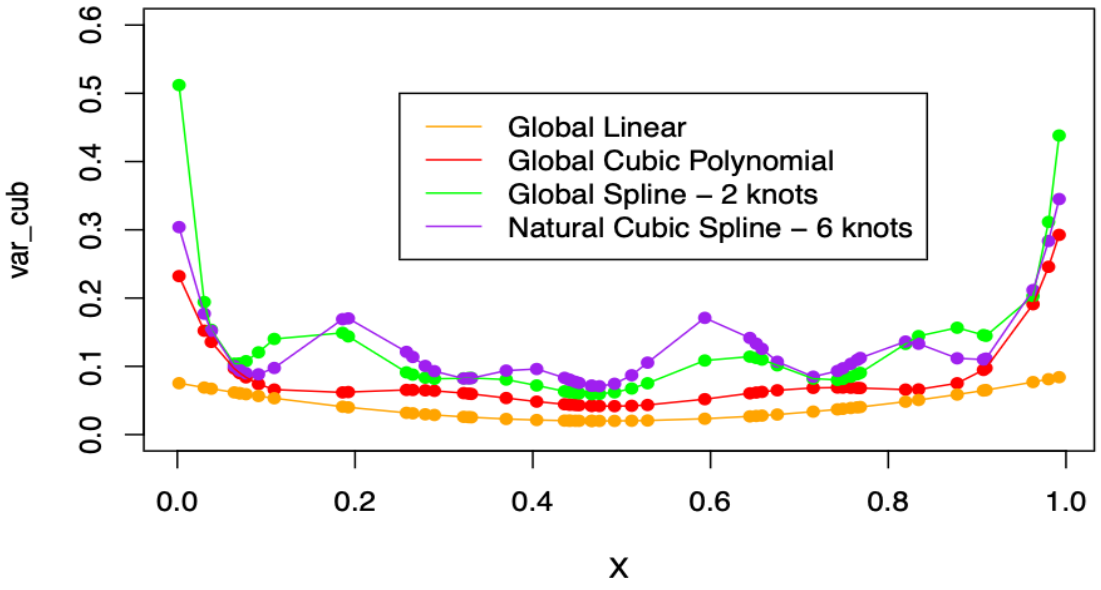
STAT790 A3

2023-02-28

Q1

```
set.seed(20)
library(splines)
x <- runif(50)
y <- x + rnorm(50)
H1<- bs(x, degree = 1, df=2, intercept = TRUE, Boundary.knots = c(0,1))
sigma_lin <- solve(t(H1)%*%H1)
var_lin <- diag(H1%*%sigma_lin%*%t(H1))
H2 <- bs(x, degree = 3, df=4, intercept = TRUE, Boundary.knots = c(0,1))
sigma_cub <- solve(t(H2)%*%H2)
var_cub <- diag(H2%*%sigma_cub%*%t(H2))
H3 <- bs(x, degree = 3, df=4, intercept = TRUE, Boundary.knots = c(0,1), knots = c(0.33, 0.66))
sigma_spl <- solve(t(H3)%*%H3)
var_spl <- diag(H3%*%sigma_spl%*%t(H3))
knots <- seq(0, 1, length.out = 6)[2:5]
H4 <- ns(x, intercept = TRUE, Boundary.knots = c(0,1), knots = knots)
sigma_nat <- solve(t(H4)%*%H4)
var_nat <- diag(H4%*%sigma_nat%*%t(H4))
plot(x, var_cub, ylim = c(0,0.6), col = 'red', pch = 16, xlab = 'X')
points(x, var_lin, col='orange', pch = 16)
points(x, var_spl, col = 'green', pch = 16)
points(x, var_nat, col = 'purple', pch = 16)
lines(x[order(x)], var_lin[order(x)], col='orange')
lines(x[order(x)], var_spl[order(x)], col = 'green')
lines(x[order(x)], var_cub[order(x)], col = 'red')
lines(x[order(x)], var_nat[order(x)], col = 'purple')
legend(0.25, 0.5,
       legend = c('Global Linear', 'Global Cubic Polynomial', 'Global Spline - 2 knots',
                  'Natural Cubic Spline - 6 knots'),
       col = c('orange', 'red', 'green', 'purple'), lty = 1)
```


var_cub
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0
X
Global Linear
Global Cubic Polynomial
Global Spline – 2 knots
Natural Cubic Spline – 6 knots

Q2

```
library(splines)
library(segmented)
```

```
## Loading required package: MASS
```

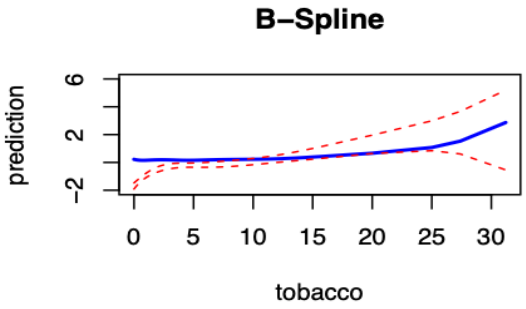
```
## Loading required package: nlme
```

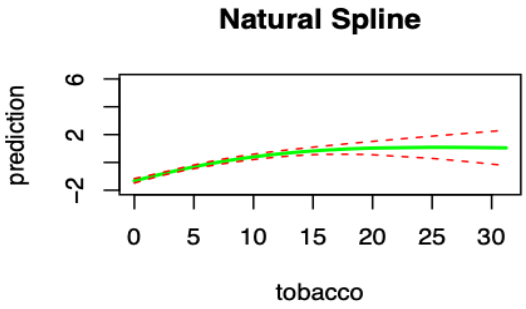
```
library(psre)
library(mgcv)
```

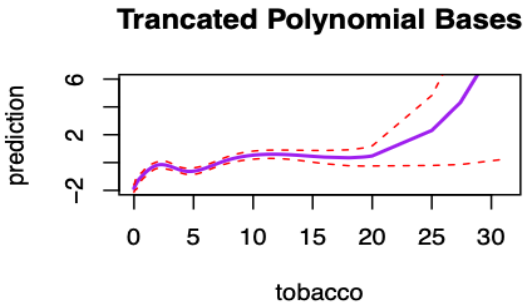
```
## This is mgcv 1.8-41. For overview type 'help("mgcv-package")'.
```

```
# Load data
data <- read.table("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data", header = TI

#fit model
bs_fit <- glm(chd ~ bs(tobacco, knots = 5), data = data, family = binomial)
ns_fit <- glm(chd ~ ns(tobacco, knots = 5), data = data, family = binomial)
tpb_fit <- glm(chd ~ tpb(tobacco, nknots = 5, degree = 2), data = data, family = binomial)

#coef
bs_coef <- bs_fit$coefficients
bs_prdt <- cbind(1,bs(data$tobacco, knots = 5)) %*% bs_coef
bs_var <- summary(bs_fit)$cov.scaled
bs_se <- sqrt(diag(cbind(1,bs(data$tobacco, knots = 5)) %*% bs_var %*% t(cbind(1,bs(data$tobacco, knots


ns_coef <- ns_fit$coefficients
ns_prdt <- cbind(1,ns(data$tobacco, knots = 5)) %*% ns_coef
ns_var <- summary(ns_fit)$cov.scaled
ns_se <- sqrt(diag(cbind(1,ns(data$tobacco, knots = 5)) %*% ns_var %*% t(cbind(1,ns(data$tobacco, knots

tpb_coef <- tpb_fit$coefficients
tpb_prdt <- cbind(1,tpb(data$tobacco, nknots = 5, degree = 2)) %*% tpb_coef
tpb_var <- summary(tpb_fit)$cov.scaled
tpb_se <- sqrt(diag(cbind(1,tpb(data$tobacco, nknots = 5, degree = 2)) %*% tpb_var %*% t(cbind(1,tpb(da


plt <- data.frame(tobacco = data$tobacco,
                      bs_prdt =bs_prdt,
                      bs_se = bs_se,
                      ns_prdt = ns_prdt,
                      ns_se = ns_se,
                      tpb_prdt= tpb_prdt,
                      tpb_se = tpb_se)
plt <- plt[order(plt$tobacco),]

par(mfrow = c(2,2))
plot(plt$tobacco, plt$bs_se, type = "l",
     col = "blue", lwd = 2, ylab = "prediction", xlab = "tobacco",
     ylim = c(-2, 6), main = "B-Spline")
lines(plt$tobacco, plt$bs_prdt-plt$bs_se, col = "red",
      lty = 2)
lines(plt$tobacco, plt$bs_prdt+plt$bs_se, col = "red",
```

```
      lty = 2)

plot(plt$tobacco, plt$ns_prdt, type = "l",
     col = "green", lwd = 2, ylab = "prediction", xlab = "tobacco",
     ylim = c(-2, 6), main = "Natural Spline")
lines(plt$tobacco, plt$ns_prdt-plt$ns_se, col = "red",
      lty = 2)
lines(plt$tobacco,  plt$ns_prdt+plt$ns_se, col = "red",
      lty = 2)

plot(plt$tobacco, plt$tpb_prdt, type = "l",
     col = "purple", lwd = 2, ylab = "prediction", xlab = "tobacco",
     ylim = c(-2, 6), main = "Trancated Polynomial Bases")
lines(plt$tobacco, plt$tpb_prdt-plt$tpb_se, col = "red",
      lty = 2)
lines(plt$tobacco,plt$tpb_prdt+plt$tpb_se, col = "red",
      lty = 2)
```

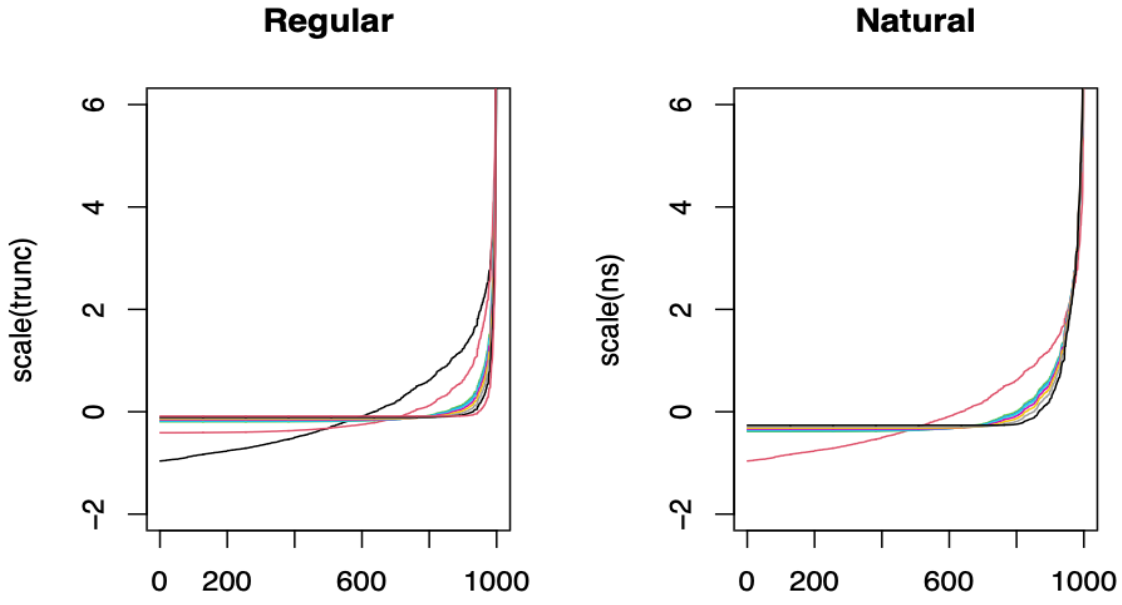






Q3

```
trunc_fun <- function(k) (x>=k)*(x-k)^3
trunc_poly_ns <- function(x, df, natural = FALSE) {
  knots <- quantile(x, probs = seq(0.05, 0.95, length = df - 1))
  S <- sapply(knots[1:(df-2)], trunc_fun)
  S <- as(S, "CsparseMatrix")
  S <- cbind(x, x^2, S)
  if (natural){
    S_temp <- sapply(knots, trunc_fun)
    Ndk <- matrix(ncol = ncol(S_temp)-2,nrow = nrow(S_temp))
    for (k in 1:(ncol(S_temp)-2)){
      Ndk[,k] <- (S_temp[,k]-S_temp[,ncol(S_temp)])/(knots[ncol(S_temp)]-knots[k])}
    Nk1 <- (S_temp[,ncol(S_temp)-1]-S_temp[,ncol(S_temp)])/(knots[ncol(S_temp)]-knots[ncol(S_temp)-1])
    S <- cbind(1, x, Ndk-Nk1)}
  as(S, "CsparseMatrix")

}

x <- rexp(1000)
x=x[order(x)]
trunc <- trunc_poly_ns(x, df = 10, FALSE)
ns <- trunc_poly_ns(x, df = 10, TRUE)


par(mfrow = c(1,2))
matplot(scale(trunc), ylim = c(-2,6),type = "l", col=1:10,main = "Regular",lty=1)
matplot(scale(ns), ylim = c(-2,6),type = "l", col=1:10,main = "Natural",lty=1)
```



Q4

```
generate_data <- function(n, poly_order, noise_sd) {
  x <- seq(0, 1, length.out = sqrt(n))
  y <- seq(0, 1, length.out = sqrt(n))
  grid <- expand.grid(x = x, y = y)

  # True function
  true_func <- function(x, y) {
    poly <- matrix(1, nrow = length(x), ncol = poly_order)
    for (i in 1:poly_order) {
      for (j in 0:i) {
        poly[, i] <- poly[, i] * x^j * (1 - x)^(i - j)
      }
    }
    beta <- rnorm(poly_order)
    poly %*% beta + 0.5
  }

  # Simulate data
  z <- true_func(grid$x, grid$y) + rnorm(n, sd = noise_sd)
  data.frame(x = grid$x, y = grid$y, z = z)
}
set.seed(123)
sim_data <- generate_data(n = 10000, poly_order = 3, noise_sd = 0.1)
head(sim_data)
```

```
##            x y         z
## 1 0.00000000 0 0.5070508
## 2 0.01010101 0 0.5073244
## 3 0.02020202 0 0.6604107
## 4 0.03030303 0 0.5296163
## 5 0.04040404 0 0.3517500
## 6 0.05050505 0 0.4044121
```

```
library(mgcv)
library(R.utils)
```

```
## Loading required package: R.oo
## Loading required package: R.methodsS3
## R.methodsS3 v1.8.2 (2022-06-13 22:00:14 UTC) successfully loaded. See ?R.methodsS3 for help.
## R.oo v1.25.0 (2022-06-12 02:20:02 UTC) successfully loaded. See ?R.oo for help.
##
## Attaching package: 'R.oo'
## The following object is masked from 'package:R.methodsS3':
##
##     throw
## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods
## The following objects are masked from 'package:base':
##
```

```
##     attach, detach, load, save
## R.utils v2.12.2 (2022-11-11 22:00:03 UTC) successfully loaded. See ?R.utils for help.
##
## Attaching package: 'R.utils'
## The following object is masked from 'package:utils':
##
##     timestamp
## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, isOpen, nullfile, parse, warnings
```

```
library(tictoc)
# set up simulation parameters
n_sims <- 250
n_obs <- 1000
true_func <- function(x, y) sin(pi*x)*cos(pi*y)
sigma <- 0.1

# set up output storage
times1 <- numeric(n_sims)
bias1 <- numeric(n_sims)
variance1 <- numeric(n_sims)
mse1 <- numeric(n_sims)

times2 <- numeric(n_sims)
bias2 <- numeric(n_sims)
variance2 <- numeric(n_sims)
mse2 <- numeric(n_sims)

# perform simulations and evaluate GAM fits
for (i in 1:n_sims) {

  # simulate data
  set.seed(i)
  x <- runif(n_obs)
  y <- runif(n_obs)
  z <- true_func(x, y) + rnorm(n_obs, 0, sigma)
  dat=sim_data
  #dat <- data.frame(x = x, y = y, z = z)

  # fit GAM with GCV.Cp
  tic()
  fit1 <- gam(z ~ te(x, y), data = dat, method = "GCV.Cp")
  times1[i] <- toc()

  # compute predictions and errors
  pred1 <- predict(fit1, newdata = dat, type = "response")
  bias1[i] <- mean(pred1 - true_func(x, y))
  variance1[i] <- var(pred1)
  mse1[i] <- mean((pred1 - true_func(x, y))^2)

  # fit GAM with REML
```

```
  tic()
  fit2 <- gam(z ~ te(x, y), data = dat, method = "REML")
  times2[i] <- toc()

  # compute predictions and errors
  pred2 <- predict(fit2, newdata = dat, type = "response")
  bias2[i + n_sims] <- mean(pred2 - true_func(x, y))
  variance2[i + n_sims] <- var(pred2)
  mse2[i + n_sims] <- mean((pred2 - true_func(x, y))^2)

}
```

```
## 0.133 sec elapsed
## 0.792 sec elapsed
## 0.062 sec elapsed
## 0.525 sec elapsed
## 0.051 sec elapsed
## 0.528 sec elapsed
## 0.045 sec elapsed
## 0.517 sec elapsed
## 0.053 sec elapsed
## 0.304 sec elapsed
## 0.046 sec elapsed
## 0.309 sec elapsed
## 0.045 sec elapsed
## 0.294 sec elapsed
## 0.05 sec elapsed
## 0.305 sec elapsed
## 0.046 sec elapsed
## 0.309 sec elapsed
## 0.041 sec elapsed
## 0.51 sec elapsed
## 0.041 sec elapsed
## 0.307 sec elapsed
## 0.04 sec elapsed
## 0.314 sec elapsed
## 0.041 sec elapsed
## 0.513 sec elapsed
## 0.047 sec elapsed
## 0.311 sec elapsed
## 0.036 sec elapsed
## 0.298 sec elapsed
## 0.052 sec elapsed
## 0.515 sec elapsed
## 0.042 sec elapsed
## 0.301 sec elapsed
## 0.041 sec elapsed
## 0.305 sec elapsed
## 0.046 sec elapsed
## 0.515 sec elapsed
## 0.045 sec elapsed
## 0.305 sec elapsed
## 0.056 sec elapsed
## 0.323 sec elapsed
```

```
## 0.044 sec elapsed
## 0.546 sec elapsed
## 0.042 sec elapsed
## 0.303 sec elapsed
## 0.042 sec elapsed
## 0.311 sec elapsed
## 0.046 sec elapsed
## 0.532 sec elapsed
## 0.051 sec elapsed
## 0.299 sec elapsed
## 0.047 sec elapsed
## 0.522 sec elapsed
## 0.042 sec elapsed
## 0.301 sec elapsed
## 0.05 sec elapsed
## 0.305 sec elapsed
## 0.042 sec elapsed
## 0.514 sec elapsed
## 0.041 sec elapsed
## 0.302 sec elapsed
## 0.038 sec elapsed
## 0.302 sec elapsed
## 0.051 sec elapsed
## 0.506 sec elapsed
## 0.047 sec elapsed
## 0.297 sec elapsed
## 0.037 sec elapsed
## 0.298 sec elapsed
## 0.045 sec elapsed
## 0.516 sec elapsed
## 0.042 sec elapsed
## 0.304 sec elapsed
## 0.037 sec elapsed
## 0.301 sec elapsed
## 0.042 sec elapsed
## 0.521 sec elapsed
## 0.037 sec elapsed
## 0.296 sec elapsed
## 0.048 sec elapsed
## 0.307 sec elapsed
## 0.048 sec elapsed
## 0.509 sec elapsed
## 0.045 sec elapsed
## 0.311 sec elapsed
## 0.041 sec elapsed
## 0.309 sec elapsed
## 0.059 sec elapsed
## 0.59 sec elapsed
## 0.041 sec elapsed
## 0.304 sec elapsed
## 0.052 sec elapsed
## 0.521 sec elapsed
## 0.043 sec elapsed
## 0.506 sec elapsed
```

```
## 0.048 sec elapsed
## 0.3 sec elapsed
## 0.045 sec elapsed
## 0.524 sec elapsed
## 0.045 sec elapsed
## 0.3 sec elapsed
## 0.051 sec elapsed
## 0.298 sec elapsed
## 0.041 sec elapsed
## 0.512 sec elapsed
## 0.041 sec elapsed
## 0.306 sec elapsed
## 0.037 sec elapsed
## 0.302 sec elapsed
## 0.054 sec elapsed
## 0.51 sec elapsed
## 0.045 sec elapsed
## 0.299 sec elapsed
## 0.036 sec elapsed
## 0.308 sec elapsed
## 0.046 sec elapsed
## 0.506 sec elapsed
## 0.041 sec elapsed
## 0.311 sec elapsed
## 0.036 sec elapsed
## 0.289 sec elapsed
## 0.041 sec elapsed
## 0.3 sec elapsed
## 0.043 sec elapsed
## 0.308 sec elapsed
## 0.043 sec elapsed
## 0.522 sec elapsed
## 0.042 sec elapsed
## 0.303 sec elapsed
## 0.042 sec elapsed
## 0.304 sec elapsed
## 0.048 sec elapsed
## 0.509 sec elapsed
## 0.046 sec elapsed
## 0.303 sec elapsed
## 0.043 sec elapsed
## 0.304 sec elapsed
## 0.041 sec elapsed
## 0.508 sec elapsed
## 0.046 sec elapsed
## 0.314 sec elapsed
## 0.046 sec elapsed
## 0.522 sec elapsed
## 0.047 sec elapsed
## 0.307 sec elapsed
## 0.042 sec elapsed
## 0.302 sec elapsed
## 0.046 sec elapsed
## 0.508 sec elapsed
```

```
## 0.041 sec elapsed
## 0.299 sec elapsed
## 0.045 sec elapsed
## 0.312 sec elapsed
## 0.047 sec elapsed
## 0.288 sec elapsed
## 0.05 sec elapsed
## 0.301 sec elapsed
## 0.042 sec elapsed
## 0.303 sec elapsed
## 0.042 sec elapsed
## 0.515 sec elapsed
## 0.045 sec elapsed
## 0.306 sec elapsed
## 0.048 sec elapsed
## 0.302 sec elapsed
## 0.047 sec elapsed
## 0.517 sec elapsed
## 0.05 sec elapsed
## 0.334 sec elapsed
## 0.042 sec elapsed
## 0.301 sec elapsed
## 0.047 sec elapsed
## 0.505 sec elapsed
## 0.046 sec elapsed
## 0.306 sec elapsed
## 0.044 sec elapsed
## 0.312 sec elapsed
## 0.259 sec elapsed
## 0.299 sec elapsed
## 0.042 sec elapsed
## 0.299 sec elapsed
## 0.042 sec elapsed
## 0.309 sec elapsed
## 0.258 sec elapsed
## 0.296 sec elapsed
## 0.049 sec elapsed
## 0.298 sec elapsed
## 0.042 sec elapsed
## 0.306 sec elapsed
## 0.037 sec elapsed
## 0.29 sec elapsed
## 0.041 sec elapsed
## 0.303 sec elapsed
## 0.038 sec elapsed
## 0.301 sec elapsed
## 0.041 sec elapsed
## 0.311 sec elapsed
## 0.039 sec elapsed
## 0.291 sec elapsed
## 0.044 sec elapsed
## 0.299 sec elapsed
## 0.039 sec elapsed
## 0.304 sec elapsed
```

```
## 0.042 sec elapsed
## 0.307 sec elapsed
## 0.04 sec elapsed
## 0.299 sec elapsed
## 0.059 sec elapsed
## 0.314 sec elapsed
## 0.047 sec elapsed
## 0.305 sec elapsed
## 0.041 sec elapsed
## 0.569 sec elapsed
## 0.041 sec elapsed
## 0.305 sec elapsed
## 0.042 sec elapsed
## 0.305 sec elapsed
## 0.043 sec elapsed
## 0.527 sec elapsed
## 0.043 sec elapsed
## 0.302 sec elapsed
## 0.042 sec elapsed
## 0.31 sec elapsed
## 0.043 sec elapsed
## 0.508 sec elapsed
## 0.036 sec elapsed
## 0.3 sec elapsed
## 0.042 sec elapsed
## 0.304 sec elapsed
## 0.041 sec elapsed
## 0.513 sec elapsed
## 0.042 sec elapsed
## 0.303 sec elapsed
## 0.036 sec elapsed
## 0.299 sec elapsed
## 0.047 sec elapsed
## 0.518 sec elapsed
## 0.046 sec elapsed
## 0.304 sec elapsed
## 0.042 sec elapsed
## 0.305 sec elapsed
## 0.041 sec elapsed
## 0.509 sec elapsed
## 0.047 sec elapsed
## 0.31 sec elapsed
## 0.042 sec elapsed
## 0.305 sec elapsed
## 0.043 sec elapsed
## 0.513 sec elapsed
## 0.046 sec elapsed
## 0.305 sec elapsed
## 0.042 sec elapsed
## 0.311 sec elapsed
## 0.042 sec elapsed
## 0.541 sec elapsed
## 0.042 sec elapsed
## 0.298 sec elapsed
```

```
## 0.041 sec elapsed
## 0.3 sec elapsed
## 0.04 sec elapsed
## 0.516 sec elapsed
## 0.041 sec elapsed
## 0.302 sec elapsed
## 0.056 sec elapsed
## 0.307 sec elapsed
## 0.046 sec elapsed
## 0.512 sec elapsed
## 0.045 sec elapsed
## 0.31 sec elapsed
## 0.043 sec elapsed
## 0.313 sec elapsed
## 0.041 sec elapsed
## 0.523 sec elapsed
## 0.042 sec elapsed
## 0.301 sec elapsed
## 0.041 sec elapsed
## 0.312 sec elapsed
## 0.256 sec elapsed
## 0.307 sec elapsed
## 0.042 sec elapsed
## 0.306 sec elapsed
## 0.042 sec elapsed
## 0.314 sec elapsed
## 0.048 sec elapsed
## 0.512 sec elapsed
## 0.05 sec elapsed
## 0.316 sec elapsed
## 0.047 sec elapsed
## 0.303 sec elapsed
## 0.042 sec elapsed
## 0.515 sec elapsed
## 0.041 sec elapsed
## 0.307 sec elapsed
## 0.043 sec elapsed
## 0.348 sec elapsed
## 0.051 sec elapsed
## 0.304 sec elapsed
## 0.049 sec elapsed
## 0.329 sec elapsed
## 0.037 sec elapsed
## 0.307 sec elapsed
## 0.043 sec elapsed
## 0.317 sec elapsed
## 0.042 sec elapsed
## 0.546 sec elapsed
## 0.043 sec elapsed
## 0.309 sec elapsed
## 0.042 sec elapsed
## 0.309 sec elapsed
## 0.052 sec elapsed
## 0.521 sec elapsed
```

```
## 0.05 sec elapsed
## 0.311 sec elapsed
## 0.042 sec elapsed
## 0.315 sec elapsed
## 0.253 sec elapsed
## 0.303 sec elapsed
## 0.041 sec elapsed
## 0.316 sec elapsed
## 0.043 sec elapsed
## 0.313 sec elapsed
## 0.258 sec elapsed
## 0.301 sec elapsed
## 0.042 sec elapsed
## 0.305 sec elapsed
## 0.05 sec elapsed
## 0.311 sec elapsed
## 0.258 sec elapsed
## 0.299 sec elapsed
## 0.051 sec elapsed
## 0.303 sec elapsed
## 0.042 sec elapsed
## 0.309 sec elapsed
## 0.253 sec elapsed
## 0.306 sec elapsed
## 0.043 sec elapsed
## 0.302 sec elapsed
## 0.045 sec elapsed
## 0.522 sec elapsed
## 0.044 sec elapsed
## 0.313 sec elapsed
## 0.043 sec elapsed
## 0.406 sec elapsed
## 0.08 sec elapsed
## 0.526 sec elapsed
## 0.046 sec elapsed
## 0.313 sec elapsed
## 0.041 sec elapsed
## 0.315 sec elapsed
## 0.041 sec elapsed
## 0.523 sec elapsed
## 0.041 sec elapsed
## 0.301 sec elapsed
## 0.041 sec elapsed
## 0.308 sec elapsed
## 0.045 sec elapsed
## 0.519 sec elapsed
## 0.052 sec elapsed
## 0.297 sec elapsed
## 0.036 sec elapsed
## 0.303 sec elapsed
## 0.041 sec elapsed
## 0.519 sec elapsed
## 0.042 sec elapsed
## 0.304 sec elapsed
```

```
## 0.048 sec elapsed
## 0.512 sec elapsed
## 0.055 sec elapsed
## 0.299 sec elapsed
## 0.042 sec elapsed
## 0.319 sec elapsed
## 0.062 sec elapsed
## 0.547 sec elapsed
## 0.055 sec elapsed
## 0.315 sec elapsed
## 0.044 sec elapsed
## 0.304 sec elapsed
## 0.049 sec elapsed
## 0.536 sec elapsed
## 0.049 sec elapsed
## 0.307 sec elapsed
## 0.046 sec elapsed
## 0.309 sec elapsed
## 0.05 sec elapsed
## 0.517 sec elapsed
## 0.048 sec elapsed
## 0.307 sec elapsed
## 0.042 sec elapsed
## 0.323 sec elapsed
## 0.047 sec elapsed
## 0.531 sec elapsed
## 0.053 sec elapsed
## 0.315 sec elapsed
## 0.042 sec elapsed
## 0.317 sec elapsed
## 0.263 sec elapsed
## 0.31 sec elapsed
## 0.043 sec elapsed
## 0.317 sec elapsed
## 0.044 sec elapsed
## 0.317 sec elapsed
## 0.258 sec elapsed
## 0.309 sec elapsed
## 0.043 sec elapsed
## 0.308 sec elapsed
## 0.041 sec elapsed
## 0.316 sec elapsed
## 0.268 sec elapsed
## 0.297 sec elapsed
## 0.05 sec elapsed
## 0.301 sec elapsed
## 0.043 sec elapsed
## 0.313 sec elapsed
## 0.037 sec elapsed
## 0.291 sec elapsed
## 0.041 sec elapsed
## 0.302 sec elapsed
## 0.037 sec elapsed
## 0.303 sec elapsed
```

```
## 0.042 sec elapsed
## 0.316 sec elapsed
## 0.037 sec elapsed
## 0.292 sec elapsed
## 0.042 sec elapsed
## 0.307 sec elapsed
## 0.038 sec elapsed
## 0.305 sec elapsed
## 0.042 sec elapsed
## 0.314 sec elapsed
## 0.037 sec elapsed
## 0.295 sec elapsed
## 0.042 sec elapsed
## 0.303 sec elapsed
## 0.042 sec elapsed
## 0.311 sec elapsed
## 0.041 sec elapsed
## 0.521 sec elapsed
## 0.044 sec elapsed
## 0.307 sec elapsed
## 0.042 sec elapsed
## 0.306 sec elapsed
## 0.046 sec elapsed
## 0.52 sec elapsed
## 0.041 sec elapsed
## 0.3 sec elapsed
## 0.042 sec elapsed
## 0.308 sec elapsed
## 0.041 sec elapsed
## 0.517 sec elapsed
## 0.037 sec elapsed
## 0.304 sec elapsed
## 0.041 sec elapsed
## 0.317 sec elapsed
## 0.041 sec elapsed
## 0.512 sec elapsed
## 0.042 sec elapsed
## 0.304 sec elapsed
## 0.037 sec elapsed
## 0.298 sec elapsed
## 0.049 sec elapsed
## 0.51 sec elapsed
## 0.045 sec elapsed
## 0.305 sec elapsed
## 0.041 sec elapsed
## 0.314 sec elapsed
## 0.041 sec elapsed
## 0.525 sec elapsed
## 0.048 sec elapsed
## 0.305 sec elapsed
## 0.042 sec elapsed
## 0.301 sec elapsed
## 0.041 sec elapsed
## 0.507 sec elapsed
```

```
## 0.047 sec elapsed
## 0.313 sec elapsed
## 0.041 sec elapsed
## 0.305 sec elapsed
## 0.041 sec elapsed
## 0.519 sec elapsed
## 0.042 sec elapsed
## 0.297 sec elapsed
## 0.041 sec elapsed
## 0.302 sec elapsed
## 0.042 sec elapsed
## 0.519 sec elapsed
## 0.041 sec elapsed
## 0.298 sec elapsed
## 0.042 sec elapsed
## 0.307 sec elapsed
## 0.045 sec elapsed
## 0.504 sec elapsed
## 0.042 sec elapsed
## 0.3 sec elapsed
## 0.042 sec elapsed
## 0.302 sec elapsed
## 0.041 sec elapsed
## 0.512 sec elapsed
## 0.042 sec elapsed
## 0.3 sec elapsed
```

```
# compute summary statistics
mean_times1 <- mean(unlist(times1))
mean_bias1 <- mean(bias1)
mean_variance1 <- mean(variance1)
mean_mse1 <- mean(mse1)

# print results
cat("Average computation time: ", mean_times1, "\n")
```

```
## Average computation time:  65.75165
```

```
cat("Bias: ", mean_bias1, "\n")
```

```
## Bias:  0.4057118
```

```
cat("Variance: ", mean_variance1, "\n")
```

```
## Variance:  0.001820344
```

```
cat("Mean Squared Error: ", mean_mse1, "\n")
```

```
## Mean Squared Error:  0.4170228
```

```
# compute summary statistics
mean_times2 <- mean(unlist(times2))
mean_bias2 <- mean(bias2)
mean_variance2 <- mean(variance2)
mean_mse2 <- mean(mse2)

# print results
```

```
cat("Average computation time: ", mean_times2, "\n")
```

```
## Average computation time:  65.83054
```

```
cat("Bias: ", mean_bias2, "\n")
```

```
## Bias:  0.2028559
```

```
cat("Variance: ", mean_variance2, "\n")
```

```
## Variance:  0.0009192246
```

```
cat("Mean Squared Error: ", mean_mse2, "\n")
```

```
## Mean Squared Error:  0.2085207
```

from the results, we could see that the average computation time for REML is a little longer, but this method has smaller bias, variance and mean squared error.

exercise 5.4

When X $< \xi_1$, we have $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$, f(X) is linear, so we have $\beta_2 = \beta_3 = 0$

When X $> \xi_K$, we have $f(X) = \beta_0 + \beta_1 X + 3X \sum_{k=1}^{K} \theta_k \xi_k^2 - 3X^2 \sum_{k=1}^{K} \theta_k \xi_k + X^3 \sum_{k=1}^{K} \theta_k - \sum_{k=1}^{K} \theta_k \xi_k^3$, since f(X) would be linear at this case, so $\sum_{k=1}^{K} \theta_k = 0$ and $\sum_{k=1}^{K} \theta_k \xi_k = 0$

Then we rewrite the formula of f(X) as $\sum_{k=1}^{K} \alpha_k N_k(X)$, we could see that $\alpha_1 N_1(X) = \beta_0, \alpha_2 N_2(X) = \beta_1 X$, so $N_1(X) = 1, N_2(X) = X$

$\sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$

$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \theta_{K-1}(X - \xi_{K-1})_+^3 + \theta_K (X - \xi_K)_+^3$

$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + (\theta_{K-1} + \theta_K)(X - \xi_K)_+^3 + \theta_{K-1}((X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3)$

$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + (\theta_{K-1} + \theta_K)(X - \xi_K)_+^3 - \frac{-(\xi_K + \xi_{K-1})\theta_{K-1} + \xi_K \theta_K - \xi_K \theta_K}{\xi_K - \xi_{K-1}}((X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3)$

Since $\sum_{k=1}^{K} \theta_k = 0, \sum_{k=1}^{K} \theta_k \xi_k = 0$, we have $\sum_{k=1}^{K-2} \theta_k = -\theta_K - \theta_{K-1}, \sum_{k=1}^{K-2} \theta_k \xi_k = -\theta_K \xi_K - \theta_{K-1} \xi_{K-1}$, so:

$\sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$

$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \sum_{k=1}^{K-2} \theta_k (X - \xi_K)_+^3 - \frac{\xi_K (\sum_{k=1}^{K-2} \theta_k) - \sum_{k=1}^{K-2} \theta_k \xi_k}{\xi_K - \xi_{K-1}}((X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3)$

$= \sum_{k=1}^{K-2} (\xi_K - \xi_k)\theta_k [\frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3)}{\xi_K - \xi_k} - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3)}{\xi_K - \xi_{K-1}}]$

$= \sum_{k=1}^{K-2} \alpha_{k+2} N_{k+2}(X)$

so we have $\alpha_{k+2} = (\xi_K - \xi_k)\theta_k$

$N_{k+2}(X) = d_k(X) - d_{K-1}(X)$

where $d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3)}{\xi_K - \xi_k}$

exercise 5.13

If we augment the original sample with the pair $(x_0, \hat{f}_\lambda(x_0))$ and refit the smoothing spline, the resulting fitted curve will change. In general, the new fitted curve will be smoother and more likely to pass through the point $(x_0, \hat{f}_\lambda(x_0))$. This is because the addition of a new data point constrains the spline to pass through that point, which can cause the rest of the curve to adjust accordingly.

Specifically, the effect of adding a new data point depends on the location of x_0 and the value of $\hat{f}_\lambda(x_0)$ relative to the existing sample. If x_0 is close to an existing data point, the impact of adding x_0 may be minimal. However, if x_0 is far from the existing data points, the new fitted curve will likely be more affected.

we know $CV(\hat{f}_\lambda) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2$

$f_\lambda^{(-i)}(x_i) = \frac{\sum_{j\neq i} S_\lambda(i,j)y_j}{1-S_\lambda(i,i)}$

we also know $\sum_{j=1} S_\lambda(i,j)y_j = \hat{f}_\lambda(x_i)$

so $f_\lambda^{(-i)}(x_i) = \frac{\hat{f}_\lambda(x_i)-S_\lambda(i,i)y_i}{1-S_\lambda(i,i)}$

$y_i - \hat{f}_\lambda^{(-i)}(x_i) = y_i - \frac{\hat{f}_\lambda(x_i)-S_\lambda(i,i)y_i}{1-S_\lambda(i,i)} = \frac{y_i-\hat{f}_\lambda(x_i)}{1-S_\lambda(i,i)}$

so $CV(\hat{f}_\lambda) = \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i-\hat{f}_\lambda(x_i)}{1-S_\lambda(i,i)})^2$