Student Name: Sophie Stelmach
Student ID: 400172067

Stats 790
Assignment 3

---

1. Question 1

   We will recreate Figure 5.6 from ESL. We first load the data in, and filter by males and females. Then, the function **ss** is used to create smoothing splines for each gender and these are plotted together.

```
url <- "https://hastie.su.domains/ElemStatLearn/datasets/bone.data"
bmd <- read.csv(url, sep = "")

library(dplyr)
males <- filter(bmd, gender == "male")
females <- filter(bmd, gender == "female")

library(npreg)
ss_male <- ss(males$age, males$spnbmd, df=12)
ss_female <- ss(females$age, females$spnbmd, df=12)

plot(ss_male, col="blue",
     xlab = "Age",
     ylab="Relative Change in Spinal BMD",
     ylim = c(-0.05,0.2),
     main="",
     lty=1, ci = F)
points(males[,c(2,4)], col="blue" , pch=20)
lines(ss_female, col="red",
     lty=1)
points(females[,c(2,4)], col="red" , pch=20)
abline(h=0,lty=3, col="darkgray")
legend(18,0.2,legend=c("Male","Female"),col=c("blue","red"),
       lty=1)
```

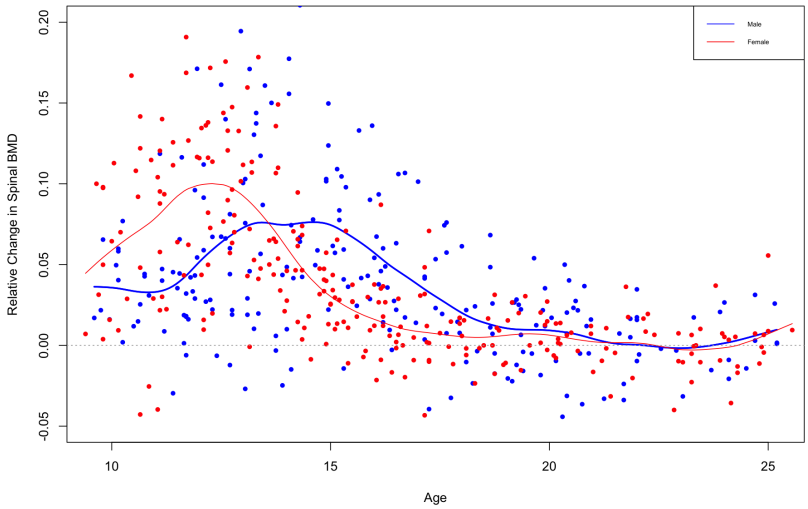

Figure 1: ESL Figure 5.6.

2. Question 2

First we load packages and the data:

```
set.seed(5)

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

Next we create the GLM's using the natural spline, b-spline and truncated polynomial spline. Then, the predicted values are calculated.

```
x <- seq(0, 1, length = 462)
knots <- quantile(x, seq(0,1, length=5))

tobacco<- dd$tobacco
chd <- dd$chd
mod_ns <- glm(as.numeric(chd) ~ ns(tobacco, knots=knots), family = "binomial", data = dd)
mod_bs <- glm(chd ~ bs(tobacco, knots = knots), family = binomial, data = dd)

pred_ns <- model.matrix(mod_ns)[,1:6] %*% mod_ns$coefficients[1:6]
#coefficient 7 was NA

pred_ns <- model.matrix(mod_bs)[,1:8] %*% mod_ns$coefficients[1:8]
#coefficient 9 was NA

truncpolyspline <- function(x, df) {
  if (!require("Matrix")) stop("need Matrix package")
  knots <- quantile(x, seq(0, 1, length = df - 1))
```

```
  ## should probably use seq() instead of ':'
  ## dim: n x (df-2)
  trunc_fun <- function(k) (x>=k)*(x-k)^3
  S <- sapply(knots[1:(df-2)], trunc_fun)
  S <- as(S, "CsparseMatrix")
  ## dim: n x df
  S <- cbind(x, x^2, S)
  return(S)
}

tS <- truncpolyspline(dd$tobacco, df=6)
mod_ts <- glm(chd ~ as.matrix(tS), family = binomial)

pred_ts <- model.matrix(mod_ts) %*% mod_ts$coefficients

#add to data frame
dd$ns.fit <- pred_ns
dd$bs.fit <- pred_bs
dd$ts.fit <- as.matrix(pred_ts)
```

Variance for prediction is calculated to get standard error bars:

```
#variance for error bars
nsX <- model.matrix(mod_ns)[,1:6]
bsX <- model.matrix(mod_bs)[,1:8]
tsX <- model.matrix(mod_ts)

ns_s <- diag(nsX %*% vcov(mod_ns)[1:6,1:6] %*% t(nsX))
dd$ns_max <- pred_ns + ns_s
dd$ns_min <- pred_ns - ns_s

bs_s <- diag(bsX %*% vcov(mod_bs)[1:8,1:8] %*% t(bsX))
dd$bs_max <- pred_bs + bs_s
dd$bs_min <- pred_bs - bs_s

ts_s <- diag(tsX %*% vcov(mod_ts) %*% t(tsX))
dd$ts_max <-pred_ts + ts_s
dd$ts_min <- pred_ts - ts_s
```

Finally, the plots can be made for the predicted values.

```
p1 <- ggplot(dd, aes(log(tobacco), ns.fit)) +
  labs(y = "Predicted values", title = "Natural spline")+
  geom_line()+
  geom_line(aes(x = log(tobacco), y = ns_max, col = "red"), show.legend = FALSE)+
  geom_line(aes(x = log(tobacco), y = ns_min, col = "red"), show.legend = FALSE)


p2 <- ggplot(dd, aes(log(tobacco), bs.fit)) +
  labs(y = "Predicted values", title = "B-spline")+
  geom_line()+
  geom_line(aes(x = log(tobacco), y = bs_max, col = "red"), show.legend = FALSE)+
  geom_line(aes(x = log(tobacco), y = bs_min, col = "red"), show.legend = FALSE)

p3 <- ggplot(dd, aes(log(tobacco), ts.fit)) +
  labs(y = "Predicted values", title = "Truncated polynomial spline")+
  geom_line()+
```

```
  geom_line(aes(x = log(tobacco), y = ts_max, col = "red"), show.legend = FALSE)+
  geom_line(aes(x = log(tobacco), y = ts_min, col = "red"), show.legend = FALSE)


library(gridExtra)
grid.arrange(p1,p2,p3)
```

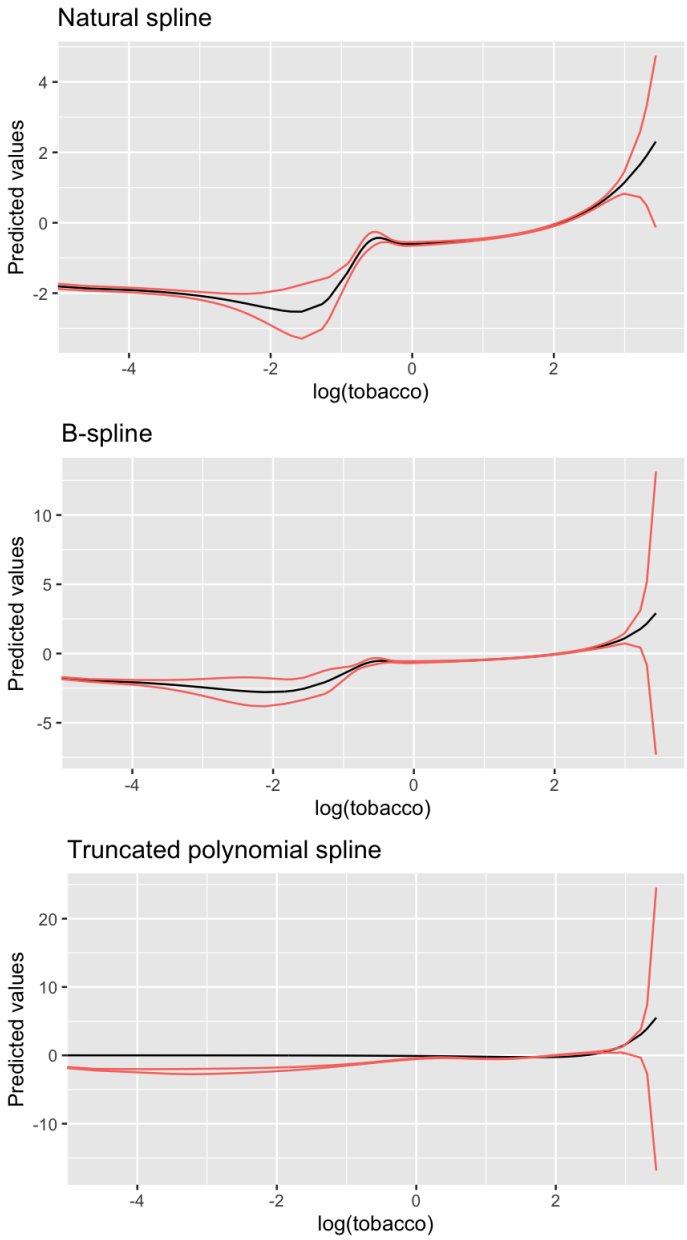

Figure 2: Predictors for tobacco with +/- 1 SE (shown in red), plotted against log(tobacco).

Looking at these 3 figures, the plot of the predicted values almost seems to become smoother going from a natural spline to a b-spline to a truncated polynomial spline. The standard error for all three models explodes near the end of the plot.

3. Question 3

The function below takes two arguments, x and df, where x is the data to be inputted and df is degrees of freedom. First, the function defines the knots, k, and then implements the natural spline if $x \geq k_i$, where $i = 1, .., K$.

```
truncspline <- function(x, df){
  knots <- quantile(x, seq(0,1,length=df-1))

  for (k in seq_along(knots)){
    bs <- (x >= knots[k])*(x - knots[k])^3
    bs <- as(bs, "CsparseMatrix")
    bs <- cbind(x, x^2, bs)
  }
  return(bs)
}

par(mfrow = c(1,2),las =1, bty ="l")
matplot(ns(1:20, df = 5), type = "l", main = "natural spline")
matplot(scale(truncspline(1:20, df = 5)), type = "l", main = "truncated spline")
```

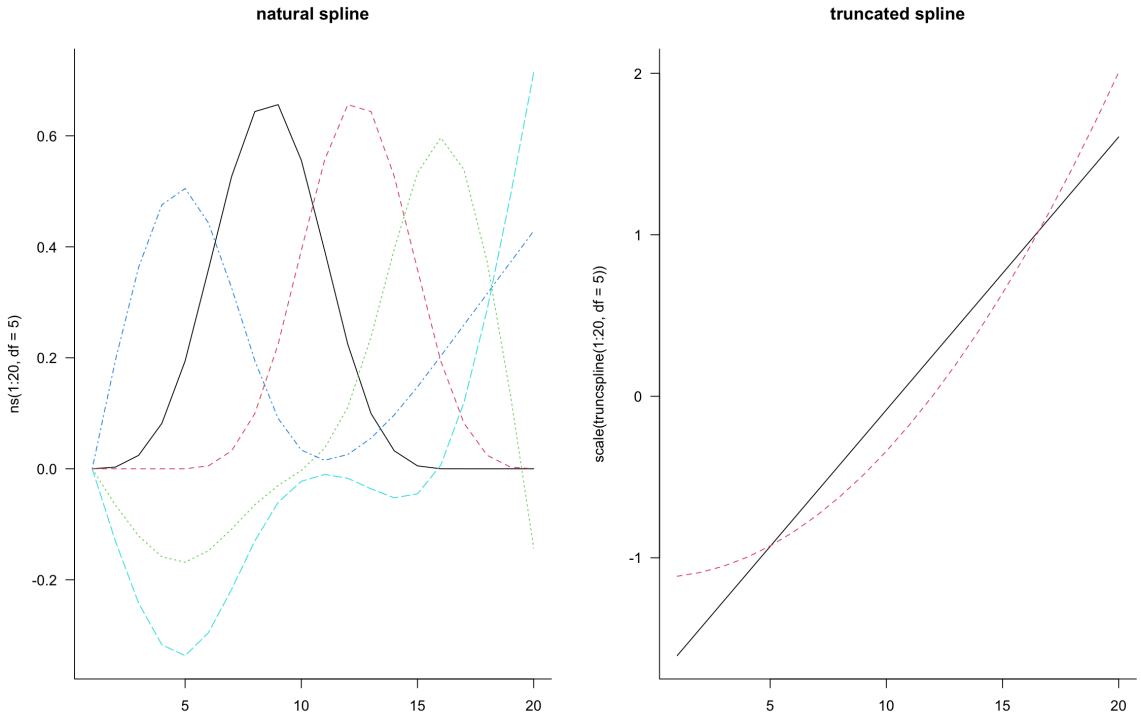

Figure 3: Natural spline vs. truncated polynomial spline.

4. Question 4

We begin by creating a high-order polynomial function and use it to create our data for the simulation.

```
set.seed(5)
x <- seq(0, 1 , length.out = 101)
y <- seq(0, 1 , length.out = 101)

fxn <- function(x,y){
  result <- 2*x^3 + 3*x^4 + y^3 + 5*y^6 + 4*(x*y)^3
  return(result)
}
```

We initialize an empty data frame and create a function that runs two GAM's (one with GCV and one with REML), does predictions for each. It then calculates average computational time, variance, bias, and mean squared error for both model predictions and appends these to the data frame. We then run the simulation 250 times and calculate averages over the 250 simulations for statistics.

```
library(mgcv)
library(microbenchmark)
library(dplyr)
df <- matrix(ncol=8)
simgam <- function(x,y){
  f <- fxn(x,y)
  z <- f + rnorm(n=length(x), mean = 0, sd=0.5*sd(f))
  mod_gcv <- mgcv::gam(z ~ te(x,y,  bs="gp"), method = "GCV.Cp")
  mod_reml <- mgcv::gam(z ~ te(x,y,  bs="gp"), method ="REML")

  pred_gcv <- predict(mod_gcv)
  pred_reml <- predict(mod_reml)

  timing <- microbenchmark(predict(mod_gcv),
                           predict(mod_reml))
  summ <- summary(timing)
  avgtime <- summ$mean
  var_gcv <- var(pred_gcv)
  var_reml <- var(pred_reml)
  bias_gcv <- mean(z-mean(pred_gcv)) #from equation 5.23 in ESL
  bias_reml <- mean(z-mean(pred_reml))
  mse_gcv <- mean(bias_gcv^2 - var_gcv)
  mse_reml <- mean(bias_reml^2 - var_reml)

  tab <- matrix(c(avgtime[1],avgtime[2],
                  var_gcv,var_reml,
                  bias_gcv,bias_reml,
                  mse_gcv,mse_reml),ncol=8)
  result <- rbind(df, tab)
  return(result)
}

suppressWarnings(reps <- replicate(250,simgam(x,y)))
final <- matrix(reps, ncol=8)
final <- colMeans(final,na.rm=T)
final <- matrix(final, nrow=2)
final <- as.data.frame(final)
colnames(final)<- c("Comp. time","Variance","Bias","MSE")
```

```
rownames(final) <- c("GCV","REML")
final
```

The output is decribed in a table:

```
     Comp. time  Variance      Bias       MSE
GCV   0.3644589 0.3752861 0.3624586 0.3686235
REML  0.4718330 0.2159236 0.4715059 0.2301990
```

The REML method has a higher bias, but a lower variance. This displays the bias-variance trade-off. The GCV did have a faster computational time, but also had a slightly higher mean squared error. The methods are comparable, so it depends on the size of the training/test set and the goals of the analysis to decide which one is more appropriate to use.

5. ESL 5.4

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

We want to satisfy the conditions that $f'(X) = 0$ and $f''(X) = 0$.

For $x < \xi_k$:

$$\begin{aligned} f(X) &= \sum_{j=0}^{3} \beta_j X^j \\ f(X) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \\ f'(X) &= \beta_1 + 2\beta_2 X + 3\beta_3 X^2 \\ f''(X) &= 2\beta_2 + 6\beta_3 X \end{aligned}$$

For $f''(X) = 0$ we require $\beta_3 = 0$ and $\beta_2 = 0$.

For $x \geq \xi_k$:

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

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K} \theta_k (X - \xi_k)^3$$

$$f'(X) = \beta_1 + 2\beta_2 X + 3\beta_3 X^2 + 3\sum_{k=1}^{K} \theta_k (X - \xi_k)^2$$

$$f''(X) = 2\beta_2 + 6\beta_3 X + 6\sum_{k=1}^{K} \theta_k (X - \xi_k)$$

$$f''(X) = 0 + 6\sum_{k=1}^{K} \theta_k (X - \xi_k)$$

$$f''(X) = 6\sum_{k=1}^{K} \theta_k X - \theta_k \xi_k$$

$$f''(X) = 6\sum_{k=1}^{K} \theta_k X - 6\sum_{k=1}^{K} \theta_k \xi_k)$$

Thus, we require $\sum_{k=1}^{K} \theta_k \xi_k = 0$ and $\sum_{k=1}^{K} \theta_k = 0$.

Now we move on to constructing the basis (5.4) and (5.5). First, we want $N_1(X) = 1$. Since $h_j(X) = X^{j-1}$ for $j = 1, ..., M$, when $j = 1$, then $h_1(X) = X^{1-1} = X^0 = 1 = N_1(X)$.

Next, we want $N_2(X) = X$. If $j = 2$, then $h_2(X) = X^{2-1} = X^1 = X = N_2(X)$.

From this we can see by observation that the coefficients for N_1 and N_2 are β_0 and β_1, respectively. Finally, we want $N_{k+2}(X) = d_k(X) - d_{K-1}(X)$, where $d_k(X) = \frac{(X-\xi_k)^3-(X-\xi_K)^3}{\xi_K-\xi_k}$.

$$N_{k+2}(X) = d_k(X) - d_{K-1}(X)$$
$$N_{k+2}(X) = \frac{(X-\xi_k)^3 - (X-\xi_K)^3}{\xi_K - \xi_k} - \frac{(X-\xi_k)^3 - (X-\xi_{K-1})^3}{\xi_{K-1} - \xi_k}$$

Since $h_{M+l}(X) = (X-\xi_l)^{M-1}$, set $M = 4$ and then:

$$h_{4+k}(X) - h_{4+K}(X) = (X-\xi_k)^3 - (X-\xi_K)^3$$

And the coefficient must be $\xi_K - \xi_k$.
Thus, $f(X) = \sum_{k=1}^{K} \theta(\xi_K - \xi_k)(d_k(X) - d_{K-1}(X)) = \sum_{k=1}^{K} \theta(\xi_K - \xi_k)N_{k+2}(X)$ If we set $\theta' = \theta(\xi_K - \xi_k)$, we get $f(X) = \sum_{k=1}^{K} \theta' N_{k+2}(X)$ and $\theta_1 = \beta_0, \theta_2 = \beta_1$.

6. ESL 5.13

We want to derive $CV(\hat{f}_\lambda) = 1/N \sum_{i=1}^{N}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2 = 1/N \sum_{i=1}^{N}(\frac{y_i - \hat{f}_\lambda(x_i)}{1-S_\lambda(i,i)})^2$ (LOOCV curve).

We know:

$$\hat{f}_\lambda(x_i) = \sum_{j=1}^{n} S_\lambda(i,j)y_j$$

If we define:

$$\hat{f}_\lambda^{(-i)}(x_i) = -\sum_{j=1}^{n} S_\lambda(i,j)y_j/1 - S_\lambda(i,i)$$

We can then proceed:

$$\begin{aligned}
\hat{f}_\lambda^{(-i)}(x_i) &= -\sum_{j=1}^{n} S_\lambda(i,j)y_j/1 - S_\lambda(i,i) \\
\hat{f}_\lambda^{(-i)}(x_i) &= -\hat{f}_\lambda(x_i)/1 - S_\lambda(i,i) \\
y_i hat f_\lambda^{(-i)}(x_i) &= y_i - \hat{f}_\lambda(x_i)/1 - S_\lambda(i,i)
\end{aligned}$$

Then,

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