STATS 790
Assignment 3

Doudou Jin – 400174871

3/6/23

Q1

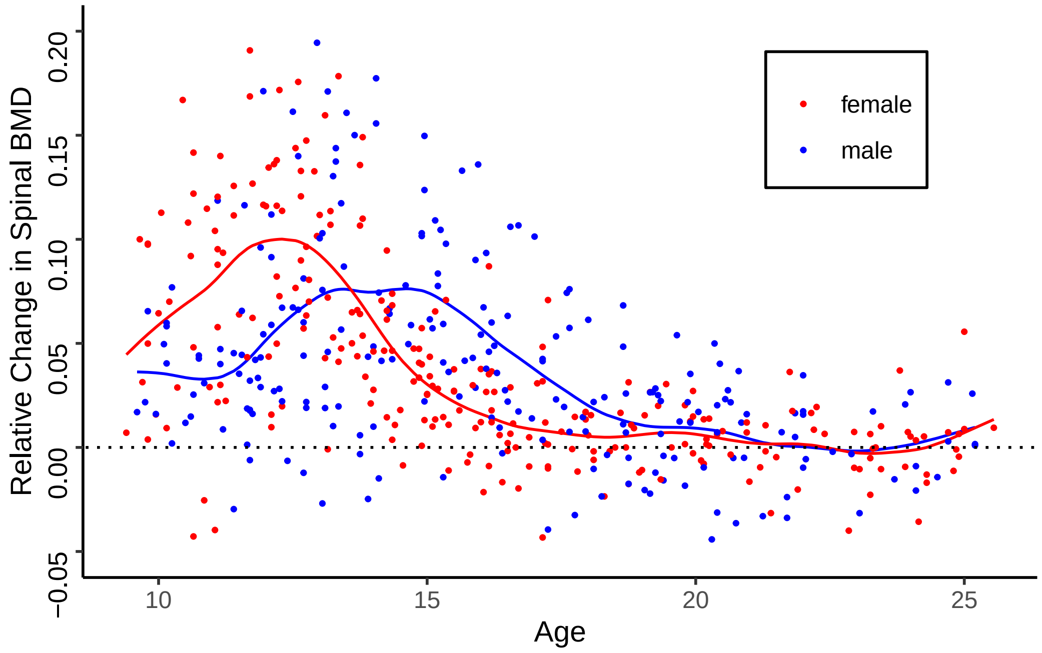
```r
library(dplyr)
library(ggplot2)

bone <- read.delim("bone.data")

m <- filter(bone, gender == "male")
f <- filter(bone, gender == "female")

male_smooth <- smooth.spline(m$age, m$spnbmd, df=12 )
female_smooth <- smooth.spline(f$age, f$spnbmd, df=12 )

ggplot() +
  geom_line(aes(x = male_smooth$x, y = male_smooth$y)
            , color = "blue") +
  geom_line(aes(x = female_smooth$x, y = female_smooth$y)
            , color = "red") +
  geom_point(data = bone
             , aes(x = age, y = spnbmd, color = factor(gender))
             , size = 0.5) +
  scale_color_manual(values = c("male" = "blue", "female" = "red")) +
  ylim(-0.05,0.20) +
  geom_abline(intercept = 0, slope = 0, linetype=3) +
  theme_classic() +
  theme(axis.text.y = element_text(angle = 90
                                   , size = 10
                                   , color = "black"),
        legend.position=c(0.8,0.8),
        legend.title=element_blank(),
        legend.background = element_rect(fill = NA
                                         , color = "black")) +
  ylab("Relative Change in Spinal BMD") +
  xlab("Age")
```



Q2

```r
#bs(), ns()
library(splines)

#tpb()
library(psre)

heart_disease <- read.table(
  "http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data"
  ,sep=","
  ,head=T
  ,row.names=1)
```

b-spline basis

```r
# b-spline basis with 5 knots
knots <- quantile(heart_disease$tobacco, seq(0, 1, length = 5))
model_bs <- glm(chd ~ bs(tobacco, knots = knots, intercept = TRUE)
                , data = heart_disease
                , family = binomial)
summary(model_bs)
```

```
Call:
glm(formula = chd ~ bs(tobacco, knots = knots, intercept = TRUE),
    family = binomial, data = heart_disease)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5824  -0.9912  -0.5583   1.1551   2.1631

Coefficients: (3 not defined because of singularities)
```

```
                                             Estimate Std. Error z value
(Intercept)                                     3.844      4.266   0.901
bs(tobacco, knots = knots, intercept = TRUE)1      NA         NA      NA
bs(tobacco, knots = knots, intercept = TRUE)2  -5.624      4.275  -1.316
bs(tobacco, knots = knots, intercept = TRUE)3  -6.304      4.355  -1.448
bs(tobacco, knots = knots, intercept = TRUE)4  -3.863      4.225  -0.914
bs(tobacco, knots = knots, intercept = TRUE)5  -4.617      4.398  -1.050
bs(tobacco, knots = knots, intercept = TRUE)6  -2.606      3.857  -0.676
bs(tobacco, knots = knots, intercept = TRUE)7  -4.075      6.449  -0.632
bs(tobacco, knots = knots, intercept = TRUE)8      NA         NA      NA
bs(tobacco, knots = knots, intercept = TRUE)9      NA         NA      NA
                                             Pr(>|z|)
(Intercept)                                     0.368
bs(tobacco, knots = knots, intercept = TRUE)1      NA
bs(tobacco, knots = knots, intercept = TRUE)2   0.188
bs(tobacco, knots = knots, intercept = TRUE)3   0.148
bs(tobacco, knots = knots, intercept = TRUE)4   0.360
bs(tobacco, knots = knots, intercept = TRUE)5   0.294
bs(tobacco, knots = knots, intercept = TRUE)6   0.499
bs(tobacco, knots = knots, intercept = TRUE)7   0.528
bs(tobacco, knots = knots, intercept = TRUE)8      NA
bs(tobacco, knots = knots, intercept = TRUE)9      NA

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 596.11  on 461  degrees of freedom
Residual deviance: 538.42  on 455  degrees of freedom
AIC: 552.42

Number of Fisher Scoring iterations: 5
```

natural spline basis

```r
# natural spline basis with 5 knots
model_ns <- glm(chd ~ ns(tobacco, df= 5, intercept = TRUE)
                , data = heart_disease
                , family = binomial)
summary(model_ns)
```

```

Call:
glm(formula = chd ~ ns(tobacco, df = 5, intercept = TRUE), family = binomial,
    data = heart_disease)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6793  -0.9942  -0.5290   1.1877   2.0178

Coefficients: (1 not defined because of singularities)
                                       Estimate Std. Error z value Pr(>|z|)
(Intercept)                               8.543      4.478   1.908   0.0564 .
ns(tobacco, df = 5, intercept = TRUE)1   -8.744      4.504  -1.941   0.0522 .
ns(tobacco, df = 5, intercept = TRUE)2   -9.116      4.368  -2.087   0.0369 *
ns(tobacco, df = 5, intercept = TRUE)3   -2.368      2.579  -0.918   0.3585
ns(tobacco, df = 5, intercept = TRUE)4  -19.248      9.019  -2.134   0.0328 *
ns(tobacco, df = 5, intercept = TRUE)5       NA         NA      NA       NA
---
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: 539.80  on 457  degrees of freedom
AIC: 549.8
```

```
Number of Fisher Scoring iterations: 4
```

truncated polynomial spline basis

```r
# truncated polynomial spline basis with 5 knots
model_tpb <- glm(chd ~ tpb(tobacco, nknots = 5)
                 , data = heart_disease
                 , family = binomial)
summary(model_tpb)
```

```

Call:
glm(formula = chd ~ tpb(tobacco, nknots = 5), family = binomial,
    data = heart_disease)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4377  -0.9535  -0.5375   1.1055   2.0031

Coefficients: (1 not defined because of singularities)
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -1.86176    0.27227  -6.838 8.03e-12 ***
tpb(tobacco, nknots = 5)tpb1  2.27991    4.44505   0.513   0.6080
tpb(tobacco, nknots = 5)tpb2 -2.26246   11.25045  -0.201   0.8406
tpb(tobacco, nknots = 5)tpb3  1.29193    7.61745   0.170   0.8653
tpb(tobacco, nknots = 5)tpb4       NA         NA      NA       NA
tpb(tobacco, nknots = 5)tpb5 -1.33604    7.91160  -0.169   0.8659
tpb(tobacco, nknots = 5)tpb6  0.12973    0.39589   0.328   0.7432
tpb(tobacco, nknots = 5)tpb7 -0.11279    0.07300  -1.545   0.1223
tpb(tobacco, nknots = 5)tpb8  0.02944    0.01519   1.938   0.0526 .
---
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: 535.40  on 454  degrees of freedom
AIC: 551.4

Number of Fisher Scoring iterations: 6
```

predictions for b-spline

```
#Predicted values for b-spline
bs_matrix <- as.data.frame(model.matrix(model_bs))

#remove NA
bs_matrix <- select(.data = bs_matrix
                    , -c("bs(tobacco, knots = knots, intercept = TRUE)1"
                         , "bs(tobacco, knots = knots, intercept = TRUE)8"
                         , "bs(tobacco, knots = knots, intercept = TRUE)9"))
bs_matrix <- as.matrix(bs_matrix)
bs_coef <- na.omit(coef(model_bs))
pred_bs <- bs_matrix %*% bs_coef
```

```
#remove NA
vcov_bs <- as.matrix(vcov(model_bs))
vcov_bs <- vcov_bs[rowSums(is.na(vcov_bs)) != ncol(vcov_bs), ]
vcov_bs <- vcov_bs[, !colSums(is.na(vcov_bs))]

#variance and standard error of predictions
pred_bs_var <- diag(bs_matrix %*% vcov_bs %*% t(bs_matrix))
pred_bs_se <- diag(sqrt(bs_matrix %*% vcov_bs %*% t(bs_matrix)))
```

```r
#prediction ± 1 se
pred_bs_se_up <- pred_bs + pred_bs_se
pred_bs_se_down <- pred_bs - pred_bs_se
```

predictions for natural spline

```r
#Predicted values for natural spline
ns_matrix <- as.data.frame(model.matrix(model_ns))

#remove NA
ns_matrix <- select(.data = ns_matrix
                   , -"ns(tobacco, df = 5, intercept = TRUE)5")
ns_matrix <- as.matrix(ns_matrix)
ns_coef <- na.omit(coef(model_ns))
pred_ns <- ns_matrix %*% ns_coef
```

```r
#remove NA
vcov_ns <- as.matrix(vcov(model_ns))
vcov_ns <- vcov_ns[rowSums(is.na(vcov_ns)) != ncol(vcov_ns), ]
vcov_ns <- vcov_ns[, !colSums(is.na(vcov_ns))]

#variance and standard error of predictions
pred_ns_var <- diag(ns_matrix %*% vcov_ns %*% t(ns_matrix))
pred_ns_se <- diag(sqrt(ns_matrix %*% vcov_ns %*% t(ns_matrix)))
```

```r
#prediction ± 1 se
pred_ns_se_up <- pred_ns + pred_ns_se
pred_ns_se_down <- pred_ns - pred_ns_se
```

predictions for truncated polynomial spline

```
#Predicted values for truncated polynomial spline
tpb_matrix <- as.data.frame(model.matrix(model_tpb))

#remove NA
tpb_matrix <- select(.data = tpb_matrix
                    , -"tpb(tobacco, nknots = 5)tpb4")
tpb_matrix <- as.matrix(tpb_matrix)
tpb_coef <- na.omit(coef(model_tpb))
pred_tpb <- tpb_matrix %*% tpb_coef
```

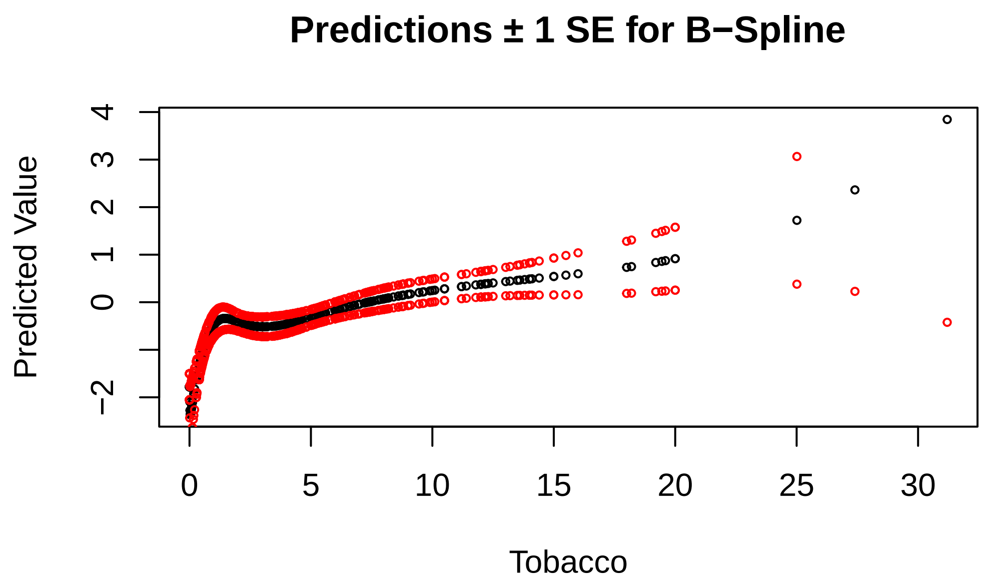
```
#remove NA
vcov_tpb <- as.matrix(vcov(model_tpb))
vcov_tpb <- vcov_tpb[rowSums(is.na(vcov_tpb)) != ncol(vcov_tpb), ]
vcov_tpb <- vcov_tpb[, !colSums(is.na(vcov_tpb))]

#variance and standard error of predictions
pred_tpb_var <- diag(tpb_matrix %*% vcov_tpb %*% t(tpb_matrix))
pred_tpb_se <- diag(sqrt(tpb_matrix %*% vcov_tpb %*% t(tpb_matrix)))
```

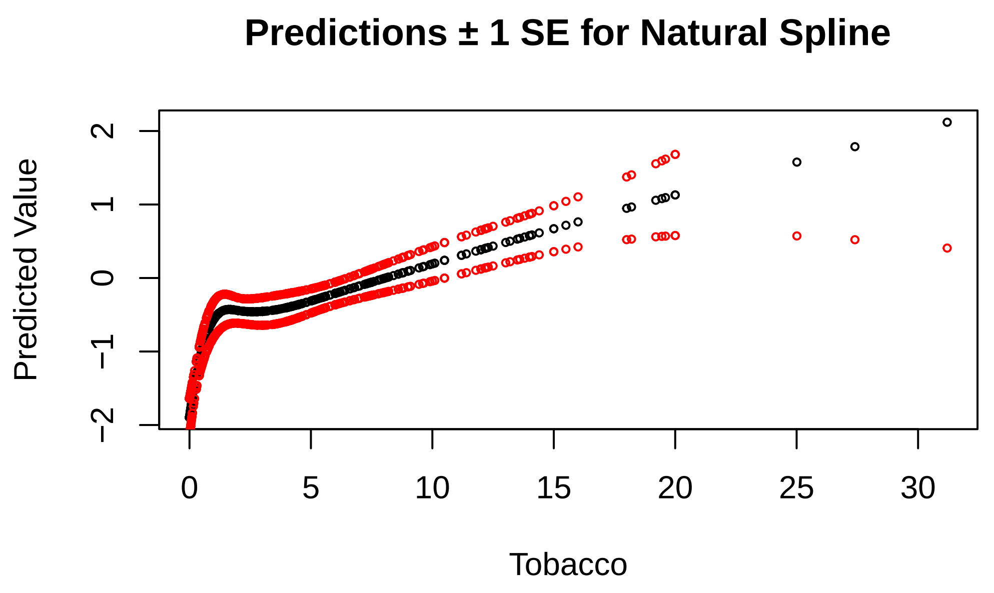
```
#prediction ± 1 se
pred_tpb_se_up <- pred_tpb + pred_tpb_se
pred_tpb_se_down <- pred_tpb - pred_tpb_se
```

Plots of the predictions ± 1 SE for each of the three bases

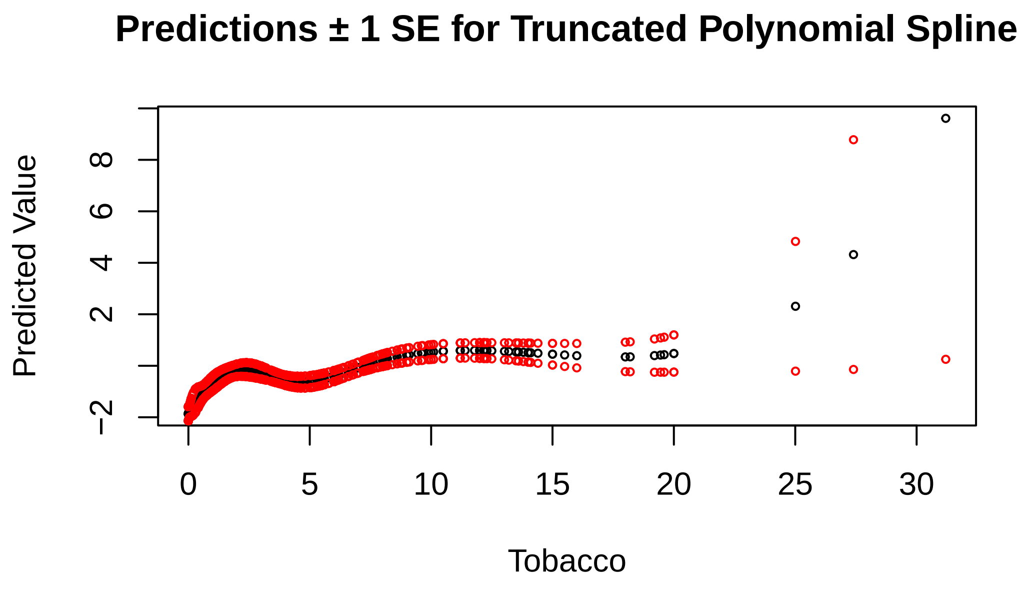
```
#plot predictions ± 1 se for b-spline
plot(y = pred_bs, x=heart_disease$tobacco
     , xlab = "Tobacco"
     , ylab = "Predicted Value"
     , main = "Predictions ± 1 SE for B-Spline"
     , cex = 0.5)
points(y=pred_bs_se_up, x=heart_disease$tobacco, col = "red", cex = 0.5)
points(y=pred_bs_se_down, x=heart_disease$tobacco, col = "red", cex = 0.5)
```



```
#plot predictions ± 1 se for natural spline
plot(y = pred_ns, x=heart_disease$tobacco
     , xlab = "Tobacco"
     , ylab = "Predicted Value"
     , main = "Predictions ± 1 SE for Natural Spline"
     , cex = 0.5)
points(y=pred_ns_se_up, x=heart_disease$tobacco, col = "red", cex = 0.5)
points(y=pred_ns_se_down, x=heart_disease$tobacco, col = "red", cex = 0.5)
```



```
#plot predictions ± 1 se for truncated polynomial spline
plot(y = pred_tpb, x=heart_disease$tobacco
      , xlab = "Tobacco"
      , ylab = "Predicted Value"
      , main = "Predictions ± 1 SE for Truncated Polynomial Spline"
      , cex = 0.5)
points(y=pred_tpb_se_up, x=heart_disease$tobacco, col = "red", cex = 0.5)
points(y=pred_tpb_se_down, x=heart_disease$tobacco, col = "red", cex = 0.5)
```



Q3

```r
library(Matrix)
```

```r
truncpolyspline_natural <- function(x, df, natural) {
  if (!require("Matrix")) stop("need Matrix package")

  trunc_fun <- function(k) (x>=k)*(x-k)^3
  d_k <- function(k){
    (trunc_fun(k) - trunc_fun(knots[df])) / (knots[df] - k)
  }

  if (natural == FALSE){

    # K = df - 4
    knots <- quantile(x, seq(0, 1, length = df - 2))

    # dim: n x (df - 5)
    # 1 : K
    # note: trunc_fun(knots[K]) = 0 in this case
    # note: trunc_fun(knots[1]) = x^3 in this case
    # remove boundary knots

    S <- sapply(knots[2:(df-3)], trunc_fun)
    S <- as(S, "CsparseMatrix")

    # dim: n x df
    # h_1 = 1, h_2 = X, h_3 = X^2, h_4 = X^3,
    # h_(4+l) = (X-knots[l])^3_+, l=1:K-1
    S <- cbind(1, x, x^2, x^3, S)
  }
  else if (natural == TRUE){

    # K = df
    knots <- quantile(x, seq(0, 1, length = df))

    # dim: n x (df - 1)
    # d_1 to d_K-2
    S <- sapply(knots[1:(df-2)], d_k)

    # N_3 to N_K
    S <- S - d_k(knots[df-1])
    S <- as(S, "CsparseMatrix")

    # dim: n x df
    # N_1 = 1, N_2= X, N_k+2 = d_k - d_K-1
    S <- cbind(1, x, S)
  }

  return(S)
}
```

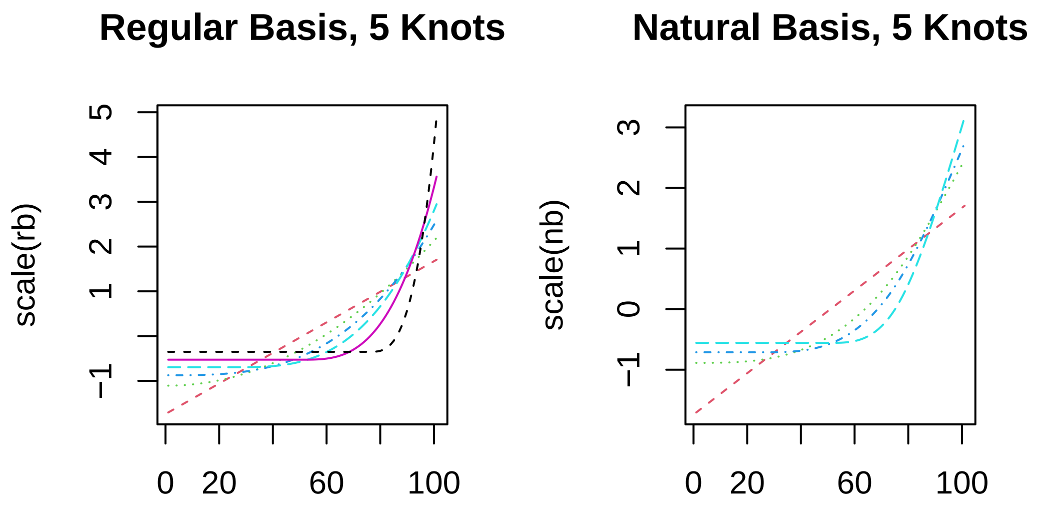
```
xvec <- seq(0, 1, length = 101)

# regular cubic basis with 5 knots, K=5, df = 7
rb <- truncpolyspline_natural(xvec, df = 7, natural = FALSE)

# natural cubic spline basis with 5 knots, K=5, df = 5
nb <- truncpolyspline_natural(xvec, df = 5, natural = TRUE)
```

```
par(mfrow=c(1,2))
matplot(scale(rb), type = "l", main = "Regular Basis, 5 Knots")
```

```r
matplot(scale(nb), type = "l", main = "Natural Basis, 5 Knots")
```



Q4

(a)

```r
library(MASS)

z_surface_values <- function(n){

  # set.seed(1)

  # number of samples that drawn from the smooth two-dimensional surface
  n <- n

  # index
  u <- runif(n, min = 0, max = 1)

  # store samples
  z <- rep(NA,n)
  x <- rep(NA,n)
  y <- rep(NA,n)

  # suppose the surface is the mixture of 3 bivariate Gaussian distributions
  # the weights of each Gaussian distribution is 0.2, 0.3, 0.5
  w <- c(0.2,0.3,0.5)

  # f(x,y) = sum^3_{j=1} w_j exp((((x-x_j)^2 + (y-y_j)^2)/sigma_j^2))

  # simulates 3 bivariate Gaussian distribution randomly
  mu <- runif(6, min = 0, max = 1)
  sigma <- runif(3, min = 0, max = 1)

  # generate random pair (x,y), x in (0,1), y in (0,1)
```

```r
bivxy <- function(mu1, mu2, sigma1){

  # generate a random (x,y)
  biv <- mvrnorm(1, mu=c(mu1, mu2)
                 , Sigma = matrix(c(sigma1, 0, 0, sigma1), ncol=2))

  # keep regenerating if one of x, y are outside range
  while(biv[1] < 0 || biv[1] > 1 || biv[2] < 0 || biv[2] > 1){
    biv <- mvrnorm(1, mu=c(mu1, mu2)
                   , Sigma=matrix(c(sigma1, 0, 0, sigma1), ncol=2))
    }

  # return (x,y) within the range
  return(biv)
}

# exp((((x-x_j)^2 + (y-y_j)^2)/sigma_j^2))
fxy <- function(x, y, mu1, mu2, sigma1){
  z <- exp((((x-mu1)^2 + (y-mu2)^2)/sigma1^2))

  return(z)
}




for(i in 1:n){

  # in the first bivariate Gaussian distribution
  if(u[i] < w[1]){
    biv <- bivxy(mu[1], mu[2], sigma[1])
    z[i] <- fxy(biv[1], biv[2], mu[1], mu[2], sigma[1])
    x[i] <- biv[1]
    y[i] <- biv[2]
  }

  # in the second bivariate Gaussian distribution
  else if(u[i] < w[1] + w[2]){
    biv <- bivxy(mu[3], mu[4], sigma[2])
    z[i] <- fxy(biv[1], biv[2], mu[3], mu[4], sigma[2])
    x[i] <- biv[1]
    y[i] <- biv[2]
  }

  # in the third bivariate Gaussian distribution
  else{
    biv <- bivxy(mu[5], mu[6], sigma[3])
    z[i] <- fxy(biv[1], biv[2], mu[5], mu[6], sigma[3])
    x[i] <- biv[1]
    y[i] <- biv[2]
  }
}

gaussian_noise <- function(a){

  for (i in 1:length(a)){

    #generate a noise
    a[i] <- a[i] + rnorm(1, mean = 0, sd = 0.1)

    # keep regenerating if the value is outside range
    while (a[i] < 0 || a[i] > 1) {
          a[i] <- a[i] + rnorm(1, mean = 0, sd = 0.1)
        }
      }
      return(a)
    }

    # make sure the data are drawn on unit square
    x <- gaussian_noise(x)
    y <- gaussian_noise(y)

    # no limit for z
    z <- z + rnorm(n, mean = 0, sd = 0.1)




    return(list(x = x, y = y, z = z))
  }
```

(b)

```
library(mgcv)
library(knitr)
```

```
# compute computation time, variance, bias, MSE for one simulation

comps <- function(i){

    set.seed(i)
    x <- z_surface_values(100)$x
    y <- z_surface_values(100)$y
    z <- z_surface_values(100)$z
```

```r
# fit models
mod_Cp <- gam(z ~ te(x,y, bs="gp"), method = "GCV.Cp")
mod_Ml <- gam(z ~ te(x,y, bs="gp"), method = "REML")

# make predictions
pred_Cp <- predict(mod_Cp, newdata = data.frame(x,y))
pred_Ml <- predict(mod_Ml, newdata = data.frame(x,y))


# computational times
comp_time_Cp <- system.time(gam(z ~ te(x,y, bs="gp")
                                , method = "GCV.Cp"))[1]
comp_time_Ml <- system.time(gam(z ~ te(x,y, bs="gp")
                                , method = "REML"))[1]

# bias
bias_Cp <- mean(pred_Cp) - mean(z)
bias_Ml <- mean(pred_Ml) - mean(z)

# variance
pred_Cp_var <- mean(diag(model.matrix(mod_Cp)
                         %*% vcov(mod_Cp)
                         %*% t(model.matrix(mod_Cp))))
pred_Ml_var <- mean(diag(model.matrix(mod_Ml)
                         %*% vcov(mod_Ml)
                         %*% t(model.matrix(mod_Ml))))

# MSE
MSE_Cp <- mean((pred_Cp-z)^2)
MSE_Ml <- mean((pred_Ml-z)^2)
```

```r
    return(x = rbind(comp_time_Cp, bias_Cp, pred_Cp_var, MSE_Cp
              , comp_time_Ml, bias_Ml, pred_Ml_var, MSE_Ml))
}
```

```r
# remove i = 22,26,35,50,59,63,67,78,90,97,103,107,119,146,165,181,182,
# 183,184,202,210,224,226,229,239,260,263,275

comp <- comps(1)

for (i in 2:278) {
  current_i <- i

  tryCatch({
    comp <- comp + comps(i)

  }, error = function(e) {

    message(paste0("Can't compute at i = ", current_i
                   , ": ", conditionMessage(e)))
  })
}
```

```r
avg_comp <- comp/250
Cp <- rbind(avg_comp[1], avg_comp[2], avg_comp[3], avg_comp[4])
Ml <- rbind(avg_comp[5], avg_comp[6], avg_comp[7], avg_comp[8])
df <- cbind(Cp, Ml)
colnames(df) <- c("method = GCV.Cp", "method = REML")
rownames(df) <- c("Computation Time", "Bias", "Variance", "MSE")

kable(df, caption = "Averages on 250 Simulations")
```

Table 1: Averages on 250 Simulations

	method = GCV.Cp	method = REML
Computation Time	2.232800e-02	4.174400e-02
Bias	1.257873e+55	-1.924564e+58
Variance	1.866292e+150	1.885191e+150
MSE	4.479100e+151	4.475434e+151

```
# method = "GCV.Cp" is slightly better than method = "REML"
```

Q5

The truncated power series representation for cubic splines with K interior knots is

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

Let ξ_1 and ξ_K be two boundary knots. Let $X < \xi_1$, then $\sum_{k=1}^{K} \theta_k (X-\xi_k)_+^3 = 0$ and so $f(x) = \sum_{j=0}^{3} \beta_j X^j = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$. Since the function is linear near the boundaries in natural cubic spline, $f(x) = \beta_0 + \beta_1 X$ in this case. So $\beta_2 = \beta_3 = 0$.

Let $X \geq \xi_K$, $f(x) = \beta_0 + \beta_1 X + \sum_{k=1}^{K} \theta_k (X - \xi_k)^3$. Expand $(X - \xi_k)^3$ to get $(X - \xi_k)^3 = X^3 - 3X^2\xi_k + 3X\xi_k^2 - \xi_k^3$. Then

$$\begin{aligned}
\sum_{k=1}^{K} \theta_k (X-\xi_k)^3 &= \sum_{k=1}^{K} \theta_k (X^3 - 3X^2\xi_k + 3X\xi_k^2 - \xi_k^3) \\
&= \sum_{k=1}^{K} \theta_k X^3 - 3\sum_{k=1}^{K} \theta_k X^2 \xi_k + 3\sum_{k=1}^{K} \theta_k X \xi_k^2 - \sum_{k=1}^{K} \theta_k \xi_k^3 \\
&= \sum_{k=1}^{K} \theta_k X^3 - 3\sum_{k=1}^{K} \theta_k \xi_k X^2 + 3\sum_{k=1}^{K} \theta_k \xi_k^2 X - \sum_{k=1}^{K} \theta_k \xi_k^3
\end{aligned}$$

Again, since the function is linear,

$$\sum_{k=1}^{K} \theta_k X^3 = -3\sum_{k=1}^{K} \theta_k \xi_k X^2 = 0$$

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

Then derive the basis (5.4) and (5.5).

Let

$$\begin{aligned}\sum_{k=1}^{K}\theta_k(\xi_K-\xi_k) &= \sum_{k=1}^{K-1}\theta_k(\xi_K-\xi_k)+\theta_K(\xi_K-\xi_K)\\ &= \sum_{k=1}^{K-1}\theta_k(\xi_K-\xi_k)\end{aligned}$$

Since $\sum_{k=1}^{K}\theta_k\xi_K-\sum_{k=1}^{K}\theta_k\xi_k=\sum_{k=1}^{K}\theta_k(\xi_K-\xi_k)=0$, $\sum_{k=1}^{K-1}\theta_k(\xi_K-\xi_k)=0$ as well.

Then we can have

$$\begin{aligned}\sum_{k=1}^{K-1}\theta_k(\xi_K-\xi_k) &= 0\\ \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)+\theta_{K-1}(\xi_K-\xi_{K-1}) &= 0\\ \theta_{K-1} &= -\frac{\sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)}{\xi_K-\xi_{K-1}}\\ &= -\sum_{k=1}^{K-2}\theta_k\frac{\xi_K-\xi_k}{\xi_K-\xi_{K-1}}\end{aligned}$$

$$\begin{aligned}\sum_{k=1}^{K}\theta_k &= 0\\ \sum_{k=1}^{K-1}\theta_k+\theta_K &= 0\\ \theta_K &= -\sum_{k=1}^{K-1}\theta_k\end{aligned}$$

Thus,

$$\begin{aligned}
f(X) &= \beta_0 + \beta_1 X + \sum_{k=1}^{K} \theta_k (X-\xi_k)_+^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-1} \theta_k (X-\xi_k)_+^3 + \theta_K (X-\xi_K)_+^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-1} \theta_k (X-\xi_k)_+^3 - \sum_{k=1}^{K-1} \theta_k (X-\xi_K)_+^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-1} \theta_k ((X-\xi_k)_+^3 - (X-\xi_K)_+^3) \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k ((X-\xi_k)_+^3 - (X-\xi_K)_+^3) + \theta_{K-1}((X-\xi_{K-1})_+^3 - (X-\xi_K)_+^3) \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k ((X-\xi_k)_+^3 - (X-\xi_K)_+^3) - \sum_{k=1}^{K-2} \theta_k \frac{\xi_K-\xi_k}{\xi_K-\xi_{K-1}}((X-\xi_{K-1})_+^3 - (X-\xi_K)_+^3) \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k ((X-\xi_k)_+^3 - (X-\xi_K)_+^3 - \frac{\xi_K-\xi_k}{\xi_K-\xi_{K-1}}((X-\xi_{K-1})_+^3 - (X-\xi_K)_+^3))
\end{aligned}$$

Let

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

Hence,

$$\begin{aligned}
f(X) &= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k ((\xi_K-\xi_k)d_k(X) - (\xi_K-\xi_k)d_{K-1}(X)) \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (\xi_K-\xi_k)(d_k(X) - d_{K-1}(X)) \\
&= \beta_0 N_1(X) + \beta_1 N_2(X) + \sum_{k=1}^{K-2} \theta_k (\xi_K-\xi_k) N_{k+2}(X)
\end{aligned}$$

where $N_1(X) = 1$, $N_2(X) = X$, $N_{k+2}(X) = d_k(X) - d_{K-1}(X)$ (5.4) and $d_k(X) = \frac{(X-\xi_k)_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_k}$ (5.5).

Q6

Let $\hat{f}_\lambda^{(-i)}$ be the smooth spline with the i-th pair (x_i, y_i) removed.

Since the new pair of $(x_0, \hat{f}_\lambda(x_0))$ is augmented into N samples, there are still N samples that need to be fitted for each $\hat{f}_\lambda^{(-i)}(x_i)$ and so $\hat{f}_\lambda^{(-i)}$ shares the same smoothing matrix $\mathbf{S}_\lambda$ with $\hat{f}_\lambda$.

By (5.14), $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$

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

Also,

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

since y_i was removed, when $j = i$ we use $\hat{f}_\lambda^{(-i)}(x_i)$ to replace it.

That implies

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

Therefore, the N-fold (leave-one-out) cross-validation formula is:

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

Proved.

Reference

Armstrong, Dave, and Robert Andersen. 2022. *Psre: Presenting Statistical Results Effectively.* Manual.

Bates, Douglas, Martin Maechler, and Mikael Jagan. 2022. *Matrix: Sparse and Dense Matrix Classes and Methods.* Manual.

(https://stats.stackexchange.com/users/155499/ahstat), ahstat. n.d. "Why Are the Basis Functions for Natural Cubic Splines Expressed as They Are? (ESL)." Cross Validated.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing.* Manual. Vienna, Austria: R Foundation for Statistical Computing.

Tibshirani, R., T. Hastie, and J. H. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction : With 200 Full-color Illustrations.* Springer Series in Statistics. Springer.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S.* Fourth. New York: Springer.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation.*

Wood, S. N. 2017. *Generalized Additive Models: An Introduction with R.* Second. Chapman and Hall/CRC.

Wood, S. N. 2003. "Thin-Plate Regression Splines." *Journal of the Royal Statistical Society (B)* 65 (1): 95–114.

———. 2004. "Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models." *Journal of the American Statistical Association* 99 (467): 673–86.

———. 2011. "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models." *Journal of the Royal Statistical Society (B)* 73 (1): 3–36.

Wood, S. N., N., Pya, and B. S"afken. 2016. "Smoothing Parameter and Model Selection for General Smooth Models (with Discussion)." *Journal of the American Statistical Association* 111: 1548–75.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing*

Reproducible Computational Research, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman and Hall/CRC.