STATS 790
Assignment #3

Dean Hansen - 400027416

03 March, 2023

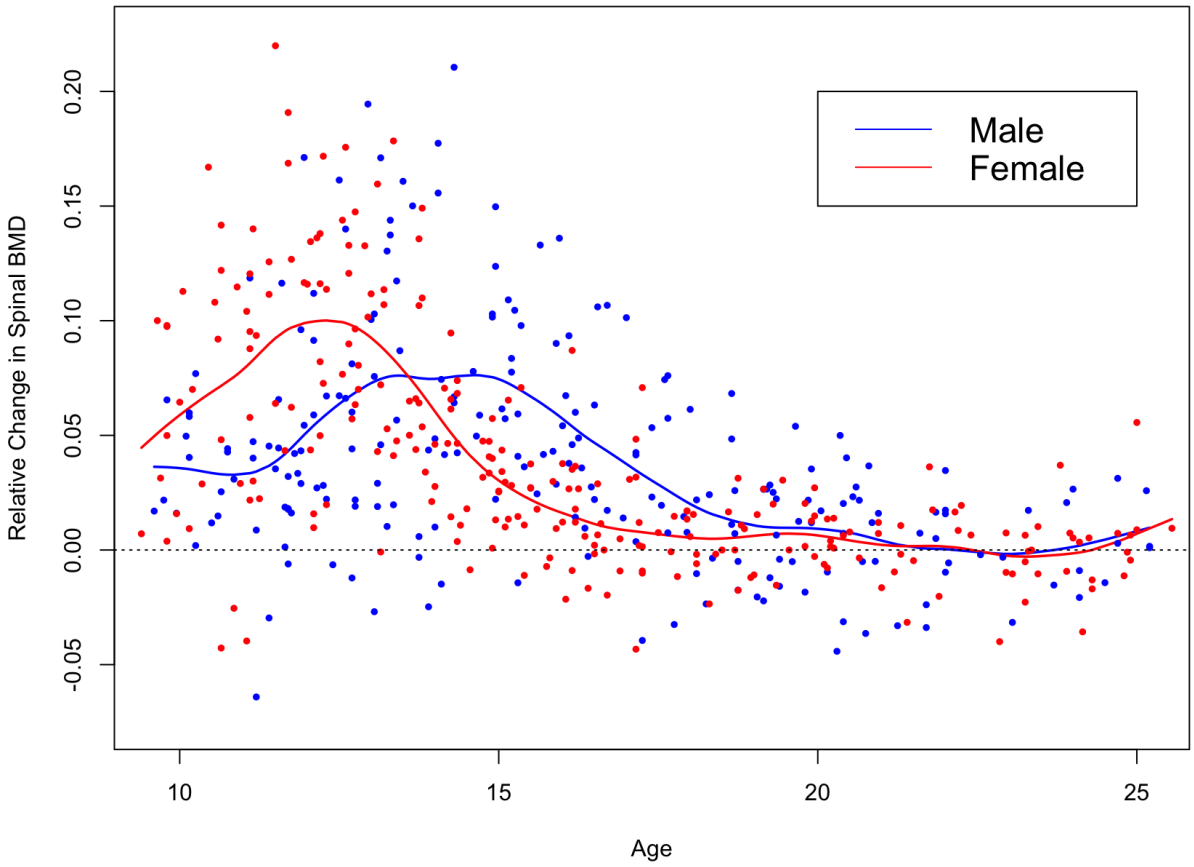
Contents

Question #1 2
Question #2 4
Question #3 8
Question #4 10
Question #5 (ESL 5.4) 12
Question #6 (ESL 5.13) 16

Question #1

Here we replicate Figure 5.6 from ESL.

```
#download dataset
url <- "https://hastie.su.domains/ElemStatLearn/datasets/bone.data"
dd_bmd <- read.delim(url)

#create male and female datasets
males <- dd_bmd %>% filter(gender == "male") %>% select(age, spnbmd)
females <- dd_bmd %>% filter(gender == "female") %>% select(age, spnbmd)

#smooth splines
males_spline <- smooth.spline(x = males$age, y = males$spnbmd, df = 12)
females_spline <- smooth.spline(x = females$age, y = females$spnbmd, df = 12)

#code to generate figure
plot(males_spline,ylim=c(-0.075,0.225),col="blue",type="l",
     lwd=1.75,xlab="",ylab="")
lines(females_spline,col="red",lwd=1.75)
points(x=males$age,y=males$spnbmd,col="blue",pch=20,lwd=0.001)
points(x=females$age,y=females$spnbmd,col="red",pch=20,lwd=0.001)
abline(h=0,lty=3)
title(xlab="Age",ylab="Relative Change in Spinal BMD")
legend(x=c(20,25),y=c(0.15,0.20),legend=c("Male","Female"),
       col=c("blue","red"),lty=1,cex=1.5)
```


Male
Female
Relative Change in Spinal BMD
-0.05
0.00
0.05
0.10
0.15
0.20
10
15
20
25
Age

Question #2

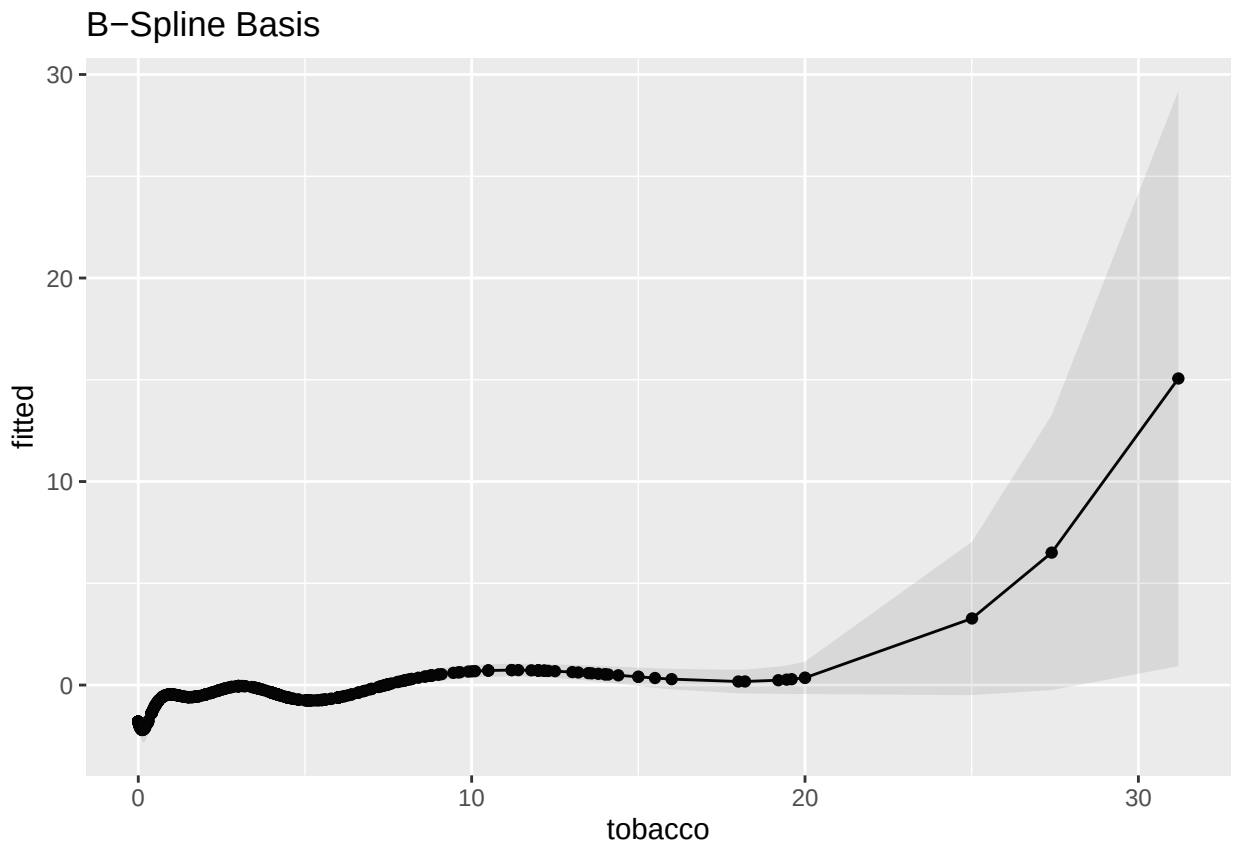
Here we compute a logistic regression model using the b-spline, natural spline and truncated power basis functions.

Custom knots were defined prior to fitting each model as the splines package defaults would produce NA values in the covariance matrix and the truncated power basis function did as well unless the lowest knot was moved to 0.30. Thus, all knots have been placed at the same quantiles so the basis can be compared easily.

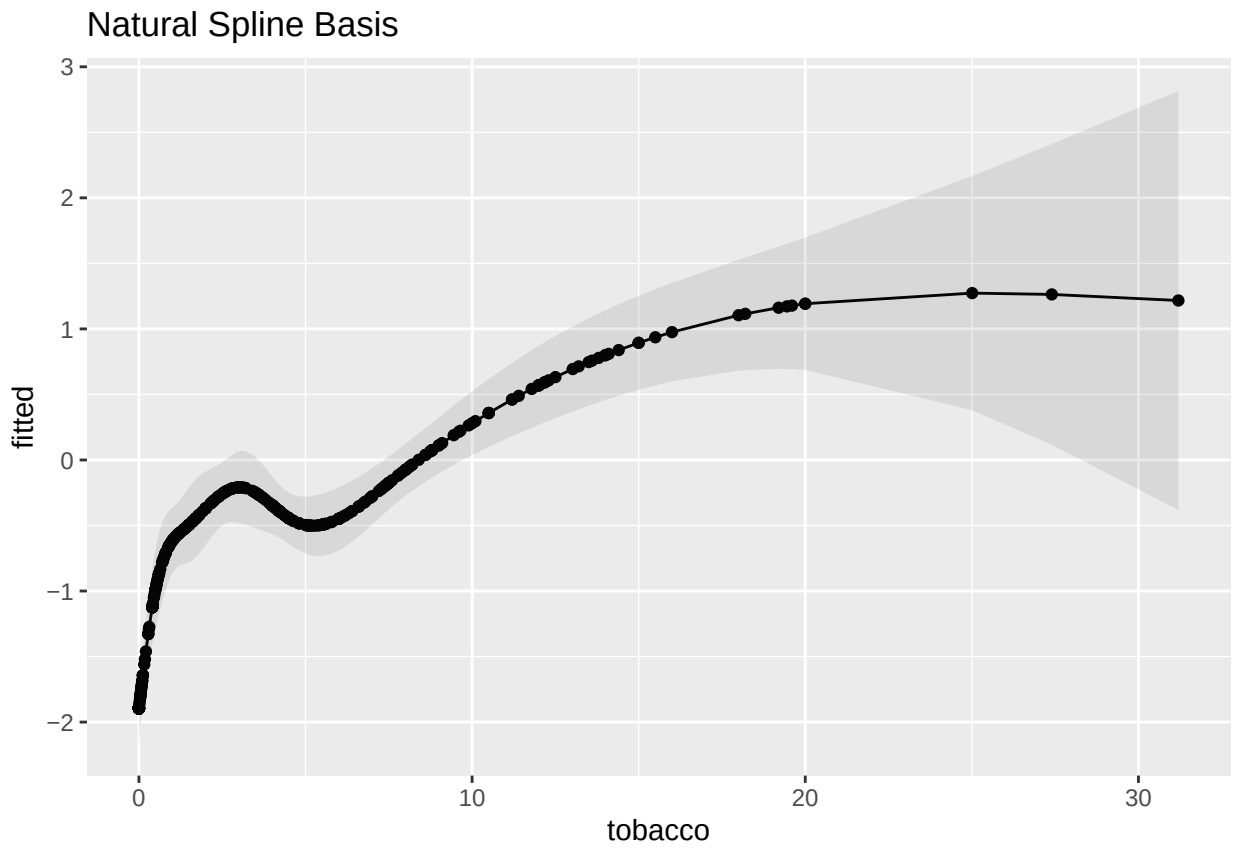
```
#download dataset (borrowed from class notes)
url <- "http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data"
fn <- "SAheart.txt"
if (!file.exists(fn)) download.file(url, destfile = fn)
raw_dd <- read.csv(fn, row.names = 1)
dd <- raw_dd %>% select(tobacco, chd)

#b-spline basis
b_spline_fit <- gam(chd ~ bs(x = tobacco, knots = quantile(tobacco, seq(0.30,0.85,length=5))), family = binomial, data = dd)
b_spline_X <- model.matrix(b_spline_fit)
vcov_b <- vcov(b_spline_fit)

#natural spline basis
natural_spline_fit <- gam(chd ~ ns(x = tobacco, knots = quantile(tobacco, seq(0.30,0.85,length=5))), family = binomial, data = dd)
natural_spline_X <- model.matrix(natural_spline_fit)
vcov_natural <- vcov(natural_spline_fit)

#truncated power basis
q2 <- function(x, nknots) {
  knots <- quantile(x, seq(0.30, 0.85, length = nknots))
  trunc_fun <- function(k) (x>=k)*(x-k)^3
  s <- sapply(knots, trunc_fun)
  s <- cbind(x, x^2, x^3, s)
  return(s)
}

truncated_fit <- gam(chd ~ q2(x = tobacco, nknots = 5), family = binomial, data = dd)
truncated_X <- model.matrix(truncated_fit)
vcov_truncated <- vcov(truncated_fit)
```

Plots of the model predictions +/- one standard error on log-odds scale.

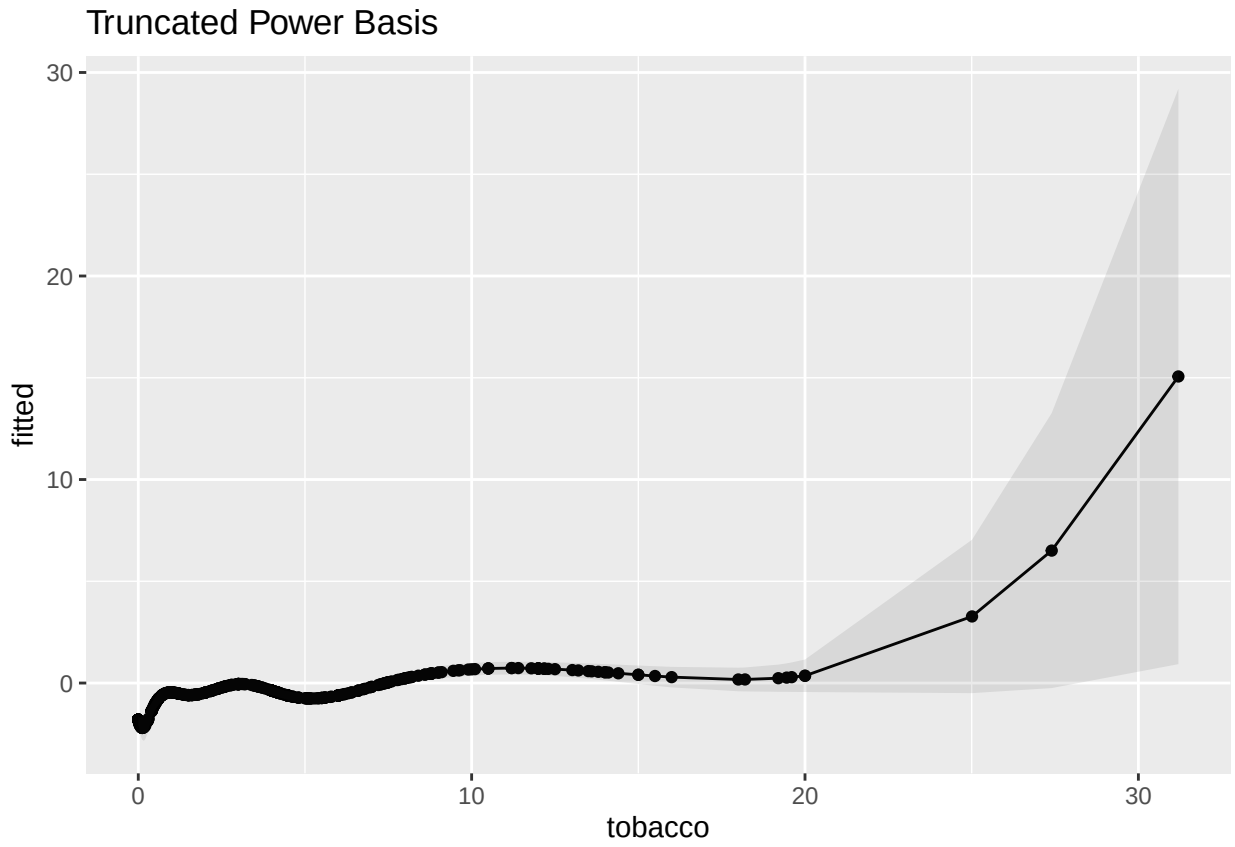
```
#b-spline basis
b_spline_se <- b_spline_X %*% vcov_b %*% t(b_spline_X)
b_spline_se <- sqrt(diag(b_spline_se))
b_spline_plot <- data.frame(tobacco = dd$tobacco,
                            fitted = b_spline_X %*% coef(b_spline_fit),
                            se = b_spline_se)

ggplot(b_spline_plot, aes(x = tobacco, y = fitted)) +
  geom_line() +
  geom_point() +
  geom_ribbon(aes(ymin = fitted - se, ymax = fitted + se), alpha = 0.1) +
  labs(title = "B-Spline Basis")
```



```r
#natural spline basis
natural_spline_se <- natural_spline_X %*% vcov_natural %*% t(natural_spline_X)
natural_spline_se <- sqrt(diag(natural_spline_se))
natural_spline_plot <- data.frame(tobacco = dd$tobacco,
                                  fitted = natural_spline_X %*% coef(natural_spline_fit),
                                  se = natural_spline_se)

ggplot(natural_spline_plot, aes(x = tobacco, y = fitted)) +
  geom_line() +
  geom_point() +
  geom_ribbon(aes(ymin = fitted - se, ymax = fitted + se), alpha = 0.1) +
  labs(title = "Natural Spline Basis")
```



```r
#truncated power basis
truncated_se <- truncated_X %*% vcov_truncated %*% t(truncated_X)
truncated_se <- sqrt(diag(truncated_se))
truncated_plot <- data.frame(tobacco = dd$tobacco,
                             fitted = truncated_X %*% coef(truncated_fit),
                             se = truncated_se)

ggplot(truncated_plot, aes(x = tobacco, y = fitted)) +
  geom_line() +
  geom_point() +
  geom_ribbon(aes(ymin = fitted - se, ymax = fitted + se), alpha = 0.1) +
  labs(title = "Truncated Power Basis")
```



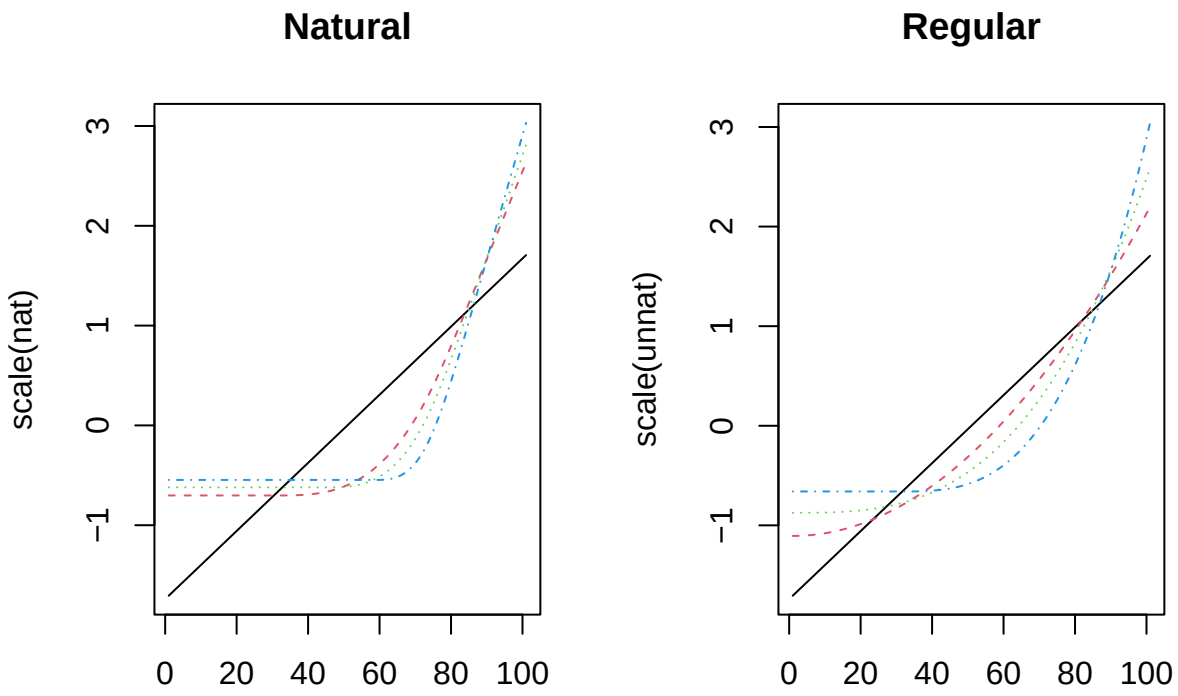
Question #3

Below we create a function that implements the truncated power basis and natural spline basis.

```
truncpolyspline <- function(x, df, natural = c(TRUE, FALSE)) {
  if (!require("Matrix")) stop("need Matrix package")

  #truncated power basis
  if(!natural) {
    knots <- quantile(x, seq(0.30, 0.85, length = df - 4))

    trunc_fun <- function(k) (x>=k)*(x-k)^3

    s <- sapply(knots, trunc_fun)
    s <- cbind(x, x^2, x^3, s)
    return(s)
  }

  #natural spline basis
  if(natural) {
    knots <- quantile(x, seq(0.30, 0.85, length = df))

    trunc_fun <- function(k, K, d_K_1) {
      ((x>=k)*(x-k)^3 - (x>=K)*(x-K)^3)/(K-k) - d_K_1
    }

    K <- as.numeric(knots[length(knots)])
    K_1 <- as.numeric(knots[length(knots)-1])
    d_K_1 <- ((x>=K_1)*(x-K_1)^3 - (x>=K)*(x-K)^3)/(K-K_1)

    s <- sapply(knots[1:df],trunc_fun,K=K, d_K_1=d_K_1)
    s <- cbind(x, s)
    return(s)
  }
}
```

Below we plot the basis functions for $df = 5$ as in the notes.

```
xvec <- seq(0, 10, length = 101)
nat <- truncpolyspline(xvec, df = 5, natural = TRUE)
unnat <- truncpolyspline(xvec, df = 5, natural = FALSE)
par(mfrow = c(1, 2))
matplot(scale(nat), type = "l", main = "Natural")
matplot(scale(unnat), type = "l", main = "Regular")
```



Question #4

a) Here we create a function that simulates n points plus Gaussian noise from the function $f(x, y) = \pi + \exp(-(x^2 + 2xy + y^2))$.

```
#function over unit square
my_function <- function(x,y) {pi + exp(-(x^2 + x*y + y^2))}

#get values with noise
sim_values <- function(n = 1000) {
  x <- runif(n)
  y <- runif(n)
  eps <- rnorm(n)
  z <- my_function(x,y) + eps
  return(data.frame(x=x,y=y,z=z))
}
```

b) Below we run an ensemble of 250 simulations and print the results for average time, bias, variance and mean-squared error for "GCV.cp" and "REML".

```
simulation <- function(k) {

#allocate exact space...
  gcv_time <- rep(0, length=k)
  gcv_bias <- rep(0, length=k)
  gcv_var <- rep(0, length=k)
  gcv_mse <- rep(0, length=k)

  reml_time <- rep(0, length=k)
  reml_bias <- rep(0, length=k)
  reml_var <- rep(0, length=k)
  reml_mse <- rep(0, length=k)

#run inner for loop k times
  for (i in 1:k) {

  #simulate data
    data <- sim_values()

  #GCV fit
    gcv_start <- Sys.time()
    gcv <- mgcv::gam(z ~ te(x,y), data = data, method = "GCV.Cp")
    gcv_end <- Sys.time()
    gcv_pred <- predict(gcv, newdata = data, type = "reponse")

  #store GCV values
    gcv_time[i] <- gcv_end - gcv_start
    gcv_bias[i] <- mean(gcv_pred - data$z)
    gcv_var[i] <- var(gcv_pred - data$z)
    gcv_mse[i] <- mean((gcv_pred - data$z)^2)

  #REML fit
    reml_start <- Sys.time()
    reml <- mgcv::gam(z ~ te(x,y), data = data, method = "REML")
    reml_end <- Sys.time()
    reml_pred <- predict(reml, newdata = data, type = "reponse")
```

```
  #store REML values
    reml_time[i] <- reml_end - reml_start
    reml_bias[i] <- mean(reml_pred - data$z)
    reml_var[i] <- var(reml_pred - data$z)
    reml_mse[i] <- mean((reml_pred - data$z)^2)
  }

#GCV
  avg_gcv_time <- mean(gcv_time)
  avg_gcv_bias <- mean(gcv_bias)
  avg_gcv_var <- mean(gcv_var)
  avg_gcv_mse <- mean(gcv_mse)

#REML
  avg_reml_time <- mean(reml_time)
  avg_reml_bias <- mean(reml_bias)
  avg_reml_var <- mean(reml_var)
  avg_reml_mse <- mean(reml_mse)

#return full set of results in table
  results <- data.frame(method = c("GCV.Cp", "REML"),
                        time = c(avg_gcv_time, avg_reml_time),
                        bias = c(avg_gcv_bias, avg_reml_bias),
                        var = c(avg_gcv_var, avg_reml_var),
                        mse = c(avg_gcv_mse, avg_reml_mse))
  return(results)
}

#run for 250 simulations and print results
simulation(k = 250)
```

```
##   method       time     bias       var      mse
## 1 GCV.Cp 0.02495406 -3.61634 0.9939511 14.07192
## 2   REML 0.09795400 -3.61634 0.9957404 14.07370
```

From the results, we see the GCV.cp is faster than the REML method, and both methods provide similar average bias, variance and mean-squared error values.

Question #5 (ESL 5.4)

Consider the truncated power series representation for cubic splines with K interior knots:

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$$

The natural boundary conditions say that outside the boundary knots, the function must be linear. For $X < \xi_1$, we have $\sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 = 0$ and $f(X) = \sum_{j=0}^{3} \beta_j X^j = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$. For this function to be linear when $X < \xi_1$, we have $\beta_2 = \beta_3 = 0$. Similarly for $X \geq \xi_k$, we have $f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$. Using $(X - \xi_k)_+^3 = X^3 - 3\xi_k X^2 + 3\xi_k^2 X - \xi_k^3$, we can write $f(X)$ in the following way:

$$\begin{aligned}
f(X) &= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 && (1)\\
&= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k [X^3 - 3\xi_k X^2 + 3\xi_k^2 X - \xi_k^3] && (2)\\
&= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K} \theta_k X^3 - 3 \sum_{k=1}^{K} \theta_k \xi_k X^2 + 3 \sum_{k=1}^{K} \theta_k \xi_k^2 X - \sum_{k=1}^{K} \theta_k \xi_k^3 && (3)\\
&= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K} \theta_k X^3 - 3 \sum_{k=1}^{K} \theta_k \xi_k X^2 + 3 \sum_{k=1}^{K} \theta_k \xi_k^2 X - \sum_{k=1}^{K} \theta_k \xi_k^3 && (4)\\
&= \beta_0 - \sum_{k=1}^{K} \theta_k \xi_k^3 + X(\beta_1 + 3 \sum_{k=1}^{K} \theta_k \xi_k^2) + X^2(\beta_2 - 3 \sum_{k=1}^{K} \theta_k \xi_k) + X^3(\beta_3 + \sum_{k=1}^{K} \theta_k) && (5)
\end{aligned}$$

For this function to be linear, we require $\beta_2 - 3\sum_{k=1}^{K} \theta_k \xi_k = 0$ and $\beta_3 + \sum_{k=1}^{K} \theta_k = 0$. From before we have $\beta_2 = \beta_3 = 0$, which implies $\sum_{k=1}^{K} \theta_k \xi_k = 0$ and $\sum_{k=1}^{K} \theta_k = 0$ as required.

The natural cubic spline basis is of the form $N_1(X) = 1$, $N_2(X) = X$ and $N_{k+2}(X) = d_k(X) - d_{K-1}(X)$ where $d_k(X) = \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k}$. To derive this basis from the truncated power series representation, we show that $f(X)$ can be written as a linear combination of the $N_k(X)$ basis functions such that $f(X) = \sum_{k=1}^{K} \alpha_k N_k(X)$ for some α_k's.

For $X < \xi_1$, we have $(X - \xi_k)_+^3 = 0$ for all k which means $N_{k+2}(X) = 0$ and

$$f(X) = \sum_{k=1}^{K} \alpha_k N_k(X) = \alpha_1 N_1(X) + \alpha_2 N_2(X) = \alpha_1 + \alpha_2 X$$

Thus, we have $\alpha_1 = \beta_0$ and $\alpha_2 = \beta_1$.

For $X \geq \xi_K$ and $K - 2 \geq k$, we have the following:

$$N_{k+2}(X) = d_k(X) - d_{K-1}(X) \tag{6}$$

$$= \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k} - \frac{(X-\xi_{K-1})_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_{K-1}} \tag{7}$$

$$= \frac{X^3 - 3\xi_k X^2 + 3\xi_k^2 X - \xi_k^3 - X^3 + 3\xi_K X^2 - 3\xi_K^2 X + \xi_K^3}{\xi_K - \xi_k} - \frac{X^3 - 3\xi_{K-1} X^2 + 3\xi_{K-1}^2 X - \xi_{K-1}^3 - X^3 + 3\xi_K X^2 - 3\xi_K^2 X + \xi_K^3}{\xi_K - \xi_{K-1}} \tag{8}$$

$$= \frac{(-3\xi_k + 3\xi_K)X^2 + (3\xi_k^2 - 3\xi_K^2)X + (\xi_K^3 - \xi_k^3)}{\xi_K - \xi_k} - \frac{(-3\xi_{K-1} + 3\xi_K)X^2 + (3\xi_{K-1}^2 - 3\xi_K^2)X + (\xi_K^3 - \xi_{K-1}^3)}{\xi_K - \xi_{K-1}} \tag{9}$$

$$= \frac{3(\xi_K - \xi_k)X^2 + 3(\xi_k^2 - \xi_K^2)X + (\xi_K^3 - \xi_k^3)}{\xi_K - \xi_k} - \frac{3(\xi_K - \xi_{K-1})X^2 + 3(\xi_{K-1}^2 - \xi_K^2)X + (\xi_K^3 - \xi_{K-1}^3)}{\xi_K - \xi_{K-1}} \tag{10}$$

$$= 3X^2 - 3X^2 + \frac{3(\xi_k^2 - \xi_K^2)X + (\xi_K^3 - \xi_k^3)}{\xi_K - \xi_k} - \frac{3(\xi_{K-1}^2 - \xi_K^2)X + (\xi_K^3 - \xi_{K-1}^3)}{\xi_K - \xi_{K-1}} \tag{11}$$

$$= \frac{3(\xi_k + \xi_K)(\xi_k - \xi_K)X + (\xi_K - \xi_k)(\xi_K^2 + 2\xi_K\xi_k + \xi_k^2)}{\xi_K - \xi_k} - \frac{3(\xi_{K-1} + \xi_K)(\xi_{K-1} - \xi_K)X + (\xi_K - \xi_{K-1})(\xi_K^2 + 2\xi_K\xi_{K-1} + \xi_{K-1}^2)}{\xi_K - \xi_{K-1}} \tag{12}$$

$$= \frac{-3(\xi_k + \xi_K)(\xi_K - \xi_k)X}{\xi_K - \xi_k} + \frac{(\xi_K - \xi_k)(\xi_K^2 + 2\xi_K\xi_k + \xi_k^2)}{\xi_K - \xi_k} + \frac{3(\xi_{K-1} + \xi_K)(\xi_K - \xi_{K-1})X}{\xi_K - \xi_{K-1}} - \frac{(\xi_K - \xi_{K-1})(\xi_K^2 + 2\xi_K\xi_{K-1} + \xi_{K-1}^2)}{\xi_K - \xi_{K-1}} \tag{13}$$

$$= -3(\xi_k + \xi_K)X + (\xi_K^2 + 2\xi_K\xi_k + \xi_k^2) + 3(\xi_{K-1} + \xi_K)X - (\xi_K^2 + 2\xi_K\xi_{K-1} + \xi_{K-1}^2) \tag{14}$$

$$= (2\xi_K\xi_k + \xi_k^2 - 2\xi_K\xi_{K-1} - \xi_{K-1}^2) + 3((\xi_{K-1} + \xi_K) - (\xi_k + \xi_K))X \tag{15}$$

$$= b + mX \tag{16}$$

This shows that $f(X)$ is linear past the last boundary knot as required.

Before showing the recursion relation holds for the basis, we need two identities. The first identity is from the first section of the proof:

$$\sum_{k=1}^{K} \theta_k = 0 \tag{17}$$

$$\implies \sum_{k=1}^{K-1} \theta_k = -\theta_K \tag{18}$$

$$\implies \sum_{k=1}^{K-2} \theta_k = -\theta_K - \theta_{K-1} \tag{19}$$

The second identity follows in the same way:

$$\sum_{k=1}^{K} \xi_k \theta_k = 0 \tag{20}$$

$$\implies \sum_{k=1}^{K-2} \xi_k \theta_k = -\xi_K \theta_K - \xi_{K-1} \theta_{K-1} \tag{21}$$

We know from the basis that for the $N_{k+2}(X)$ coefficients, the remaining term should be $\sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$. For this to happen, we need to multiply the coefficients by a factor of $(\xi_K - \xi_k)$. To see this, we now derive the last part of the basis as follows:

$$\sum_{k=1}^{K-2} \alpha_{k+2} N_{k+2}(X) = \sum_{k=1}^{K-2} (\xi_K - \xi_k) \theta_k N_{k+2}(X) \tag{22}$$

$$= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) [\frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k} - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}}] \tag{23}$$

$$= \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) [\frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k} - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}}] \tag{24}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \sum_{k=1}^{K-2} \theta_k (X - \xi_K)_+^3 - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}} \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \tag{25}$$

Using the derived identities from above we get:

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \sum_{k=1}^{K-2} \theta_k (X - \xi_K)_+^3 - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}} (\xi_K \sum_{k=1}^{K-2} \theta_k - \sum_{k=1}^{K-2} \theta_k \xi_k) \tag{26}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \sum_{k=1}^{K-2} \theta_k (X - \xi_K)_+^3 - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}} (-\theta_K \xi_K - \theta_{K-1} \xi_K + \theta_K \xi_K + \theta_{K-1} \xi_{K-1}) \tag{27}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \sum_{k=1}^{K-2} \theta_k (X - \xi_K)_+^3 - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}} (-\theta_{K-1} \xi_K + \theta_{K-1} \xi_{K-1}) \tag{28}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \sum_{k=1}^{K-2} \theta_k (X - \xi_K)_+^3 + \theta_{K-1} \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_{K-1}} (\xi_K - \xi_{K-1}) \tag{29}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - (X - \xi_K)_+^3 \sum_{k=1}^{K-2} \theta_k + \theta_{K-1} ((X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3) \tag{30}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - (X - \xi_K)_+^3 (-\theta_K - \theta_{K-1}) + \theta_{K-1} ((X - \xi_{K-1})_+^3 - (X - \xi_K)_+^3) \tag{31}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \theta_K (X - \xi_K)_+^3 + \theta_{K-1} (X - \xi_K)_+^3 + \theta_{K-1} (X - \xi_{K-1})_+^3 - \theta_{K-1} (X - \xi_K)_+^3 \tag{32}$$

$$= \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \theta_K (X - \xi_K)_+^3 + \theta_{K-1} (X - \xi_{K-1})_+^3 \tag{33}$$

$$= \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 \tag{34}$$

Thus, $\alpha_{k+2} = (\xi_K - \xi_k)\theta_k$ for k from 1 to $K-2$. We have shown the two basis are equivalent and that $f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$ can be written as $f(X) = \sum_{k=1}^{K} \alpha_k N_k(X)$ for above α_k.

Question #6 (ESL 5.13)

Suppose we fit the smoothing spline $\hat{f}_\lambda = S_\lambda y$ where $S_\lambda = N(N^T N + \lambda \Omega_N)^{-1} N^T$. We will use the notation $\hat{f}^{(-i)}(x_i)$ to denote the prediction made at x_i using a smoothing spline fitted to data that excludes the i^{th} observation. When performing LOOCV, we remove each point and re-fit the model. In vector notation, we can write this out as $y' = y + (\hat{f}_\lambda^{(-i)}(x_i) - y_i)e_i$, where y is the original vector of responses. The vector e_i is the standard basis vector in $\mathbb{R}^N$, and in this representation we replace the i^{th} observation of y with the predicted value $\hat{f}_\lambda^{(-i)}(x_i)$. The fitted values for the response vector y' are given by:

$$\begin{aligned}
\hat{f}_\lambda^{(-i)} &= S_\lambda y' && (35)\\
&= S_\lambda(y + (\hat{f}_\lambda^{(-i)}(x_i) - y_i)e_i) && (36)\\
&= S_\lambda y + S_\lambda(\hat{f}_\lambda^{(-i)}(x_i) - y_i)e_i && (37)\\
&= \hat{f}_\lambda + S_\lambda(\hat{f}_\lambda^{(-i)}(x_i) - y_i)e_i && (38)
\end{aligned}$$

The term $S_\lambda y$ is equal to fitting the spline using all of the data, which is just $\hat{f}_\lambda$ from above. To get the coordinate for x_i, we can take the dot product with e_i^T as follows:

$$\begin{aligned}
e_i^T \hat{f}_\lambda^{(-i)} &= e_i^T \hat{f}_\lambda + e_i^T S_\lambda(\hat{f}^{(-i)}(x_i) - y_i)e_i && (39)\\
\hat{f}_\lambda^{(-i)}(x_i) &= \hat{f}_\lambda(x_i) + (\hat{f}^{(-i)}(x_i) - y_i)e_i^T S_\lambda e_i && (40)\\
\hat{f}_\lambda^{(-i)}(x_i) &= \hat{f}_\lambda(x_i) + (\hat{f}^{(-i)}(x_i) - y_i)S_\lambda(i,i) && (41)
\end{aligned}$$

We have $e_i^T \hat{f}_\lambda^{(-i)} = \hat{f}_\lambda^{(-i)}(x_i)$ since we are picking out the i^{th} coordinate of the fitted values for y'. Further, $e_i^T \hat{f}_\lambda = \hat{f}_\lambda(x_i)$, which is the predicted value for x_i from the full data model. Note that the term $(\hat{f}^{(-i)}(x_i) - y_i)$ is a scalar and the term $S_\lambda(i,i) = e_i^T S_\lambda e_i$ denotes the i^{th} row and column of the smoothing matrix (notation from ESL).

To get the form of equation (5.26), we multiply both sides by -1 and add y_i to both sides:

$$y_i - \hat{f}_\lambda^{(-i)}(x_i) = y_i - \hat{f}_\lambda(x_i) - (\hat{f}^{(-i)}(x_i) - y_i)S_\lambda(i,i) \tag{42}$$

$$y_i - \hat{f}_\lambda^{(-i)}(x_i) + (\hat{f}^{(-i)}(x_i) - y_i)S_\lambda(i,i) = y_i - \hat{f}_\lambda(x_i) \tag{43}$$

$$y_i - \hat{f}_\lambda^{(-i)}(x_i) + \hat{f}^{(-i)}(x_i)S_\lambda(i,i) - y_i S_\lambda(i,i) = y_i - \hat{f}_\lambda(x_i) \tag{44}$$

$$y_i(1 - S_\lambda(i,i)) + \hat{f}_\lambda^{(-i)}(x_i)(S_\lambda(i,i) - 1) = y_i - \hat{f}_\lambda(x_i) \tag{45}$$

$$(1 - S_\lambda(i,i))(y_i - \hat{f}_\lambda^{(-i)}(x_i)) = y_i - \hat{f}_\lambda(x_i) \tag{46}$$

$$y_i - \hat{f}_\lambda^{(-i)}(x_i) = \frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)} \tag{47}$$

Thus, we have the N-fold cross validation formula as required below:

$$CV(\hat{f}_\lambda) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2 \tag{48}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)})^2 \tag{49}$$