Stats_790_Assignment3

Amandeep Sidhu

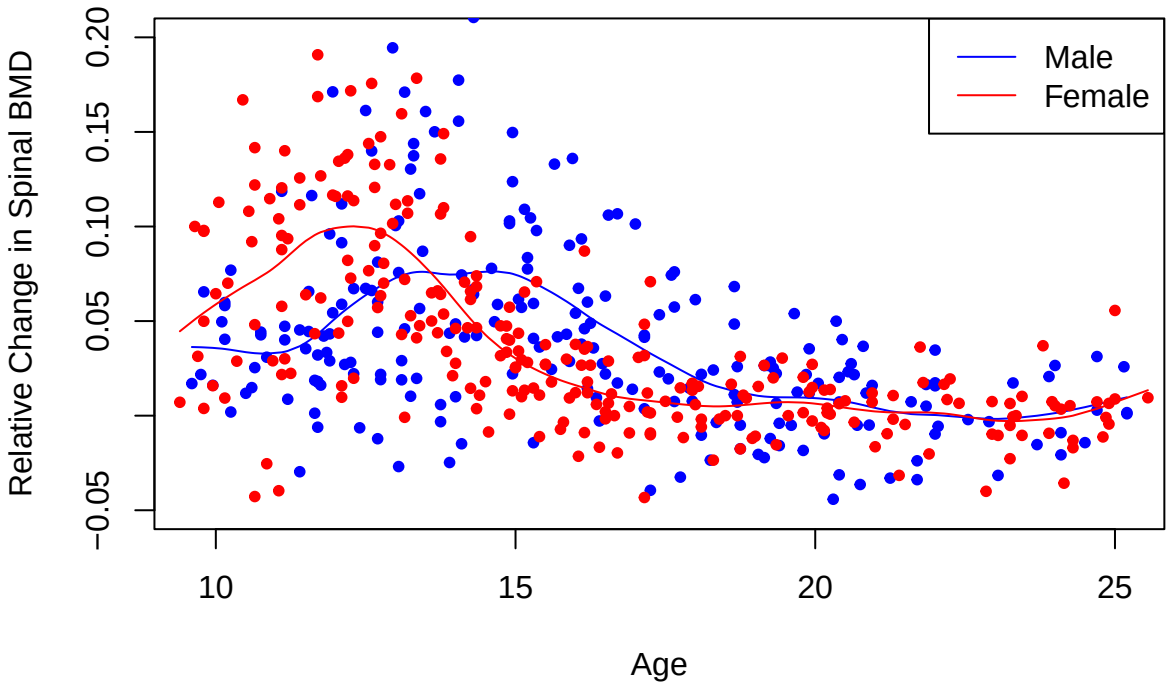
2023-03-01

Question 1

Replicate a figure from ESL Chapter 5

The figure I will be replicating for this question is Figure 5.6 which involves the bone data set from the library ElemStatLearn data base.

```
set.seed(400076920)


#Import the table from the website linked
df <- read.table("https://hastie.su.domains/ElemStatLearn/datasets/bone.data",
                 header=TRUE, col.names=c("idnum", "age", "gender", "spnbmd"))

#split up into two new dataframes from males and females respectively
df_males <- df[df$gender == 'male',]
df_females <- df[df$gender == 'female',]

#Now, create a smooth spline of the dataset for each group created (Note,
#we will use age vs spnbmd)

MaleSmoothSpline <- smooth.spline(df_males$age, df_males$spnbmd, df = 12)
FemaleSmoothSpline <- smooth.spline(df_females$age, df_females$spnbmd, df = 12)

#note df = 12 as this was described by the text.

#Now plot the points and lines of the groups

plot(MaleSmoothSpline, ylim = c(-0.05, 0.20), col = "blue", type = 'l',
     xlab = "Age", ylab = "Relative Change in Spinal BMD")

#I will do my best to tell the colour given my poor colour deception
points(df_males$age, df_males$spnbmd, col = "blue", pch = 20)

#Now put the female data smooth spline onto the same plot

lines(FemaleSmoothSpline, ylim = c(-0.05, 0.20), col = "red")
points(df_females$age, df_females$spnbmd, col = "red", pch = 20)
legend(x = "topright",
       legend = c("Male", "Female"),
       lty = c(1,1),
       col = c("blue", "red"))
```



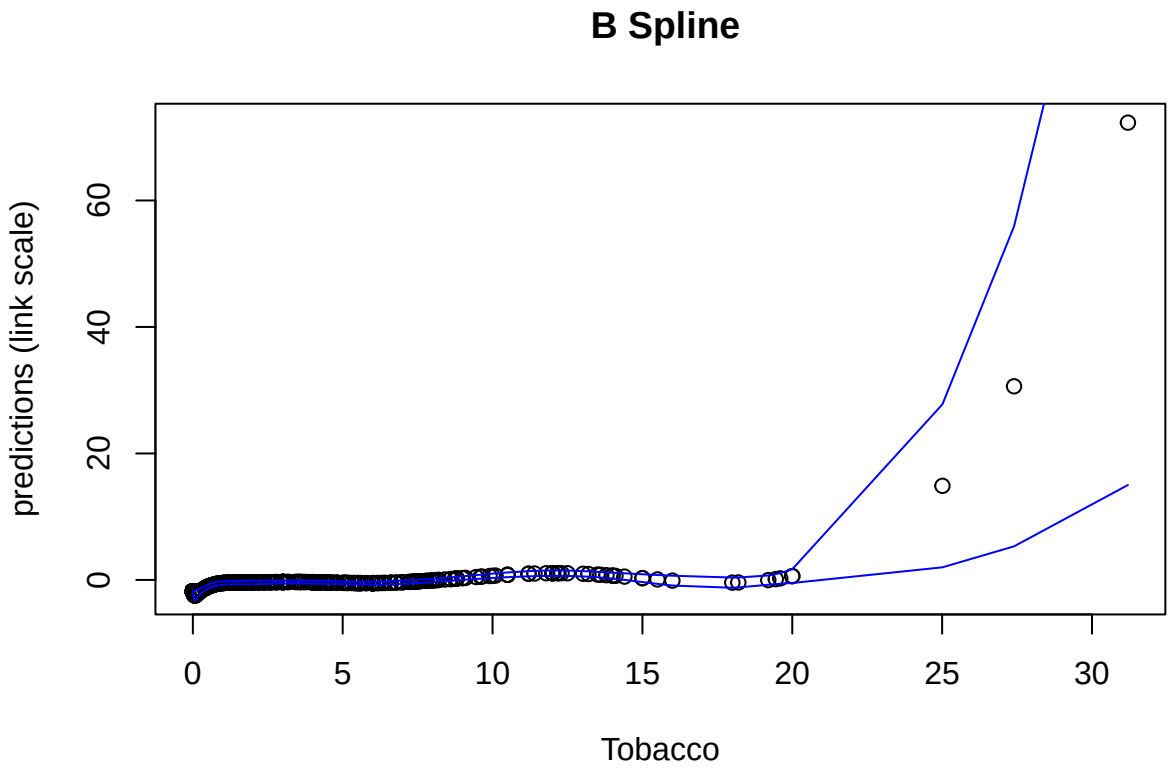
Thus, after coding, figure 5.6 of ESL chapter 5 has been replicated.

Question 2

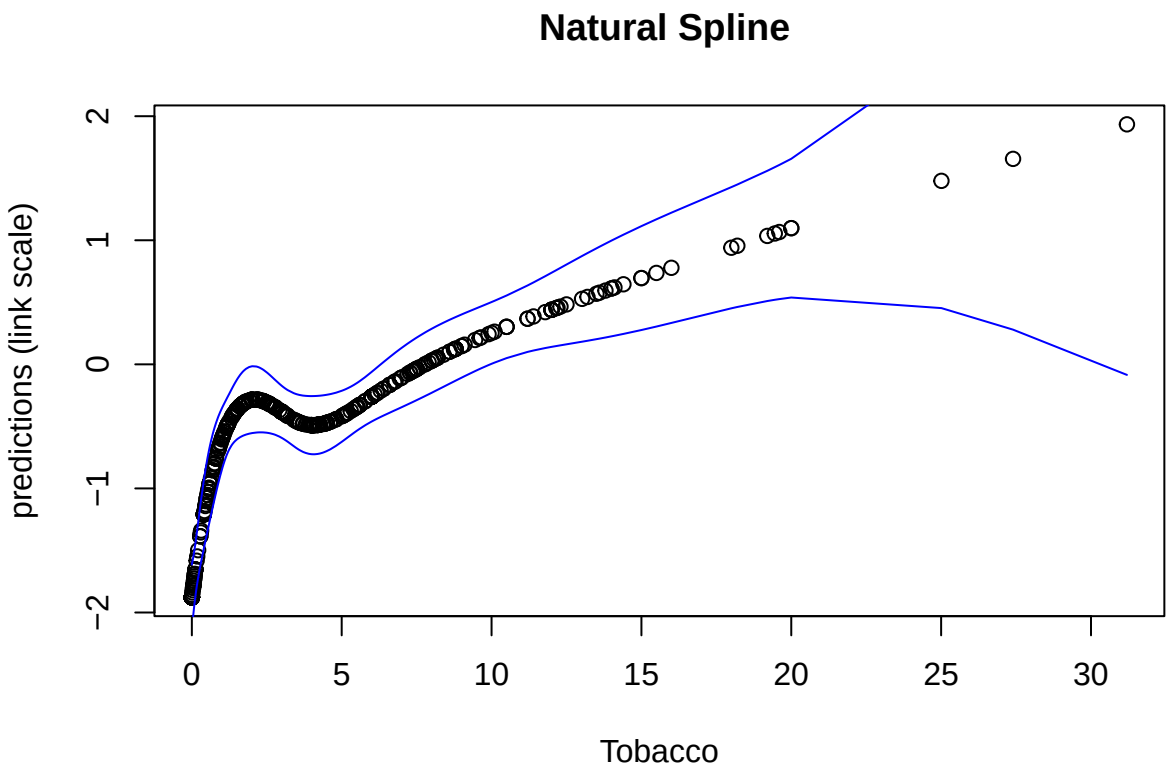
Using the South Africa coronary heart disease data (see code file); construct b-spline, natural spline, and truncated polynomial spline bases (all with 5 knots: you can use the splines package or equivalents, you don't have to do this part "from scratch"). Fit a logistic regression between tobacco (predictor) and chd (response) using each of these bases. Compute the predicted value and the predicted variance for the linear predictor (i.e. on the log-odds scale), without using the built-in predict(..., se.fit = TRUE) methods. Plot the predictions ± 1 SE for each of the three bases.

```
library(splines)
library(ggplot2)
url <- "http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data"
dd <- read.csv(url, row.names = 1)

#create spline bases in R as instructed

knots = quantile(dd$tobacco, probs = seq(0.27, 0.95, length = 5))

#implement the bspline and natural splines using the built in packages
bspline <- bs(dd$tobacco, knots = knots, intercept = FALSE)
nspline <- ns(dd$tobacco, knots = knots, intercept = FALSE)

#implement the truncated polynomial spline function from lecture
truncpolyspline <- function (x, k) {
  knots = quantile(dd$tobacco, probs = seq(0.27, 0.95, length = k))
  trunc_fun <- function(k)
    (x > k)*(x-k)^3
  S <- sapply(knots, trunc_fun)
  S <-cbind(x, x^2, x^3, S)

  return(S)
}

tp <- truncpolyspline(dd$tobacco, 5)


#now fit a logistic regression model between 'chd' and 'tobacco'
bs_model <- glm(chd ~ bspline, family = binomial(), data = dd)
ns_model <- glm(chd ~ nspline, family = binomial(), data = dd)
tp_model <- glm(chd ~ tp, family = binomial(), data = dd)

#find predicted values for bspline
X <- model.matrix(chd ~ bspline, data = dd)
beta_hat_bs <- as.matrix(unname(bs_model$coefficients), ncol = 1)
pred_bs <- X %*% beta_hat_bs

#find predicted values and se of +- 1 for bspline
vcov_bs <- vcov(bs_model)
pred_var_bs <- X%*%vcov_bs%*%t(X)
se_bs <- as.matrix(diag(sqrt(pred_var_bs)))
bs_upper <- pred_bs + se_bs
bs_lower <- pred_bs - se_bs
```

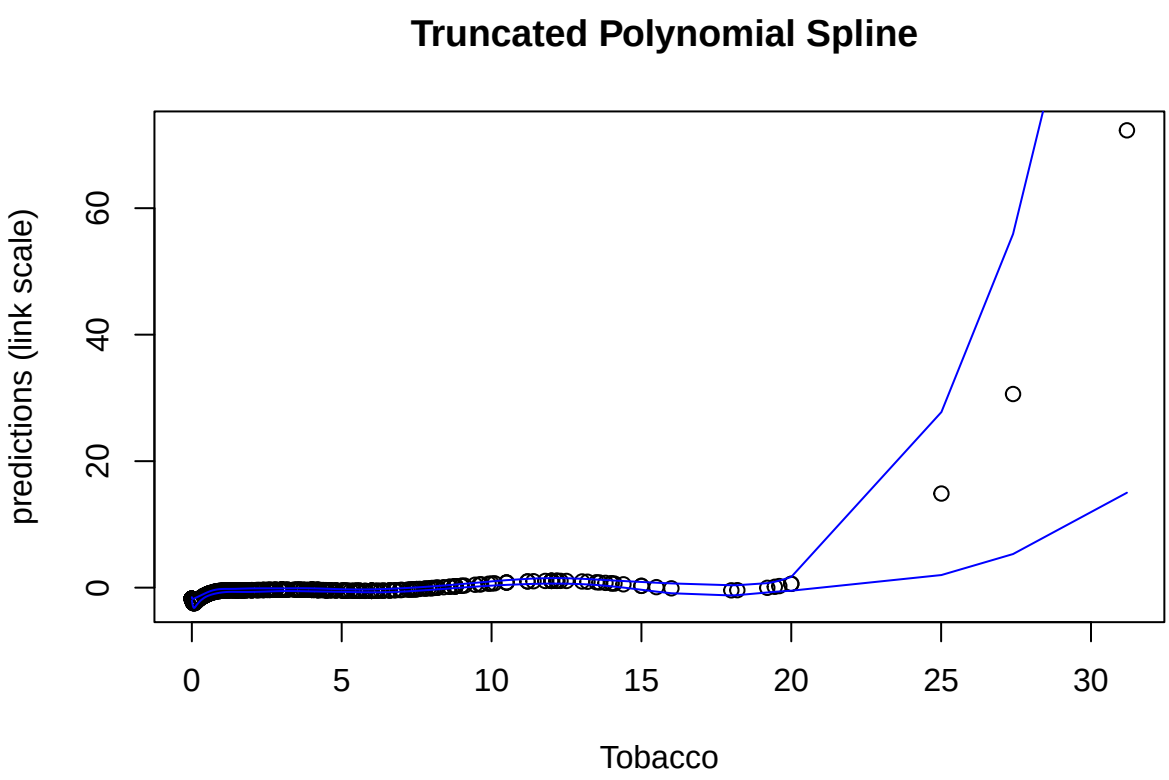
```r
#plot the graph for the bspline
plot(dd$tobacco, pred_bs, xlab = "Tobacco", ylab = "predictions (link scale)",
     main = " B Spline")
lines(sort(dd$tobacco), bs_upper[order(dd$tobacco)], col = "blue")
lines(sort(dd$tobacco), bs_lower[order(dd$tobacco)], col = "blue")
```




```r
#find predicted values for nspline
X <- model.matrix(chd ~ nspline, data = dd)
beta_hat_ns <- as.matrix(unname(ns_model$coefficients), ncol = 1)
pred_ns <- X %*% beta_hat_ns

#find predicted values and se of +- 1 for nspline
vcov_ns <- vcov(ns_model)
pred_var_ns <- X%*%vcov_ns%*%t(X)
se_ns <- as.matrix(diag(sqrt(pred_var_ns)))
ns_upper <- pred_ns + se_ns
ns_lower <- pred_ns - se_ns

#plot the graph for the natural spline
plot(dd$tobacco, pred_ns, xlab = "Tobacco", ylab = "predictions (link scale)",
     main = " Natural Spline")
lines(sort(dd$tobacco), ns_upper[order(dd$tobacco)], col = "blue")
lines(sort(dd$tobacco), ns_lower[order(dd$tobacco)], col = "blue")
```




```r
#find predicted values for truncated polynomial spline
X <- model.matrix(chd ~ tp, data = dd)
beta_hat_tp <- as.matrix(unname(tp_model$coefficients), ncol = 1)
pred_tp <- X %*% beta_hat_tp

#find predicted values and se of +- 1 for truncated polynomial spline
vcov_tp <- vcov(tp_model)
pred_var_tp <- X%*%vcov_tp%*%t(X)
se_tp <- as.matrix(diag(sqrt(pred_var_tp)))
tp_upper <- pred_tp + se_tp
tp_lower <- pred_tp - se_tp

#plot the graph for the truncated polynomial spline discussed in class
plot(dd$tobacco, pred_tp, xlab = "Tobacco", ylab = "predictions (link scale)",
     main = " Truncated Polynomial Spline")
lines(sort(dd$tobacco), tp_upper[order(dd$tobacco)], col = "blue")
lines(sort(dd$tobacco), tp_lower[order(dd$tobacco)], col = "blue")
```

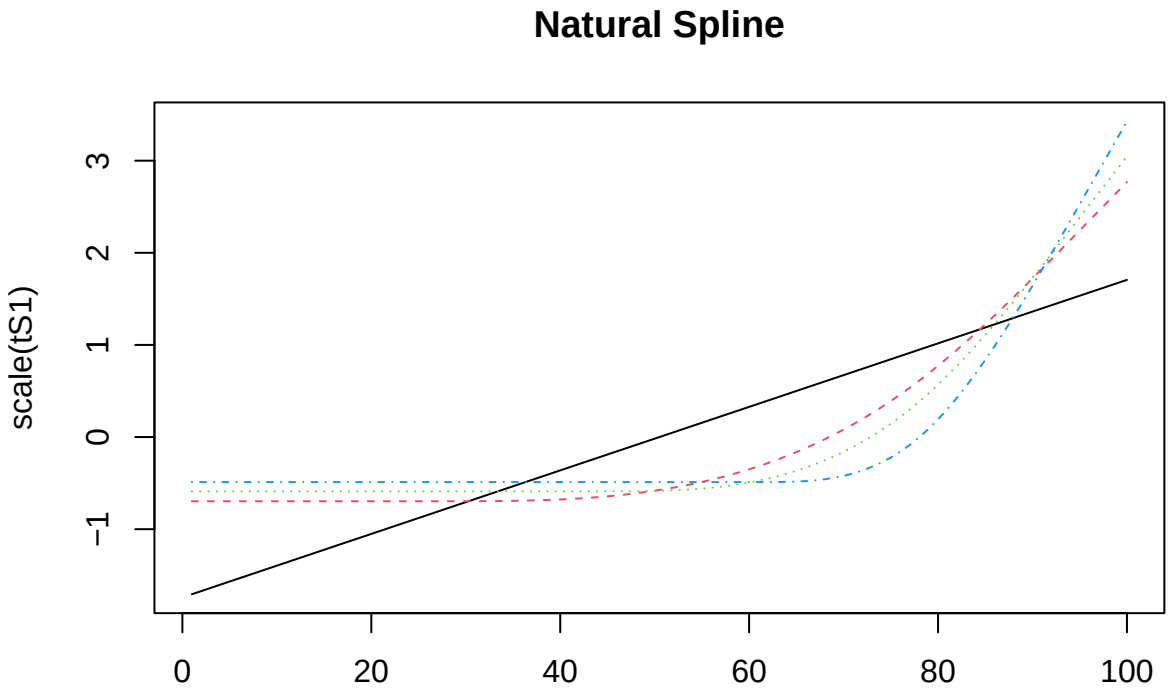


Thus, all three basis have a computed predicted value and standard error. The plots have the standard error of +1 and -1 in a blue line accordingly. One common trend is that the standard error band with higher tobacco values tends to get very wide while all other points tend to be tightly compact with eachother. This suggests that at larger tobacco values we are less likely to observe the true value and at smaller tobacco values we may observe the true value. As a whole, the prediction of these models suggest that the more usage of tobacco means we are more likely to see coronary heart disease in south africa (based on the models produced). One small observation is the bspline and the truncated polynomial spline follow a similar structure to one another. However, upon closer inspection we see that their respective predicted values and se values are entirely different from one another.

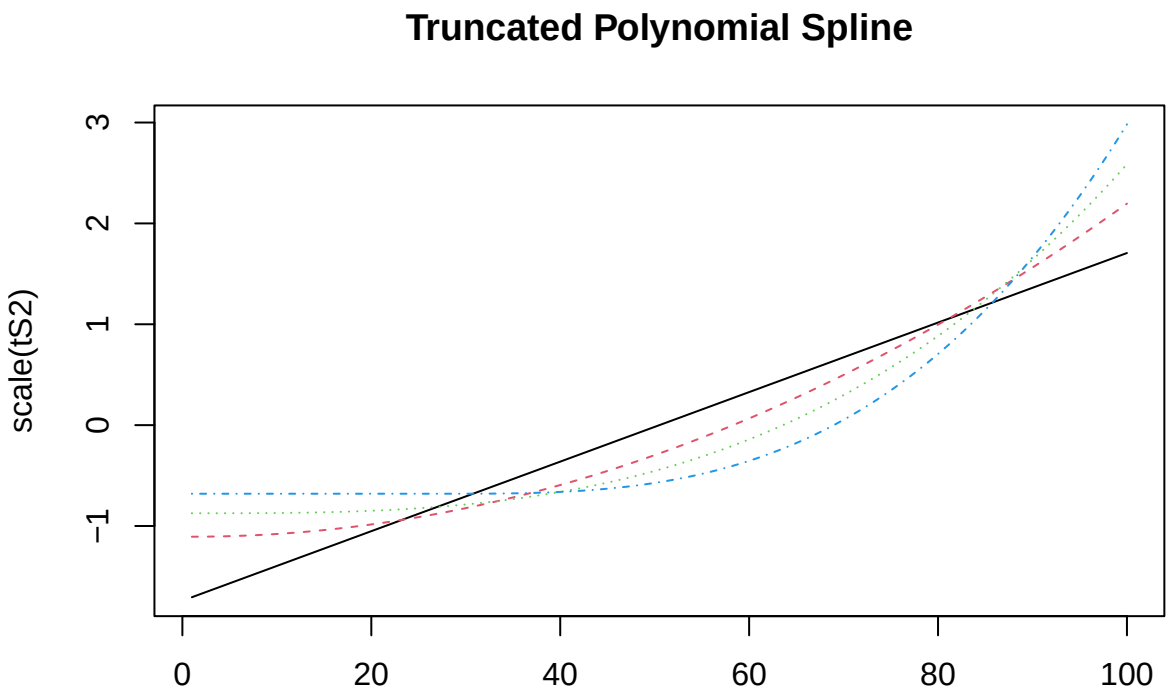
Question 3

Essentially, the purprose of this question is to factor in the result of exercise 5.4 from ESL by using the results of (5.4) and (5.5) in the textbook. In other words, we must create an if statement within the truncated polynomial spline function that will account for this type of natural spline. In order to see this, see the code below inside the for loop for S[, j].

```
#implement the truncated polynomial spline function from lecture
truncpolyspline <- function (x, k, natural) {
  knots = quantile(x, probs = seq(0.27, 0.95, length = k))
  trunc_fun <- function(k)
    (x > k)*(x-k)^3

  if (natural) { #this is when natural constraint is true
    #Build Design Matrix
    S <- matrix(0, nrow = length(x), ncol = k)

    #Begin injecting the appropriate values of S into the function
    S[,1] <- x
    for (j in 2:(k-1)) {

      #This is the magic (i.e. here we are adding the linear constraint)
      S[, j] <- ((x > knots[j-1])*(x - knots [j-1])^3  -
                   (x > knots[k])*(x - knots[k])^3)/(knots[k] - knots[j-1]) -
        ((x - knots [k-1])^3*(x>knots[k-1])  -
           (x - knots[k])^3*(x>knots[k]))/(knots[k] - knots[k-1])
    }
  } else{ #this is when natural constraint is false
    S <- sapply(knots, trunc_fun)
    S <-cbind(x, x^2, x^3, S)
  }


  return(S)
}

#Create a vector - I used the same as in class

xvec <- seq(0,100, length = 100)

#Apply the function with both values of natural (use K - 4) for natural = FALSE
#to ensure the number of lines within the plots are the same

tS1 <- truncpolyspline(xvec, k = 5, natural = TRUE)
tS2 <- truncpolyspline(xvec, k = 1, natural = FALSE)

#Use matplot to plot the graphs
matplot(scale(tS1), type = 'l', main = "Natural Spline")
```




```
matplot(scale(tS2), type = 'l', main = "Truncated Polynomial Spline")
```



So, using the constraint equations described in the link provided, a natural condition was created and this was used to obtain a natural spline with an alteration to the truncated polynomial spline function provided in class. The plots above have been labelled appropriately for the respective type of splines.

Question 4

(a) Write a function to simulate data drawn from a smooth two-dimensional surface on the unit square with Gaussian noise. You could use (for example) a reasonably high-order bivariate polynomial, or a kernel function/mixture model or using a 2D spline.

```r
sim_poly_data <- function(n, noise_sd) {
  # create a grid of x,y values
  x <- y <- seq(0, 1, length.out = n)
  xy <- expand.grid(x, y)

  # define the polynomial function - arbitrary (can change)
  poly_func <- function(x, y) {
    1 + x + y + x^2 + y^2 + x*y + x^3 + y^3 + x*y^2 + x^2*y
  }

  # simulate z values from the polynomial function plus Gaussian noise
  z <- poly_func(xy[,1], xy[,2]) + rnorm(n^2, mean = 0, sd = noise_sd)

  # return the simulated data as a data frame
  return(data.frame(x = xy[,1], y = xy[,2], z = z))
}
```

Thus, a function was created to simulate data from a high-order bivariate polynomial on the unit square with Gaussian noise as instructed. See code above.

(b) Use mgcv::gam() to fit two-dimensional splines to your simulated data, using z ~ te(gp, x, y). Over an ensemble of 250 simulations, compute the average computation time, bias, variance, and mean-squared error of your predictions for (i) method = "GCV.Cp" (generalized cross-validation) and (ii) method = "REML" (restricted maximum likelihood).

```
#I repeated this function
sim_poly_data <- function(n, noise_sd) {
  # create a grid of x,y values
  x <- y <- seq(0, 1, length.out = n)
  xy <- expand.grid(x, y)

  # define the polynomial function - arbitrary (can change)
  poly_func <- function(x, y) {
    1 + x + y + x^2 + y^2 + x*y + x^3 + y^3 + x*y^2 + x^2*y
  }

  # simulate z values from the polynomial function plus Gaussian noise
  z <- poly_func(xy[,1], xy[,2]) + rnorm(n^2, mean = 0, sd = noise_sd)

  # return the simulated data as a data frame
  return(data.frame(x = xy[,1], y = xy[,2], z = z))
}


#impor tthe library as required
library(mgcv)

#intialize the parameters discussed
n <- 50
sd <- 0.1
num_sims <- 250

# Initialize variables as a vector to store results after computation
times_gcv <- rep(0, num_sims)
times_reml <- rep(0, num_sims)
biases_gcv <- rep(0, num_sims)
biases_reml <- rep(0, num_sims)
variances_gcv <- rep(0, num_sims)
variances_reml <- rep(0, num_sims)
mse_gcv <- rep(0, num_sims)
mse_reml <- rep(0, num_sims)

for (i in 1:num_sims) {
  # Simulate data
  data <- sim_poly_data(n, sd)

  # Fit GAM with GCV.Cp
  start_time <- Sys.time()
  fit_gcv <- gam(z ~ te(x, y, bs = "tp", k = 10), data = data, method = "GCV.Cp")
  end_time <- Sys.time()
  times_gcv[i] <- end_time - start_time

  # Fit GAM with REML
  start_time <- Sys.time()
```

```
  fit_reml <- gam(z ~ te(x, y, bs = "tp", k = 10), data = data, method = "REML")
  end_time <- Sys.time()
  times_reml[i] <- end_time - start_time

  # Compute predictions and errors for the gcv and reml vectors
  pred_gcv <- predict(fit_gcv, newdata = data, type = "response")
  pred_reml <- predict(fit_reml, newdata = data, type = "response")
  bias_gcv <- mean(pred_gcv - data$z)
  bias_reml <- mean(pred_reml - data$z)
  var_gcv <- var(pred_gcv - data$z)
  var_reml <- var(pred_reml - data$z)
  mse_gcv[i] <- mean((pred_gcv - data$z)^2)
  mse_reml[i] <- mean((pred_reml - data$z)^2)

  # Store results in their respective vectors
  biases_gcv[i] <- bias_gcv
  biases_reml[i] <- bias_reml
  variances_gcv[i] <- var_gcv
  variances_reml[i] <- var_reml
}

# Compute average results for both gcv and reml as instructed
avg_time_gcv <- mean(times_gcv)
avg_time_reml <- mean(times_reml)
avg_bias_gcv <- mean(biases_gcv)
avg_bias_reml <- mean(biases_reml)
avg_var_gcv <- mean(variances_gcv)
avg_var_reml <- mean(variances_reml)
avg_mse_gcv <- mean(mse_gcv)
avg_mse_reml <- mean(mse_reml)

#Now, create a table with the average values and discuss them briefly to compare
#and contrast the results

results <- data.frame(method = c("GCV.Cp", "REML"),
                      avg_time = c(avg_time_gcv, avg_time_reml),
                      bias = c(avg_bias_gcv, avg_bias_reml),
                      variance = c(avg_var_gcv, avg_var_reml),
                      mse = c(avg_mse_gcv, avg_mse_reml))
results
```

Description: df [2 × 5]

method <chr>	avg_time <dbl>	bias <dbl>	variance <dbl>	mse <dbl>
GCV.Cp	0.1702018	6.794298e-16	0.009795752	0.009791833
REML	2.0385455	-2.766996e-16	0.009679306	0.009675435

2 rows

So, 250 simulations were conducted as instructed by the homework assignment, along with the table of values asked to compute. As we see in the table above, the GCV function has a smaller avg_time, relatively even variance and MSE as compared with the REML. However, a negative bias means it is underestiamted as opposed to a positive bias for overestimation. Notice though, the exponential to the power of -16 which means it is essentially negligible. Thus, we conclude that the GCV function slightly outperforms the REML function based off these metrics

Question 5

ESL Exercise 5.4

Let $f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$ by definition in the question. We need to show that the natural boundary condition for the natural cubic splines (in section 5.2.1) imply the linear constraints displayed in the question of ESL.

Proof:

Recognize that when we assume $X < \xi_1$, the function of $f(X)$ will simply to $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 = \sum_{j=0}^{3} \beta_j X^j$.

Also, note that $\beta_2 = \beta_3 = 0$ as $f(x)$ is linear when $X < \xi_1$ is true. Now, we must consider the case when $X > \xi_K$.

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

and $f(x)$ is once again linear here and we can actually expand the term $(X - \xi_k)^3$ some more.

$$(X - \xi_k)^3 = X^3 - 3X^2\xi_k + 3X\xi_k^2 - \xi_k^3$$

Combining the above result with the initial equation, we obtain the following conditions

$$\beta_3 + \sum_{k=1}^{K} \theta_k = 0$$

$$\beta_2 + \sum_{k=1}^{K} -3\xi_k\theta_k = 0$$

and when you reduce it further we obtain that $\sum_{k=1}^{K} \xi_k\theta_k = 0$ and $\sum_{k=1}^{K} \theta_k = 0$.

Recall the purpose of this question is essentially derive the bases of (5.4) and (5.5) outlined in the question in ESL (i.e. show that (5.5) and (5.4) is equivalent to the definitions of (5.70) and (5.71)). Remember that equivalence means going both ways of the statement (i.e. must show (5.4) and (5.5) imply (5.70) and (5.71) and vice-versa).

To show that going from (5.4) and (5.5) to (5.70) and (5.71) is true. Observe the following arguments.

For $X < \xi_1$, by (5.4) it follows that all N_{k+2} are equal to 0 and thus the function is linear. Now, if we look at the other half of the interval (i.e. $x \geq \xi_k$ with $k \leq K - 2$), N_{k+2} now evaluates to the following expression (with the help of 5.5 in ESL).

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

Now, let us expand out the terms and simplify it a little bit.

$$= \frac{X^3 - 3X^2\xi_k + 3X\xi_k^2 - \xi_k^3 - X^3 - 3X^2\xi_K + 3X\xi_K^2 - \xi_K^3}{\xi_K - \xi_k}$$

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

Note, all we have done this far is cancel terms and re-index k to $K - 1$ to compare.

Continuing to break up the fraction further,

$$= 3\frac{\xi_{K-1}+\xi_K}{\xi_K-\xi_{K+1}}X^2 - 3\frac{\xi_{K-1}+\xi_K}{\xi_K-\xi_{K+1}}X^2 + A_1X + A_0 = A_1X + A_0$$

for some arbitrary constants A_1, A_0. Notice that the first two terms of the derived expression of (5.5) cancel out so we clearly observe that the function is once again linear!

To show that going from (5.70) and (5.71) to (5.4) and (5.5) is true. Observe the following arguments.

$$N_{K+2} = (\xi_K - \xi_k)\theta_k$$

for each $k \leq K-2$ as this is the definition in the book. By our intuition, we can also observe that the values of $N_1(X)$ and $N_2(X)$ are β_0 and β_1 respectively. The goal now is, to obtain the same function by only observing the coefficients of $N_{K+2}(X)$ function. I will call these cofficients J_K.

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

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

Please note, all we have done thus fair is expand and collect terms and re-arrange them into summation form. A couple tedious algebriac steps were skipped to avoid clutter.

Now, if we use the result of (5.71) from ESL chapter 5, we observe that we can rewrite the linear constraints.

In other words,

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

$$\sum_{k=1}^{K-2}\theta_k + \theta_k + \theta_{K-1} = 0...(1)$$

$$\sum_{k=1}^{K-2}\theta_k = -\theta_K - \theta_{K-1}...(2)$$

and using a similar process of extracting terms but for the other linear constraint we obtain $\sum_{k=1}^{K-2}\xi_k\theta_k = -\xi_K\theta_K - \xi_{K-1}\theta_{K-1}$. If we sub both of these new resulting summation forms back into the large expression above (*), we arrive at the following conclusions.

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

(Above I only substituted the results of (1) and (2) into (*))

Simplifying further,

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

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

Now, collecting the last term of the expression above, we can re-add it to the sum and obtain,

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

Thus, we have shown that assuming (5.70) and (5.71) to be true, that the basis in (5.4) and (5.5) have been derived (as desired).

For exercise 5.4, I used theory similar to this link:https://stats.stackexchange.com/questions/172217/why-are-the-basis-functions-for-natural-cubic-splines-expressed-as-they-are-es

ESL Exercise 5.13

Recall (5.26) from ESL as follows

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

The goal of this exercise is to derive the equation above (i.e. the N- fold cross validation formula).

First, let us observe the result of augmenting the data with an extra point using the residual sum-of-squares found as (5.9) in ESL.

In other words,

$$RSS_1(f,\lambda) = RSS(f,\lambda) + (\hat{f}_\lambda(x_0) - f(x_0))^2 \geq RSS(f,\lambda)$$

Here, I denoted RSS_1 as the RSS with the extra point of x_0 added. As we can clearly see, $\hat{f}_\lambda$ is still the minimizer. This will be important moving forwards as this will explain why the fitted smoothing spline with and without the added point are the same. We can then use this result to derive the cross validation formula in (5.26) later on.

Suppose that we let $f_\lambda^{\hat{(-i)}}$ denote the fitted smoothing spline value with the data point x_i removed. Denote the interval as $1 \leq i \leq N$ for the sake of this derivation. Let us start the derivation using the ϵ_i which corresponds to the error.

$$\epsilon = y^{(-l)} - y = (\hat{f_\lambda^{-i}} - y)e_i$$

In the above, $y^{(-l)}$ corresponds to the output vector of the missing data point spline and e_i is some arbitrary standard basis vector.

Recall in the text that S_λ is a smoother matrix that depends on the x_i and the λ which means that it will be the same matrix for the spline with and without the point (i.e. $\hat{f}_\lambda$ and $\hat{f_\lambda^{-i}}$ should in theory have the same smoother matrix.). With this in mind, let us inject the smoother matrix into the error formula above as follows.

$$y^{(-l)} - y = (\hat{f_\lambda^{-i}} - y)e_i$$

$$y^{(-l)} = y + (\hat{f_\lambda^{-i}} - y)e_i$$

Now, multiply both sides by S_λ,

$$S_\lambda y^{(-l)} = S_\lambda y + S_\lambda(\hat{f_\lambda^{-i}} - y_i)e_i$$

But, recall that by the product of the smoother matrix S_λ and some y will be equal to the value of $\hat{f}_\lambda(x_i)$. Using this property and the property of smoother matrix does not depend on a missing point, we can rewrite the above to attain,

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

Now, multiply both sides by (-1) to get the term $(f_\lambda^{\hat{(-i)}} - y)e_i$ to match the one in (5.26),

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

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

Now, add another term of y_i to both sides,

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

Collect the terms and factor out the terms to isolate properly.

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

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

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

Thus, substituting this value back into the cross-validation error formula, we arrive at

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

which is exactly what we wanted to derive in (5.26). Note, to obtain (5.27) it is fairly simple to do so, just substitute the RHS of the last step of the derivation!

For Exercise 5.13, I used this link to understand the theory and apply appropriate transformations to obtain the required result https://stats.stackexchange.com/questions/17431/a-mathematical-formula-for-k-fold-cross-validation-prediction-error

For Exercise 5.13, to understand the relationship between the smoother matrix and the spline fits: https://stats.stackexchange.com/questions/234671/smoother-matrix-from-smooth-spline