2, ann = FALSE, ylim = ylim)
  title("derivative penalty")
}

pdf("fig6.pdf", width = 7, height = 3.5)
ComparePrior1(d = 4, m = 2, shape1 = 3, shape2 = 3)
dev.off()

ComparePrior2 <- function (d, m, shape1, shape2) {
  ## draw 1000 samples (large samples for assessing distribution)
  b <- SamplePrior(n = 1000, d, m, shape1, shape2)
  ## flatten them
  b1 <- c(b$gps)
  b2 <- c(b$os)
  ## correlation
  rho <- cor(b1, b2)
  ## fit a regression line b1 ~ b2
  fit <- .lm.fit(cbind(1, b2), b1)
  intercept <- fit$coefficients[1]
  slope <- fit$coefficients[2]
  ## plot ribbon
  b2.breaks <- quantile(b2, probs = seq(0, 1, 0.01))
  b2.bins <- cut(b2, breaks = b2.breaks)
  n.bins <- length(b2.breaks) - 1
  resid.per.bin <- split(fit$residuals, b2.bins)
  resid.qts <- lapply(resid.per.bin, quantile, prob = c(0.025, 0.25, 0.75, 0.975))
  resid.qts <- do.call(rbind, resid.qts)
  mean.line <- intercept + slope * b2.breaks
  shift <- rbind(resid.qts[1, ],
                 0.5 * (resid.qts[-1, ] + resid.qts[-n.bins, ]) / 2,
                 resid.qts[n.bins, ])
  qts.line <- mean.line + shift
  lim <- range(qts.line, b2.breaks)
  matplot(b2.breaks, qts.line, type = "l", lty = c(1, 2, 2, 1), col = 1, ann = FALSE,
          xlim = lim, ylim = lim)
  legend("topleft", legend = sprintf("cor = %.2f", rho), cex = 1.5, bty = "n")
  title(b$distr)
}

pdf("fig7.pdf", width = 7.5, height = 5)
par(mfrow = c(2, 3), mar = c(2, 2, 1.5, 0.5))
ComparePrior2(d = 4, m = 2, shape1 = 3, shape2 = 5)
ComparePrior2(d = 4, m = 2, shape1 = 5, shape2 = 5)
ComparePrior2(d = 4, m = 2, shape1 = 5, shape2 = 3)
ComparePrior2(d = 4, m = 2, shape1 = 1, shape2 = 3)
ComparePrior2(d = 4, m = 2, shape1 = 0.5, shape2 = 0.5)
ComparePrior2(d = 4, m = 2, shape1 = 3, shape2 = 1)
dev.off()
```

3 Simulation Studies

U-Shaped Curve Example The following code produces Figure 8.

```
## n, number of data
## n2s.ratio, noise-to-signal ratio
## k, number of interior knots
UCurve <- function (n = 500, n2s.ratio = 0.4, k = 50) {
  ## g(x)
  g <- function (x) (1 / 8) * abs(x) ^ 3
  ## x-values
  x <- qnorm(seq(pnorm(-3), pnorm(3), length = n))
  ## noisy y-values
  gx <- g(x)
  y <- rnorm(length(gx), gx, n2s.ratio * sd(gx))
  ## fit O-spline and standard/general P-spline
  fit <- gps.mgcv::Fit4BS(x, y, k, select = c("os", "sps", "gps"))
  ## general P-spline
  PlotXYFit(fit$gps, x, y, g)
  title("general P-spline")
  ## boxplot of MSE
  mse <- gps.mgcv::MSE4BS(x, g, k, n2s.ratio = n2s.ratio, select = c("os", "sps", "gps"))
  boxplot(mse, main = "MSE (100 simulations)")
}

set.seed(2021)
pdf("fig8.pdf", width = 8, height = 4)
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
UCurve(n = 500, n2s.ratio = 0.4, k = 50)
dev.off()
```

Normal Mixture Example The following code produces Figure 9.

```
## n, number of data
## n2s.ratio, noise-to-signal ratio
## k, number of interior knots
NormalMix <- function (n = 500, n2s.ratio = 0.4, k = 50) {
  ## g(x)
  g <- function (x) 5 * (dnorm(x, -1, 0.5) + dnorm(x, 1.2, 0.8))
  ## sample x-values
  N <- 1.1 * n
  n1 <- n2 <- round(N / 3)
  n3 <- N - n1 - n2
  x <- c(rnorm(n1, -1, 0.35), rnorm(n2, 1.2, 0.35), rnorm(n3, 0.1, 0.2))
  x <- x[x > -2 & x < 2]
  x <- sort(x[sample.int(n)])
  ## noisy y-values
  gx <- g(x)
  y <- rnorm(n, mean = gx, sd = n2s.ratio * sd(gx))
  ## fit O-spline and standard/general P-spline
  fit <- gps.mgcv::Fit4BS(x, y, k, select = c("os", "sps", "gps"))
  ## general P-spline
  PlotXYFit(fit$gps, x, y, g)
  title("general P-spline")
  ## boxplot of MSE
  mse <- gps.mgcv::MSE4BS(x, g, k, n2s.ratio = n2s.ratio, select = c("os", "sps", "gps"))
  boxplot(mse, main = "MSE (100 simulations)")
}

set.seed(2021)
pdf("fig9.pdf", width = 8, height = 4)
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
```

```r
NormalMix(n = 500, n2s.ratio = 0.4, k = 50)
dev.off()
```

Random Curve Example The following code produces Figure 10 and 11. Simulations are time-consuming, so I did them using `nc = 4` CPU cores.

```r
pdf("fig10.pdf", width = 6, height = 3)
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
set.seed(1000)
spl <- gps.mgcv::RandomSpl(400, 3); title("random cubic spline 1")
set.seed(1001)
spl <- gps.mgcv::RandomSpl(400, 3); title("random cubic spline 2")
dev.off()

## simulations with random cubic splines
cubic_1000_0.1 <- gps.mgcv::SimStudy(n = 1000, degree = 3, n2s.ratio = 0.1, nc = 4)
cubic_1000_0.5 <- gps.mgcv::SimStudy(n = 1000, degree = 3, n2s.ratio = 0.5, nc = 4)
cubic_5000_0.1 <- gps.mgcv::SimStudy(n = 5000, degree = 3, n2s.ratio = 0.1, nc = 4)
cubic_5000_0.5 <- gps.mgcv::SimStudy(n = 5000, degree = 3, n2s.ratio = 0.5, nc = 4)

## plot grouped boxplot
PlotMSE <- function (mse, n, gamma) {
  N <- ncol(mse) / 3
  at <- matrix(seq_len(4 * N), 4, N)[1:3, ]
  lab <- rep_len(c("os", "sps", "gps"), 3 * N)
  m <- sprintf("m = %d", 1:N)
  boxplot(mse, at = at, names = lab, xlim = extendrange(at, f = 0.01))
  mtext(m, side = 1, at = colMeans(at), line = 2.5)
  legend("top", legend = sprintf("n = %d, gamma = %.1f", n, gamma), bty = "n", cex = 1.5)
}

pdf("fig11.pdf", width = 9, height = 6)
par(mfrow = c(2, 2), mar = c(3.5, 2.5, 0.5, 0.5))
PlotMSE(cubic_1000_0.1, 1000, 0.1)
PlotMSE(cubic_1000_0.5, 1000, 0.5)
PlotMSE(cubic_5000_0.1, 5000, 0.1)
PlotMSE(cubic_5000_0.5, 5000, 0.5)
dev.off()
```

4 Real Data Examples

4.1 BMC Longitudinal Data

The paper only takes male group for demonstration.

```r
BMC <- synBMC(gender = "male")
head(BMC)
```

```r
## number of subjects
N <- nlevels(BMC$id)
## split this data set by subject
BMC.id <- split(BMC[-1], BMC$id)
```

The following code produces Figure 12.

```r
pdf("fig12.pdf", width = 7.5, height = 2.5)
par(mfrow = c(1, 3), mar = c(2, 2, 1.5, 0.5))
## (A) BMC data (bmc ~ age)
with(BMC, plot(age, bmc, type = "n", ann = FALSE))
for (i in 1:N) with(BMC.id[[i]], lines(age, bmc, col = i))
title("(A) bmc ~ age")
## (B) BMC data (log(bmc) ~ age)
with(BMC, plot(age, log.bmc, type = "n", ann = FALSE))
for (i in 1:N) with(BMC.id[[i]], lines(age, log.bmc, col = i))
title("(B) log(bmc) ~ age")
## (C) distribution of age
hist(BMC$age, breaks = 15, ann = FALSE)
title("(C) distribution of age")
dev.off()
```

gps.mgcv provides a function `FitBMC` for fitting standard/general P-spline to BMC data. You may read its source code to see how an additive mixed model is specified using factor smooth (“fs”) and estimated via `mgcv::gamm`.

```r
## fit BMC using 10 and 20 B-splines
gps.10 <- gps.mgcv::FitBMC(gender = "male", p = 10, gps = TRUE)$gam
sps.10 <- gps.mgcv::FitBMC(gender = "male", p = 10, gps = FALSE)$gam
gps.20 <- gps.mgcv::FitBMC(gender = "male", p = 20, gps = TRUE)$gam
sps.20 <- gps.mgcv::FitBMC(gender = "male", p = 20, gps = FALSE)$gam
```

Predict population and subject trajectories.

```r
## a grid of age values to make prediction
age <- seq(min(BMC$age), max(BMC$age), length = 100)
## newdata
newdat1 <- data.frame(age = rep.int(age, N),
                      id = rep(levels(BMC$id), each = length(age)))
newdat2 <- data.frame(age = age, id = "M001")

## 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))
sub.gps.20 <- matrix(exp(predict(gps.20, newdat1)), nrow = length(age))
sub.sps.20 <- matrix(exp(predict(sps.20, 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.20 <- predict(gps.20, newdat2, type = "terms", terms = "s(age)")
pop.sps.20 <- predict(sps.20, 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"))
pop.gps.20 <- exp(as.numeric(pop.gps.20) + attr(pop.gps.20, "constant"))
pop.sps.20 <- exp(as.numeric(pop.sps.20) + attr(pop.sps.20, "constant"))

## extract knots
knots.gps.10 <- gps.10$smooth[[1]]$knots
knots.sps.10 <- sps.10$smooth[[1]]$knots
knots.gps.20 <- gps.20$smooth[[1]]$knots
knots.sps.20 <- sps.20$smooth[[1]]$knots
```

The following code produces Figure 13.

```r
PlotID <- function (id1, id2, pop.gps, pop.sps, sub.gps, sub.sps,
                    knots.gps, knots.sps, ylim) {
  dat1 <- BMC.id[[id1]]
  gps1 <- sub.gps[, id1]
  sps1 <- sub.sps[, id1]
  dat2 <- BMC.id[[id2]]
  gps2 <- sub.gps[, id2]
  sps2 <- sub.sps[, id2]
  matplot(age, cbind(pop.gps, pop.sps), type = "l", lwd = 2, col = 8,
          ann = FALSE, ylim = ylim, xaxt = "n", yaxt = "n")
  points(dat1$age, dat1$bmc, col = 8, pch = 19, cex = 2)
  matlines(age, cbind(gps1, sps1), lwd = 2, col = 1)
  points(dat2$age, dat2$bmc, col = 8, pch = 19, cex = 2)
  matlines(age, cbind(gps2, sps2), lwd = 2, col = 1)
  rug(knots.gps, ticksize = 0.02, side = 1, lwd = 2)
  rug(knots.sps, ticksize = 0.02, side = 3, lwd = 2)
}

## get domain knots from full knots
domain.knots.gps.10 <- knots.gps.10[4:11]
domain.knots.sps.10 <- knots.sps.10[4:11]
domain.knots.gps.20 <- knots.gps.20[4:21]
domain.knots.sps.20 <- knots.sps.20[4:21]

pdf("fig13.pdf", width = 9, height = 6)
par(mfrow = c(2, 3), mar = c(0, 0, 1.5, 0), oma = c(0.5, 2, 0, 0.5))
## p = 10
ylim <- range(sub.sps.10[, c(27, 41, 63, 47, 66, 8)])
PlotID(27, 47, pop.gps.10, pop.sps.10, sub.gps.10, sub.sps.10,
       domain.knots.gps.10, domain.knots.sps.10, ylim)
axis(side = 2)
PlotID(41, 66, pop.gps.10, pop.sps.10, sub.gps.10, sub.sps.10,
       domain.knots.gps.10, domain.knots.sps.10, ylim)
title("10 B-splines")
PlotID(63,  8, pop.gps.10, pop.sps.10, sub.gps.10, sub.sps.10,
       domain.knots.gps.10, domain.knots.sps.10, ylim)
```

```
## p = 20
ylim <- range(sub.sps.20[, c(17, 41, 63, 84, 66, 8)])
PlotID(27, 47, pop.gps.20, pop.sps.20, sub.gps.20, sub.sps.20,
       domain.knots.gps.20, domain.knots.sps.20, ylim)
axis(side = 2)
PlotID(41, 66, pop.gps.20, pop.sps.20, sub.gps.20, sub.sps.20,
       domain.knots.gps.20, domain.knots.sps.20, ylim)
title("20 B-splines")
PlotID(63,  8, pop.gps.20, pop.sps.20, sub.gps.20, sub.sps.20,
       domain.knots.gps.20, domain.knots.sps.20, ylim)
dev.off()
```

The following code produces Figure 14. Cross-validation is extremely time-consuming! I did them on a cluster using `nc = 25` CPU cores.

```
## run cross-validation
cv.gps <- gps.mgcv::cvBMC(gender = "male", gps = TRUE, nc = 25)
cv.sps <- gps.mgcv::cvBMC(gender = "male", gps = FALSE, nc = 25)

## for BMC data, "gps" is more effective that "ps"
pdf("fig14.pdf", width = 6, height = 4)
par(mar = c(3, 3, 0.5, 0.5))
matplot(5:20, cbind(cv.gps, cv.sps), type = "l", ann = FALSE,
        lwd = 2, lty = c(1, 2, 2), col = rep(1:2, each = 3))
mtext("number of B-splines (p)", side = 1, line = 2)
mtext("CV score", side = 2, line = 2)
dev.off()
```

The following code produces Figure 2.

```
pdf("fig2.pdf", width = 8, height = 4)
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
## a common ylim for the next 2 plots
ylim <- range(sub.sps.10[, c(41, 66)])
## standard P-spline, population trajectory and subject 41, 66
plot(age, pop.sps.10, type = "n", ylim = ylim, ann = FALSE)
abline(v = domain.knots.sps.10, lty = 2, col = 8)
lines(age, pop.sps.10, lwd = 2)
with(BMC.id[[41]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
with(BMC.id[[66]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
lines(age, sub.sps.10[, 41], lwd = 2, lty = 2)
lines(age, sub.sps.10[, 66], lwd = 2, lty = 2)
title("P-spline")
## general P-spline, population trajectory and subject 41, 66
plot(age, pop.gps.10, type = "n", lwd = 2, ylim = ylim, ann = FALSE)
abline(v = domain.knots.gps.10, lty = 2, col = 8)
lines(age, pop.gps.10, lwd = 2)
with(BMC.id[[41]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
with(BMC.id[[66]], points(age, bmc, pch = 19, cex = 1.5, col = 8))
lines(age, sub.gps.10[, 41], lwd = 2, lty = 2)
lines(age, sub.gps.10[, 66], lwd = 2, lty = 2)
title("O-spline")
dev.off()
```

4.2 Fossil Shell Data

Consider the `smooth.spline` fit to the data as “truth”.

```
## you may need to first install "SemiPar" from CRAN
data(fossil, package = "SemiPar")

## treat fossil data as (x, y) data
x <- fossil$age
y <- fossil$strontium.ratio
ind <- order(x)
x <- x[ind]
y <- y[ind]

## consider a cubic smoothing spline fit as g(x)
sm <- smooth.spline(x, y)
g <- function (x) predict(sm, x = x)$y
```

The following code produces Figure 15.

```
## fit 4 types of penalized B-splines with 20 interior knots
fit <- gps.mgcv::Fit4BS(x, y, k = 20, select = c("gps", "sps"))

pdf("fig15.pdf", width = 8, height = 4)
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
PlotXYFit(fit$gps, x, y, g)
title("gps (20 interior knots)")
PlotXYFit(fit$sps, x, y, g)
title("sps (20 interior knots)")
dev.off()
```

The following code produces Figure 16.

```
## use the same knots as placed by smooth.spline()
nk <- length(sm$fit$knot) - 8
fit <- gps.mgcv::Fit4BS(x, y, k = nk, select = c("gps", "sps"))

pdf("fig16.pdf", width = 8, height = 4)
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
PlotXYFit(fit$gps, x, y, g)
title("gps (62 interior knots)")
PlotXYFit(fit$sps, x, y, g)
title("sps (62 interior knots)")
dev.off()
```

```
## residual sum of squares
sum(fit$sps$residuals ^ 2)
sum(fit$gps$residuals ^ 2)
sum((y - sm$y) ^ 2)
```

The following code produces Figure 17.

```
## extract leave-one-out GCV score for various k
k <- seq(10, 60, by = 2)
pse <- matrix(0, length(k), 2)
for (i in 1:length(k)) {
  fit.i <- gps.mgcv::Fit4BS(x, y, k[i], select = c("gps", "sps"))
  pse[i, ] <- unlist(lapply(fit.i, "[[", "gcv.ubre.dev"))
}

pdf("fig17.pdf", width = 6, height = 4)
par(mar = c(3, 3, 0.5, 0.5))
matplot(k, pse, type = "l", lty = c(2, 1), lwd = 2, col = 1, ann = FALSE)
mtext("number of interior knots (k)", side = 1, line = 2)
mtext("CV score", side = 2, line = 2)
dev.off()
```

The following code produces Figure 18.

```
## a zoom-in on [92, 106]
ind <- (x < 106)
xx <- x[ind]
yy <- y[ind]

pdf("fig18.pdf", width = 8, height = 3)
par(mfrow = c(1, 3), mar = c(2, 2, 1.5, 0.5))
k <- c(10, 13, 16)
for (i in 1:3) {
  fit.i <- gps.mgcv::Fit4BS(x, y, k[i], select = "gps")
  PlotXYFit(fit.i$gps, xx, yy, g)
  title(sprintf("k = %d", k[i]), cex.main = 2)
}
dev.off()
```

Discussion

The following code computes sparse/dense “root” on the example knot sequence.

```
xt <- c(0, 0, 0, 0, 1, 3, 4, 4, 4, 4)
gps::SparseS(xt, d = 4, m = 1, root = TRUE)
gps::SparseS(xt, d = 4, m = 2, root = TRUE)
gps::SparseS(xt, d = 4, m = 3, root = TRUE)

DenseRoot <- function (xt, d, m) {
  S <- gps::SparseS(xt, d, m)
  ei <- eigen(as.matrix(S))
  truncation <- seq_len(length(ei$values) - m)
  with(ei, sqrt(values[truncation]) * t(vectors[, truncation]))
}

DenseRoot(xt, d = 4, m = 1)
DenseRoot(xt, d = 4, m = 2)
DenseRoot(xt, d = 4, m = 3)
```

The following code demonstrates Yao and Lee (2008)'s idea to improve quantile knots.

```
## you may need to first install "SemiPar" from CRAN
data(fossil, package = "SemiPar")

## treat fossil data as (x, y) data
x <- fossil$age
y <- fossil$strontium.ratio

## consider a cubic smoothing spline fit as g(x)
sm <- smooth.spline(x, y)
g <- function (x) predict(sm, x = x)$y

## fit an initial general P-spline
fit <- gps.mgcv::Fit4BS(x, y, k = 20, select = "gps")$gps
Iknots <- fit$smooth[[1]]$interior.knots

## find locations of local extrema
xg <- seq(min(x), max(x), length = 201)
yg <- unname(predict(fit, newdata = data.frame(x = xg)))
plot(xg, yg)
xp <- xg[which(diff(sign(diff(yg))) != 0) + 1]

## refit general P-spline
aug.Iknots <- sort(c(Iknots, xp))
aug.k <- length(aug.Iknots)
refit <- gps.mgcv::Fit4BS(x, y, k = aug.k, knots = aug.Iknots, select = "gps")$gps

## we can see an improvement
par(mfrow = c(1, 2), mar = c(2, 2, 1.5, 0.5))
PlotXYFit(fit, x, y, g)
title("initial gps fit")
PlotXYFit(refit, x, y, g)
title("gps refit")
```

The following code produces Figure 19.

```
g <- function (x) x + sin(5 * pi * x ^ 5)

## number of data
```

```r
n <- 500
## noise-to-signal ratio
n2s.ratio <- 0.25
## x-values
x <- seq(0, 1, length = n)
## noisy y-values
gx <- g(x)
set.seed(2021); y <- rnorm(n, gx, n2s.ratio * sd(gx))

## manually place knots for general P-splines
k <- 50
Iknots <- seq(0.61, 1, length = k + 2)[2:(k + 1)]

fit <- gps.mgcv::Fit4BS(x, y, k, knots = Iknots, select = c("sps", "gps"))

pdf("fig19.pdf", width = 9, height = 3)
par(mfrow = c(1, 3), mar = c(2, 2, 1.5, 0.5))
PlotXYFit(fit$sps, x, y, g)
title("standard P-spline")
PlotXYFit(fit$gps, x, y, g)
title("general P-spline")
mse <- gps.mgcv::MSE4BS(x, g, k, knots = Iknots, n2s.ratio = n2s.ratio,
                        select = c("sps", "gps"))
boxplot(mse, main = "MSE (100 simulations)")
dev.off()
```

Appendix: Spline and B-Splines

The following code produces Figure 20.

```r
pdf("fig20.pdf", width = 6, height = 4)
out <- gps::DemoSpl(uniform = TRUE)
dev.off()
```

The following code computes coefficients for clamped B-splines.

```r
## 8 equidistant points on the domain (because there are 8 B-splines)
x <- seq(1, 6, length = 8)
## full knot sequence for uniform B-splines
xt1 <- -2:9
## full knot sequence for clamped B-splines
xt2 <- c(1, 1, 1, 1:6, 6, 6, 6)
## evaluate the spline using uniform B-splines
B1 <- splines::splineDesign(xt1, x)
y <- B1 %*% out$bspl.coef
## back solve coefficients of clamped B-splines
B2 <- splines::splineDesign(xt2, x)
solve(B2, y)
```