

# STATS 790 Assignment 3

Yiran Zhang  
400119421

01 March, 2023

## Question 1

ESL Chapter 5 Figure 5.6 is replicated, the code is shown as below.

```
# Question 1
# Replicate ESL Chapter 5 Figure 5.6

# Import dataset
bone <- read.table('https://hastie.su.domains/ElemStatLearn/datasets/bone.data',
                  header = TRUE)

# According to ESL Figure 5.6 description, spnbmd is the target variable
# and age is the predictor.

# Plot the age against relative change in spinal BMD, color separated by gender.
plot(x = bone$age, y = bone$spnbmd,
     col = ifelse(bone$gender == 'female', 'red', 'blue'),
     pch = 20, xlab = 'Age', ylab = 'Relative Change in Spinal BMD')

# Split male and female.
male_bone <- bone[bone$gender == 'male', ]
female_bone <- bone[bone$gender == 'female', ]

# Spline, using degree of freedom = 12 (given by the textbook)
male_spline <- smooth.spline(x = male_bone$age, y = male_bone$spnbmd, df = 12)
female_spline <- smooth.spline(x = female_bone$age, y = female_bone$spnbmd, df = 12)
```

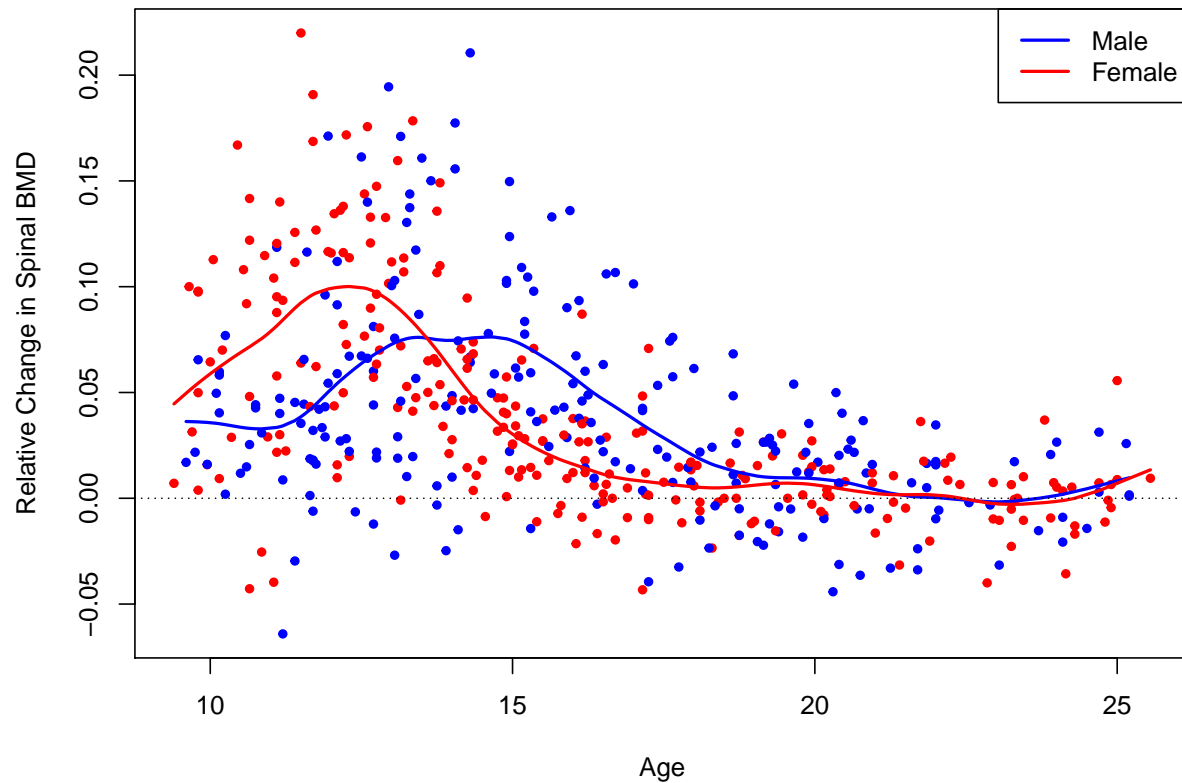
```

# Add splines to the graph with corresponding color for each gender.
lines(male_spline, col = 'blue', lwd = 2)
lines(female_spline, col = 'red', lwd = 2)

# Add a horizontal dash line at y = 0.
abline(h=0, lty=3)

# Add a legend at the top right corner.
legend(x='topright', legend=c('Male', 'Female'),
      col=c('blue', 'red'), lwd=2)

```



## Question 2 (South Africa coronary heart disease data)

```

# Question 2
# Import libraries

```

```

library(splines)

# Import South Africa coronary heart disease data.
url <- "http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data"
heart <- read.csv(url, row.names = 1)

# Create a list of 5 knots location for the bases
# Knots that can equally divide the tobacco range are chosen
# There are 107 out of 462 tobacco values are 0, 107/462 = 0.2316
# So the first knot need to be greater than 0.2316 to avoid NA values in glm
knots <- quantile(heart$tobacco, probs = c(0.24, 0.32, 0.48, 0.64, 0.8))

# B-spline base
b_spline <- bs(heart$tobacco, knots = knots)

# Natural spline base
n_spline <- ns(heart$tobacco, knots = knots)

# Truncated polynomial base
# Below is the function that creates truncated polynomial spline base
# it is modified based on Dr. Bolker's code
truncpolyspline <- function(x, knots) {
  trunc_fun <- function(k) (x > k)*(x-k)^3
  S <- sapply(knots, trunc_fun)
  S <- cbind(x, x^2, x^3, S)
  return(S)
}

poly_spline <- truncpolyspline(heart$tobacco, knots) # create basis matrix

# Fit logistic regression
# b-spline
logistic_b <- glm(chd ~ b_spline, data = heart, family = 'binomial')
# natural spline
logistic_n <- glm(chd ~ n_spline, data = heart, family = 'binomial')
# truncated polynomial spline

```

```
logistic_poly <- glm(chd ~ poly_spline, data = heart, family = 'binomial')
```

```
summary(logistic_b)
```

```
##
```

```
## Call:
```

```
## glm(formula = chd ~ b_spline, family = "binomial", data = heart)
```

```
##
```

```
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.4715  -0.9664  -0.5571   1.1216   2.2391
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.7847     0.2754  -6.480  9.2e-11 ***
## b_spline1    -1.1409     1.3165  -0.867  0.38614
## b_spline2     0.4664     0.9044   0.516  0.60610
## b_spline3     0.9577     0.7068   1.355  0.17541
## b_spline4     1.9666     0.6685   2.942  0.00326 **
## b_spline5     0.8839     0.4749   1.861  0.06271 .
## b_spline6     4.1247     1.4180   2.909  0.00363 **
## b_spline7    -0.6081     3.5395  -0.172  0.86359
## b_spline8     7.8131     6.3520   1.230  0.21869
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
```

```
##      Null deviance: 596.11  on 461  degrees of freedom
```

```
## Residual deviance: 536.41  on 453  degrees of freedom
```

```
## AIC: 554.41
```

```
##
```

```
## Number of Fisher Scoring iterations: 6
```

```
summary(logistic_n)
```

```
##
## Call:
## glm(formula = chd ~ n_spline, family = "binomial", data = heart)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.6808  -0.9949  -0.5417   1.1911   1.9959
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.8451     0.2721  -6.781 1.19e-11 ***
## n_spline1      1.2904     0.6570   1.964 0.049536 *
## n_spline2      1.7463     0.6477   2.696 0.007020 **
## n_spline3      1.2176     0.4130   2.948 0.003193 **
## n_spline4      2.7821     0.8006   3.475 0.000511 ***
## n_spline5      3.2544     1.1073   2.939 0.003292 **
## n_spline6      3.4619     1.7030   2.033 0.042078 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 596.11  on 461  degrees of freedom
## Residual deviance: 538.76  on 455  degrees of freedom
## AIC: 552.76
##
## Number of Fisher Scoring iterations: 4
```

```
summary(logistic_poly)
```

```
##
## Call:
## glm(formula = chd ~ poly_spline, family = "binomial", data = heart)
##
```

```
## Deviance Residuals:
##      Min        1Q    Median        3Q        Max
## -1.4715   -0.9664   -0.5571    1.1216    2.2392
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.785e+00  2.754e-01  -6.480  9.2e-11 ***
## poly_splines  -8.557e+01  9.873e+01  -0.867   0.386
## poly_spline    2.385e+03  2.712e+03   0.880   0.379
## poly_spline   -2.003e+04  2.278e+04  -0.879   0.379
## poly_spline24%  2.004e+04  2.280e+04   0.879   0.379
## poly_spline32% -1.358e+01  2.074e+01  -0.655   0.513
## poly_spline48%  1.729e-01  6.612e-01   0.262   0.794
## poly_spline64% -1.211e-01  9.278e-02  -1.305   0.192
## poly_spline80%  3.744e-02  2.453e-02   1.526   0.127
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 596.11  on 461  degrees of freedom
## Residual deviance: 536.41  on 453  degrees of freedom
## AIC: 554.41
##
## Number of Fisher Scoring iterations: 6
```

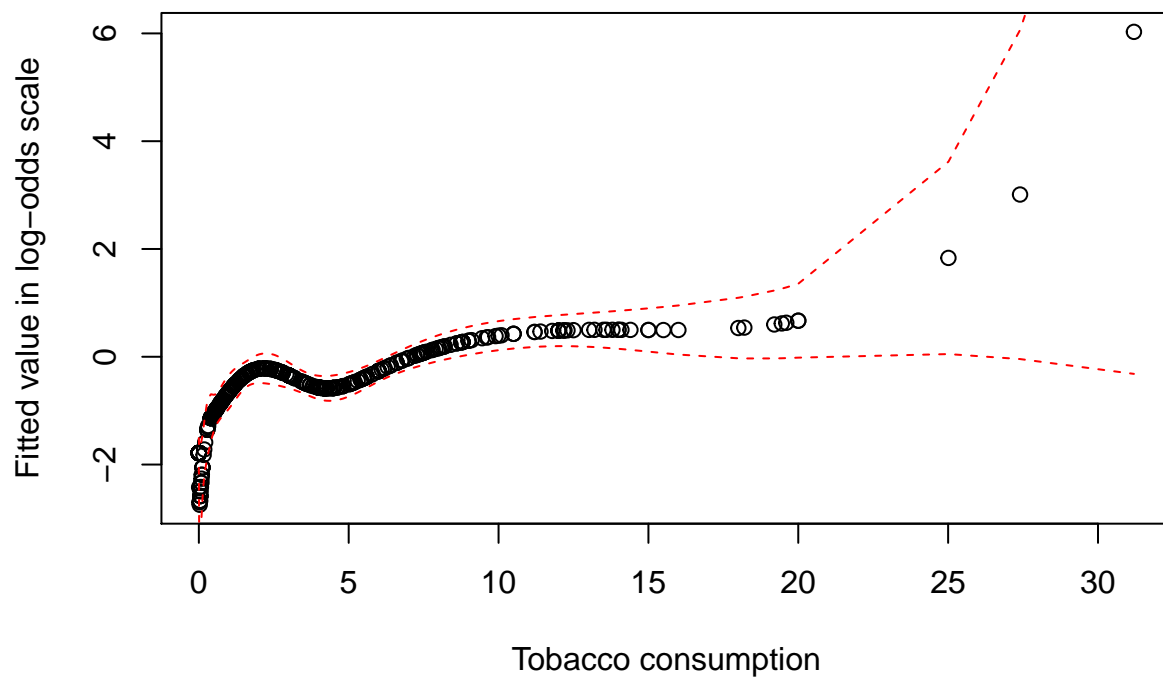
```
# Predict without using predict() function on log-odds scale
# b-spline
X_b <- model.matrix(logistic_b) # design matrix for b-spline
coef_b <- as.vector(logistic_b$coefficients) # coefficients vector for b-spline
Y_b <- X_b %*% coef_b # predicted value
var_Y_b <- diag(X_b %*% vcov(logistic_b) %*% t(X_b)) # predicted variance
se_Y_b <- as.matrix(sqrt(var_Y_b)) # predicted se
upper_b <- Y_b + se_Y_b # upper bound of the CI
lower_b <- Y_b - se_Y_b # lower bound of the CI
# Plot the predicted values
```

```

plot(heart$tobacco, Y_b,
     xlab = 'Tobacco consumption',
     ylab = 'Fitted value in log-odds scale',
     main = 'B-Spline prediction')
# Upper bound
lines(sort(heart$tobacco), upper_b[order(heart$tobacco)], col = "red", lty = 2)
# Lower bound
lines(sort(heart$tobacco), lower_b[order(heart$tobacco)], col = "red", lty = 2)

```

## B-Spline prediction



```

# natural spline
X_n <- model.matrix(logistic_n) # design matrix for b-spline
coef_n <- as.vector(logistic_n$coefficients) # coefficients vector for b-spline
Y_n <- X_n %*% coef_n # predicted value
var_Y_n <- diag(X_n %*% vcov(logistic_n) %*% t(X_n)) # predicted variance
se_Y_n <- as.matrix(sqrt(var_Y_n)) # predicted se
upper_n <- Y_n + se_Y_n # upper bound of the CI
lower_n <- Y_n - se_Y_n # lower bound of the CI

```

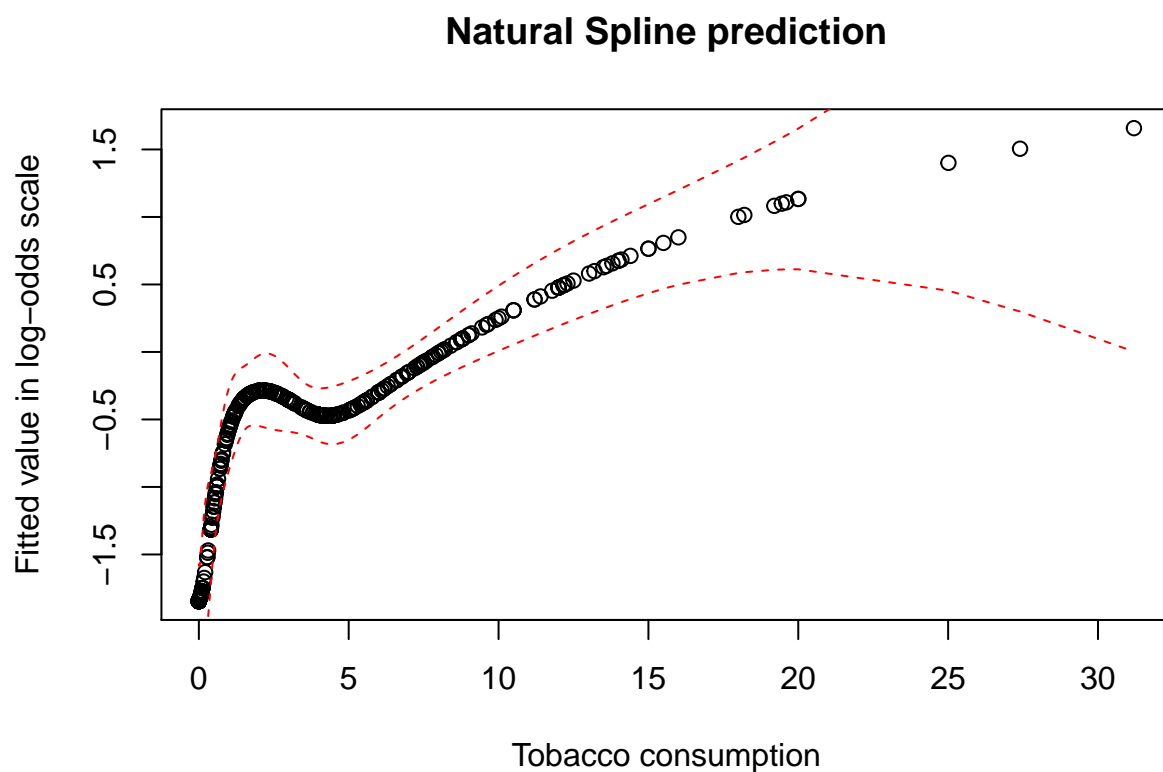
```

# Plot the predicted values
plot(heart$tobacco, Y_n,
     xlab = 'Tobacco consumption',
     ylab = 'Fitted value in log-odds scale',
     main = 'Natural Spline prediction')

# Upper bound
lines(sort(heart$tobacco), upper_n[order(heart$tobacco)], col = "red", lty = 2)

# Lower bound
lines(sort(heart$tobacco), lower_n[order(heart$tobacco)], col = "red", lty = 2)

```



```

# truncated polynomial spline
X_poly <- model.matrix(logistic_poly) # design matrix for b-spline
coef_poly <- as.vector(logistic_poly$coefficients) # coefficients vector for b-spline
Y_poly <- X_poly %*% coef_poly # predicted value
var_Y_poly <- diag(X_poly %*% vcov(logistic_poly) %*% t(X_poly)) # predicted variance
se_Y_poly <- as.matrix(sqrt(var_Y_poly)) # predicted se
upper_poly <- Y_poly + se_Y_poly # upper bound of the CI

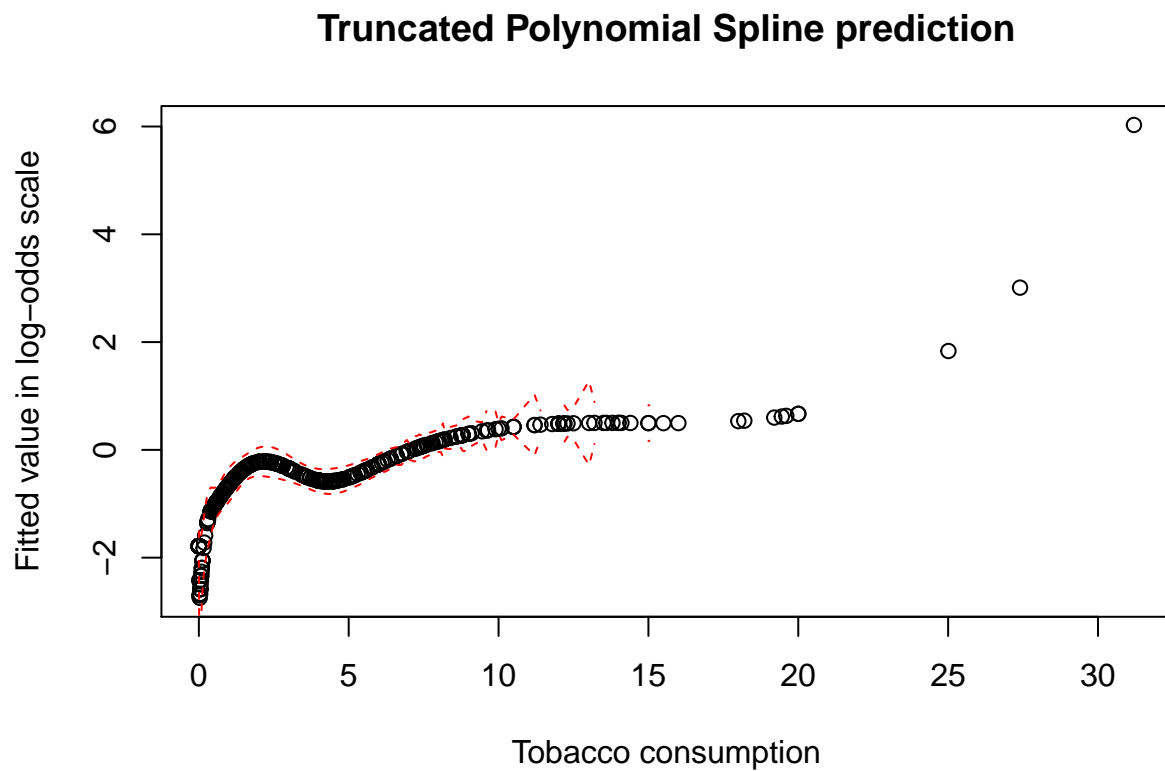
```



```

lower_poly <- Y_poly - se_Y_poly # lower bound of the CI
# Plot the predicted values
plot(heart$tobacco, Y_poly,
     xlab = 'Tobacco consumption',
     ylab = 'Fitted value in log-odds scale',
     main = 'Truncated Polynomial Spline prediction')
# Upper bound
lines(sort(heart$tobacco), upper_poly[order(heart$tobacco)], col = "red", lty = 2)
# Lower bound
lines(sort(heart$tobacco), lower_poly[order(heart$tobacco)], col = "red", lty = 2)

```



Question 3 (..)

Question 4

```

# Simulate data from a surface on the unit square with Gaussian noise
my_simulation <- function(n, noise_mean = 0, noise_sd = 1){
  # n is the number of observations required
  # noise_mean is the mean of the noise term, default is 0
  # noise_sd is the sd of the noise term, default is 1

  # draw numbers from unit square x-axis
  x <- runif(n)
  # draw numbers from unit square y-axis
  y <- runif(n)
  # noise term from Gaussian distribution
  noise <- rnorm(n, noise_mean, noise_sd)
  # create a smooth surface using a high-order bivariate polynomial
  z <- 5*x^7*y^6 + 2*x^6*y^5 + x^5 + 7*x^3*y^2 + 2*x^2*y + x*y^2 + y^3 + 3*y^4*x + y^8
  # smooth surface with gaussian noise
  z_noise <- z + noise
  # return a data frame with three columns: x, y, and z
  data.frame(x = x, y = y, z = z_noise)
}

# Example of the simulation
# Expect a list of 5 numbers that are generated using the polynomial above
# and with noises from a Gaussian distribution with mean 0.5 and sd 2.5.
my_simulation(5, 0.5, 2.5)$z

```

Part(a)

```
## [1] -1.9888952 10.9039110 2.6879659 0.2885358 11.1776966
```

```

# Import libraries
library(mgcv)

# Function below returns the true value of z, used to compare with the prediction

```

```

true_z <- function(x, y){
  5*x^7*y^6 + 2*x^6*y^5 + x^5 + 7*x^3*y^2 + 2*x^2*y + x*y^2 + y^3 + 3*y^4*x + y^8
}

# 250 simulations
simulation_num <- 250

# (i) method = "GCV.Cp".
# Create an empty matrices to store the computed value for each simulation
comp_time1 <- matrix(NA, nrow=simulation_num, ncol=1)
bias1 <- matrix(NA, nrow=simulation_num, ncol=1)
variance1 <- matrix(NA, nrow=simulation_num, ncol=1)
mse1 <- matrix(NA, nrow=simulation_num, ncol=1)

for (i in 1:simulation_num) {
  # Simulate 500 random samples for each simulation with standard gaussian noise
  my_data <- my_simulation(500, 0, 1)

  # Fit a GAM model using z ~ te(x,y) formula and method = "GCV.Cp"
  start_time <- Sys.time() # time before running model
  model1 <- gam(z ~ te(x, y), data = my_data, method = "GCV.Cp")
  end_time <- Sys.time() # time after running model
  comp_time1[i] <- as.numeric(end_time - start_time) # computation time

  # Create test set using runif()
  x_test <- runif(50)
  y_test <- runif(50)
  test_set <- as.data.frame(cbind(x = x_test, y = y_test))

  # Prediction
  pred <- predict(model1, newdata = test_set) # predicted z value
  true <- true_z(x_test, y_test) # true value of z for comparison

  # Compute bias, variance, and MSE of the predictions
  bias1[i] <- mean(pred - true)
  variance1[i] <- mean((pred - mean(pred))^2)
}

```

```

    mse1[i] <- mean((pred - true)^2)
  }

mean(comp_time1)

```

## Part (b)

```
## [1] 0.02021418
```

```
mean(bias1)
```

```
## [1] -0.004393151
```

```
mean(variance1)
```

```
## [1] 8.048547
```

```
mean(mse1)
```

```
## [1] 0.06052534
```

```

# (ii) method = "REML".
# Create an empty matrices to store the computed value for each simulation
comp_time2 <- matrix(NA, nrow=simulation_num, ncol=1)
bias2 <- matrix(NA, nrow=simulation_num, ncol=1)
variance2 <- matrix(NA, nrow=simulation_num, ncol=1)
mse2 <- matrix(NA, nrow=simulation_num, ncol=1)

for (i in 1:simulation_num) {
  # Simulate 500 random samples for each simulation with standard gaussian noise
  my_data <- my_simulation(500, 0, 1)

  # Fit a GAM model using z ~ te(x,y) formula and method = "REML"
  start_time <- Sys.time()
  model2 <- gam(z ~ te(x, y), data = my_data, method = "REML")
  end_time <- Sys.time()
  comp_time2[i] <- as.numeric(end_time - start_time) # computation time
}

```

```

# Create test set using runif()
x_test <- runif(50)
y_test <- runif(50)
test_set <- as.data.frame(cbind(x = x_test, y = y_test))

# Prediction
pred <- predict(model2, newdata = test_set) # predicted z value
true <- true_z(x_test, y_test) # true value of z for comparison

# Compute bias, variance, and MSE of the predictions
bias2[i] <- mean(pred - true)
variance2[i] <- mean((pred - mean(pred))^2)
mse2[i] <- mean((pred - true)^2)
}

mean(comp_time2)

## [1] 0.06095007

mean(bias2)

## [1] -0.00196205

mean(variance2)

## [1] 8.023825

mean(mse2)

## [1] 0.05420235

# Create a comparison table for two methods
compare <- matrix(c(mean(comp_time1), mean(bias1), mean(variance1), mean(mse1),
                    mean(comp_time2), mean(bias2), mean(variance2), mean(mse2)),
                  ncol = 2, nrow = 4)
colnames(compare) <- c("GCV.Cp", "REML")

```

```
rownames(compare) <- c("computation time", "bias", "variance", "mse")
compare
```

```
##                GCV.Cp      REML
## computation time 0.020214181 0.06095007
## bias            -0.004393151 -0.00196205
## variance        8.048547170  8.02382481
## mse            0.060525340  0.05420235
```

```
# Generally, both methods have similar computation time, bias, variance and mse.
# REML has slightly greater values in all outputs.
```

## Question 5

**ESL 5.4** The natural boundary conditions for natural cubic splines is that “the function is linear beyond the boundary knots”. When  $X < \xi_1$ ,  $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$ , to make it linear, we need  $\beta_2$  and  $\beta_3$  equal to 0. Alternatively, we can prove by taking the second derivative of  $f(x)$  and set it to 0.  $f''(x) = 2\beta_2 + 6\beta_3 x = 0$ , hence  $\beta_2 = \beta_3 = 0$ .

Similarly, the function  $f(x)$  should be linear when  $X > \xi_K$  where  $f(x) = \beta_0 + \beta_1 x + \sum_{k=1}^K \theta_k (X - \xi_k)^3$ , we take second derivative and set it to 0.  $f''(x) = 6 \sum_{k=1}^K \theta_k (X - \xi_k) = 6(\sum_{k=1}^K \theta_k X - \sum_{k=1}^K \theta_k \xi_k) = 0$ , then we have  $\sum_{k=1}^K \theta_k X - \sum_{k=1}^K \theta_k \xi_k = 0$ , which implies that  $\sum_{k=1}^K \theta_k = 0$  and  $\sum_{k=1}^K \theta_k \xi_k = 0$ , as required.

Now we derive (5.4) and (5.5). The function we have is  $f(x) = \beta_0 + \beta_1 x + \sum_{k=1}^K \theta_k (X - \xi_k)_+^3 = 0$ , by observing the function, we get that  $N_1(X) = 1, N_2(X) = X$ , we now want to prove that  $\sum_{k=1}^K \theta_k (X - \xi_k)_+^3$  can be written in the form of  $N_{k+2}(X) = d_k(X) - d_{k-1}(X)$ .

Given that  $\sum_{k=1}^K \theta_k = 0$  and  $\sum_{k=1}^K \theta_k \xi_k = 0$ , we know that  $\sum_{k=1}^{K-2} \theta_k = -\theta_K - \theta_{K-1}$  and  $\sum_{k=1}^{K-2} \theta_k \xi_k = -\theta_K \xi_K - \theta_{K-1} \xi_{K-1}$ .

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

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

Similarly,

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

Put them back together and get:

$$\begin{aligned}
\sum_{k=1}^K \theta_k (X - \xi_k)_+^3 &= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \theta_{K-1} (X - \xi_{K-1})_+^3 + \theta_K (X - \xi_K)_+^3 \\
&= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \frac{(X - \xi_{K-1})_+^3}{\xi_{K-1} - \xi_K} + \sum_{k=1}^{K-2} \theta_k (\xi_k - \xi_{K-1}) \frac{(X - \xi_K)_+^3}{\xi_{K-1} - \xi_K} \\
&= \sum_{k=1}^{K-2} \theta_k \left[ (X - \xi_k)_+^3 + (\xi_K - \xi_k) \frac{(X - \xi_{K-1})_+^3}{\xi_{K-1} - \xi_K} + (\xi_k - \xi_{K-1}) \frac{(X - \xi_K)_+^3}{\xi_{K-1} - \xi_K} \right] \\
&= \sum_{k=1}^{K-2} \theta_k \left[ (X - \xi_k)_+^3 \frac{\xi_K - \xi_k}{\xi_K - \xi_k} + (\xi_K - \xi_k) \frac{(X - \xi_{K-1})_+^3}{\xi_{K-1} - \xi_K} + (\xi_k - \xi_{K-1}) \frac{(X - \xi_K)_+^3}{\xi_{K-1} - \xi_K} \frac{\xi_K - \xi_k}{\xi_K - \xi_k} \right] \\
&= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \left[ \frac{(X - \xi_k)_+^3}{\xi_K - \xi_k} + \frac{(X - \xi_{K-1})_+^3}{\xi_{K-1} - \xi_K} + (\xi_k - \xi_{K-1}) \frac{(X - \xi_K)_+^3}{(\xi_{K-1} - \xi_K)(\xi_K - \xi_k)} \right] \\
&= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \left[ \frac{(X - \xi_k)_+^3}{\xi_K - \xi_k} + \frac{(X - \xi_{K-1})_+^3}{\xi_{K-1} - \xi_K} + (X - \xi_K)_+^3 \left( \frac{-1}{\xi_{K-1} - \xi_K} - \frac{1}{\xi_K - \xi_k} \right) \right] \\
&= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \left[ \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k} + \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_{K-1} - \xi_K} \right] \\
&= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \left[ \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k} - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}} \right] \\
&= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) (d_k(X) - d_{K-1}(X)) \\
&= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) N_{k+2}(X)
\end{aligned}$$

Therefore, to conclude, we get  $N_1(X) = 1, N_2(X) = X, N_{k+2}(X) = d_k(X) - d_{K-1}(X)$  as desired in (5.4) and (5.5).

### ESL 5.13