Stats790 Assignment3

Hanwen Ju

March 2023

1 ESL 5.4

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

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

Assume we have $x1 \leq \xi_1$ and $x_2 \geq \xi_2$. So we have the polynomial for $x1$:

$$f(x_1) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 \tag{2}$$

by take its 1st derivative:

$$f'(x_1) = \beta_1 + \beta_2 x_1 + \beta_3 x_1^2 \tag{3}$$

By the definition of nature boundary conditions for natural cubic splines, its second and third derivative should be zero. So its first derivative is constant. Thus, we must have:

$$\beta_2 = \beta_3 = 0 \tag{4}$$

Consider x_2, we have the polynomial for x_2:

$$\begin{aligned} f(x_2) &= \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + \beta_3 x_2^3 + \sum_{k=1}^{K} \theta_k (x_2 - \xi_k)_+^3 \\ &= \beta_0 + \beta_1 x_2 + \sum_{k=1}^{K} \theta_k (x_2 - \xi_k)_+^3 \end{aligned} \tag{5}$$

By taking its 1st derivative, we get:

$$\begin{aligned} f'(x_2) &= \beta_1 + 3 \sum_{k=1}^{K} \theta_k (x_2 - \xi_k)^2 \\ &= \beta_1 + 3 \left[\sum_{k=1}^{K} \theta_k x_2^2 - 2 \sum_{k=1}^{K} \theta_k \xi_k x_2 + \sum_{k=1}^{K} \theta_k \xi_k^2 \right] \end{aligned} \tag{6}$$

Similarly, we know $f'(x_2) = 0$. So we must have:

$$\sum_{k=1}^{K} \theta_k = 0 \tag{7}$$

and

$$\sum_{k=1}^{K} \theta_k \xi_k = 0 \tag{8}$$

Back to the equation (5), we know the general case for any $x_i \geq \xi_2$ is:

$$f(x_i) = \beta_0 + \beta_1 x_i + \sum_{k=1}^{K} \theta_k (x_i - \xi_k)_+^3 \tag{9}$$

So the first two basis for natural cubic spline is:

$$N_1 = 1, N_2 = X \tag{10}$$

We know we have $(K+1) \times 4 - 3 \times K - 4 = K$ parameters for natural cubic spline with K knots. We want to have the form $f(x) = \sum_{k=1}^{K} \beta_k' N_k(x)$. First, by using the constraints $\sum_{k=1}^{K} \theta_k = 0$ and $\sum_{k=1}^{K} \theta_k \xi_k = 0$, we can derive θ_K and θ_{K-1} from $\sum_{k=1}^{K} \theta_k = \sum_{k=1}^{K} \theta_k \xi_k$:

$$\theta_{K-1} = -\sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} \tag{11}$$

$$\theta_K = -\sum_{k=1}^{K-2} \theta_k \frac{\xi_{K-1} - \xi_k}{\xi_{K-1} - \xi_K} \tag{12}$$

We can rewrite the $f(X)$ as:

$$\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-2} \theta_k (X - \xi_k)^3 + \theta_{K-1}(X - \xi_{K-1})^3 + \theta_K (X - \xi_K)^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (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 - \sum_{k=1}^{K-2} \theta_k \frac{\xi_{K-1} - \xi_k}{\xi_{K-1} - \xi_K} (X - \xi_K)^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k \left[(X - \xi_k)^3 - \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})^3 - \frac{\xi_{K-1} - \xi_k}{\xi_{K-1} - \xi_K} (X - \xi_K)^3 \right] \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k \left[(X - \xi_k)^3 - \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})^3 - \frac{(\xi_K - \xi_k) - (\xi_K - \xi_{K-1})}{\xi_{K-1} - \xi_K} (X - \xi_K)^3 \right] \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k \left[(X - \xi_k)^3 - \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})^3 - \frac{\xi_K - \xi_k}{\xi_{K-1} - \xi_K} (X - \xi_K)^3 - (X - \xi_K)^3 \right] \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k \left[(X - \xi_k)^3 - (X - \xi_K)^3 - \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})^3 - \frac{\xi_K - \xi_k}{\xi_{K-1} - \xi_K} (X - \xi_K)^3 \right] \\
&= \beta_0 + \beta_1 X + \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{1}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})^3 - \frac{1}{\xi_{K-1} - \xi_K} (X - \xi_K)^3 \right] \\
&= \beta_0 + \beta_1 X + \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] \\
&= \beta_0 + \beta_1 X + \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] \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \left[d_k(X) - d_{K-1}(X) \right]
\end{aligned} \tag{13}$$

where:

$$d_k = \frac{(X - \xi_k)^3 - (X - \xi_K)^3}{\xi_K - \xi_k} \tag{14}$$

And we have:

$$N_{k+2} = d_k(X) - d_{K-1}(X) \tag{15}$$

2 ESL 5.13

Suppose we have fitted a smoothing spline $\hat{f}_\lambda$ to a sample of N pairs (x_i, y_i). We have:

$$\hat{f}_\lambda(x_0) = S_\lambda y = \sum_{j=1}^{N} S_\lambda(i,j)y_j + S_\lambda(i,0)\hat{f}_\lambda(x_0) \tag{16}$$

If we add a new pair defined in the question $(x_0, \hat{f}_\lambda(x_0))$, and let $y_0 = \hat{f}_\lambda(x_0)$ so we have:

$$\hat{f}'_\lambda(x_0) = S'_\lambda y' = \sum_{j=1}^{N} S_\lambda(i,j)y_j + S_\lambda(i,0)y_0 \tag{17}$$

We can subtract equation (16) and (17) to get:

$$\begin{aligned} \hat{f}_\lambda(x_0) - \hat{f}'_\lambda(x_0) &= \sum_{j=1}^{N} S_\lambda(i,j)y_j + S_\lambda(i,0)\hat{f}_\lambda(x_0) - (\sum_{j=1}^{N} S_\lambda(i,j)y_j + S_\lambda(i,0)y_0) \\ &= S_\lambda(i,0)\hat{f}_\lambda(x_0) - S_\lambda(i,0)y_0 \end{aligned} \tag{18}$$

so

$$\hat{f}_\lambda(x_0) = \frac{\hat{f}_\lambda(x_0) - S_\lambda(i,0)y_0}{1 - S_\lambda(i,0)} \tag{19}$$

Hence, we have:

$$y_0 - \hat{f}_\lambda(x_0) = y_0 - \frac{\hat{f}_\lambda(x_0) - S_\lambda(i,0)y_0}{1 - S_\lambda(i,0)} = \frac{y_0 - \hat{f}_\lambda(x_0)}{1 - S_\lambda(i,0)} \tag{20}$$

Stats790 A3

Hanwen Ju

2023-03-05

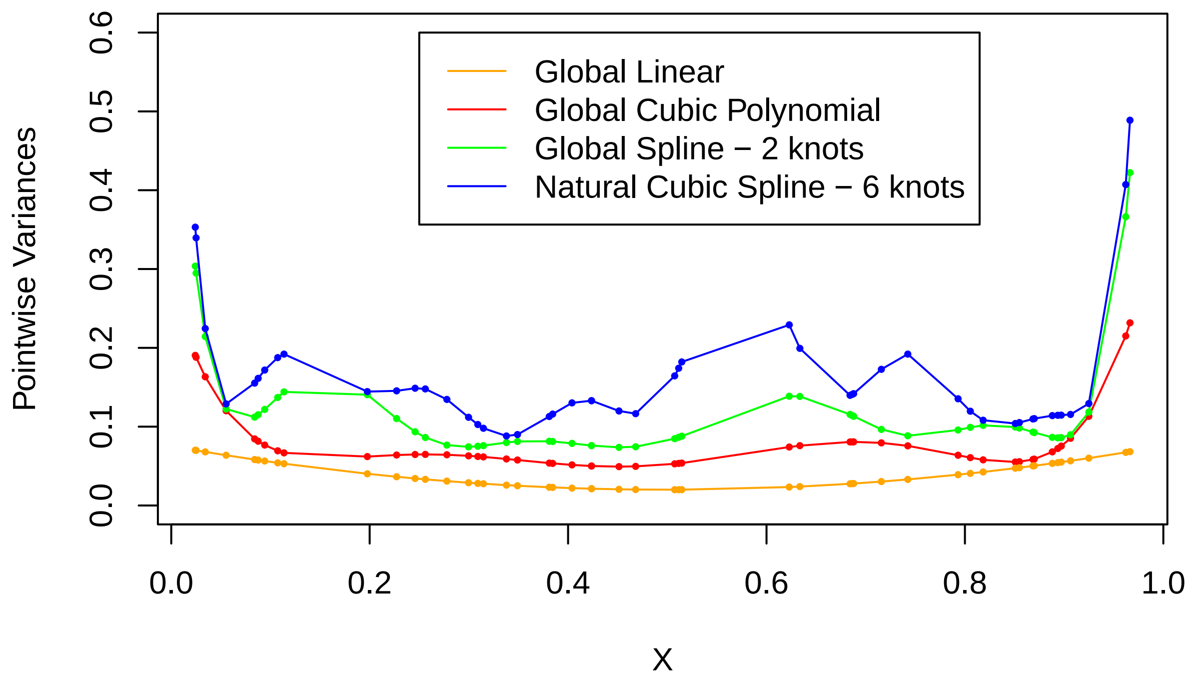
Question 1

```
#ESL figure 5.3
library(splines)

x <- runif(50)

#Pointwise variance

pvar <- function(X) {diag(X%*%solve(t(X)%*%X)%*%t(X))}


#linear
bs_linear <- bs(x, degree = 1, df=2, intercept = TRUE)
var_linear <- pvar(bs_linear)

#cubic polynomial
bs_cubic <- bs(x, degree = 3, df=4, intercept = TRUE)
var_cubic <- pvar(bs_cubic)

#2 knots cubic
bs_cubic2 <- bs(x, degree = 3, df=6, intercept = TRUE, knots = c(0.33, 0.66))
var_cubic2 <- pvar(bs_cubic2)

#6 knots natural
knots <- seq(0.1, 0.9, length.out = 6)[2:5]
bs_ns <- bs(x, degree = 3, intercept = TRUE, Boundary.knots = c(0.1,0.9), knots = knots)
```

```
## Warning in bs(x, degree = 3, intercept = TRUE, Boundary.knots = c(0.1, 0.9), :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
var_ns <- pvar(bs_ns)



plot(x, var_cubic, ylim = c(0,0.6),cex = 0.5, pch = 16, col = 'red',
     ylab = 'Pointwise Variances', xlab = 'X')
lines(x[order(x)], var_cubic[order(x)], col = "red")
points(x, var_linear, col='orange', cex = 0.5,pch = 16)
lines(x[order(x)], var_linear[order(x)], col = "orange")
points(x, var_cubic2, col = 'green', cex = 0.5,pch = 16)
lines(x[order(x)], var_cubic2[order(x)], col = "green")
points(x, var_ns, col = 'blue',cex = 0.5, pch = 16)
```

```
lines(x[order(x)], var_ns[order(x)], col = "blue")
legend(x=0.25,y= 0.6,legend = c('Global Linear', 'Global Cubic Polynomial', 'Global Spline - 2 knots',
```



Question 2

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

exclude_vars <- c("chd", "famhist")
numvars <- setdiff(names(dd), exclude_vars)
library(splines)
library(nnet)

#train/test set
train_sa <- dd[1:325, ]
test_sa <- dd[325:462, ]

train_tob <- train_sa$tobacco
test_tob <- test_sa$tobacco
```

```
#natural cubic spline####################################################################


#full_model, using all features:
spline_terms_ns <- sprintf("ns(%s, df = 6)", numvars)
ff_ns <- reformulate(c("famhist", spline_terms_ns), response = "chd")
full_model_ns <- glm(ff_ns, family = binomial, data = dd)
#reduced_model <- MASS::stepAIC(full_model_ns, direction = "backward")
#as.formula(model.frame(reduced_model))

#tobacco
```

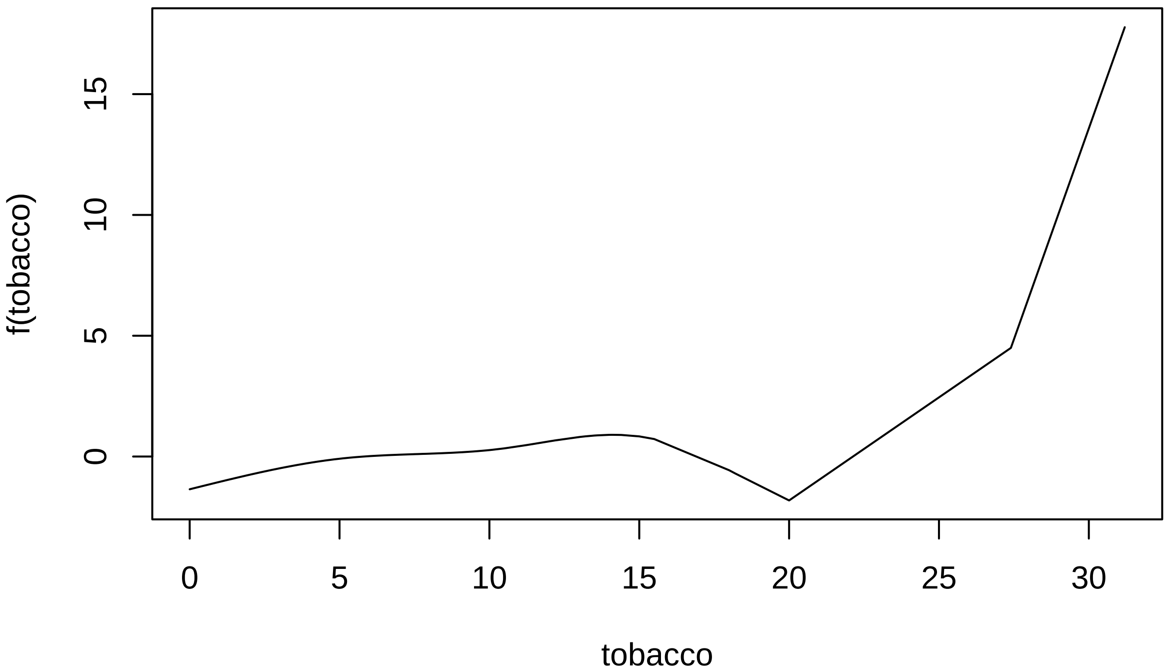
```r
knots_ns <- seq(min(train_tob), max(train_tob), length = 7)[2:6]
tob_ns <- ns(train_tob,df=6,knots = knots_ns)
#attributes(tob_ns)[c("degree", "knots", "Boundary.knots")]



#logistic regression
tob_model_ns <- multinom(train_sa$chd~tob_ns)#glm(train_sa$chd ~ tob_ns, family = binomial)
```

```
## # weights:  8 (7 variable)
## initial  value 225.272834
## iter  10 value 194.733689
## iter  20 value 194.676503
## iter  30 value 194.660710
## iter  40 value 194.654902
## iter  50 value 194.647519
## iter  60 value 194.633598
## iter  70 value 194.627033
## iter  80 value 194.621968
## iter  90 value 194.608691
## iter 100 value 194.601148
## final  value 194.601148
## stopped after 100 iterations
```

```r
tob_ns_coe <- coef(tob_model_ns)




#predict train
pred.ns <- as.matrix(cbind(rep(1,nrow(train_sa)),tob_ns)) %*% tob_ns_coe

pred.ns.logi <- exp(pred.ns)/(1+exp(pred.ns))
pred.ns.logi01 <- ifelse(pred.ns.logi > 0.5, 1, 0)
table(pred.ns.logi01,train_sa$chd)
```

```
##
## pred.ns.logi01   0   1
##              0 176  75
##              1  32  42
```

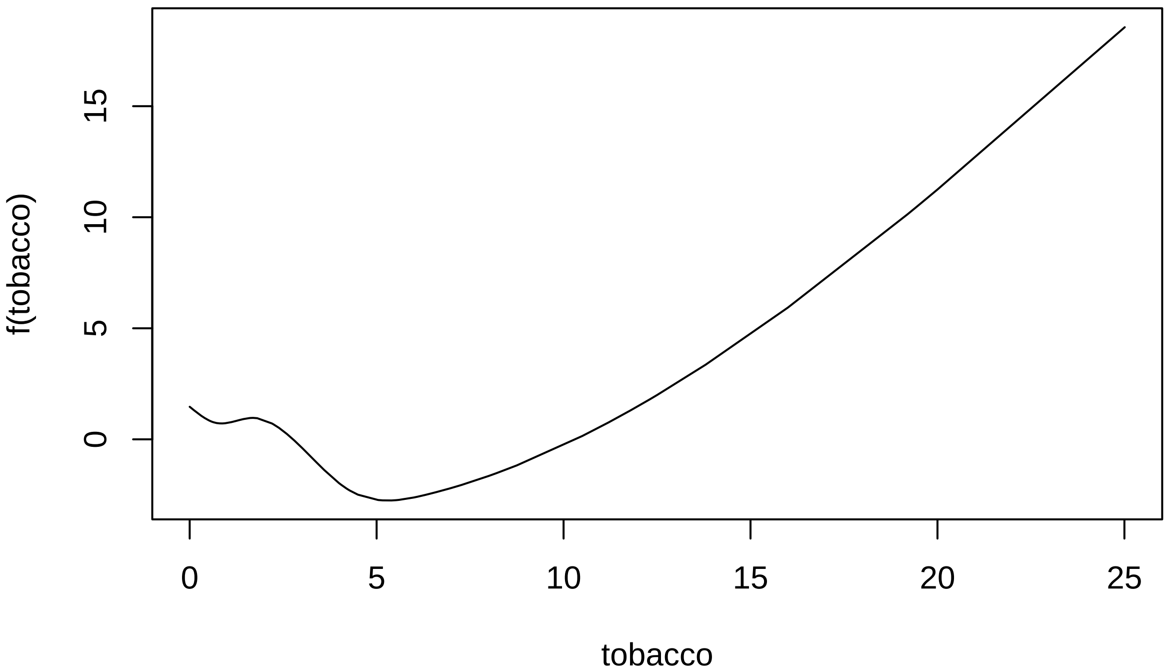
```r
vv=order(train_tob)
plot(sort(train_tob),pred.ns[vv],type='l',pch=19,xlab='tobacco',ylab='f(tobacco)')
```




```r
#predict test
test_tob_ns <- ns(test_tob, df = 6)
pred.ns.test <- as.matrix(cbind(rep(1,nrow(test_sa)),test_tob_ns)) %*% tob_ns_coe

pred.ns.logi.test <- exp(pred.ns.test)/(1+exp(pred.ns.test))
pred.ns.logi01.test <- ifelse(pred.ns.logi.test > 0.5, 1, 0)
table(pred.ns.logi01.test,test_sa$chd)
```

```
##
## pred.ns.logi01.test  0  1
##                   0 31 18
##                   1 64 25
```

```r
vv=order(test_tob)
plot(sort(test_tob),pred.ns.test[vv],type='l',pch=19,xlab='tobacco',ylab='f(tobacco)')
```




```
#b-spline##############################################################################

#full_model, using all features:
spline_terms_bs <- sprintf("bs(%s, df = 6)", numvars)
ff_bs <- reformulate(c("famhist", spline_terms_bs), response = "chd")
full_model_bs <- glm(ff_bs, family = binomial, data = dd)
#reduced_model <- MASS::stepAIC(full_model_ns, direction = "backward")
#as.formula(model.frame(reduced_model))

#tobacco
tob_bs <- bs(train_tob, df = 8)
attributes(tob_bs)[c("degree", "knots", "Boundary.knots")]
```

```
## $degree
## [1] 3
##
## $knots
## 16.66667% 33.33333%       50% 66.66667% 83.33333%
##      0.00      0.50      2.00      4.18      7.50
##
## $Boundary.knots
## [1]  0.0 31.2
```

```
#logistic regression
tob_model_bs <- multinom(train_sa$chd~tob_bs)
```

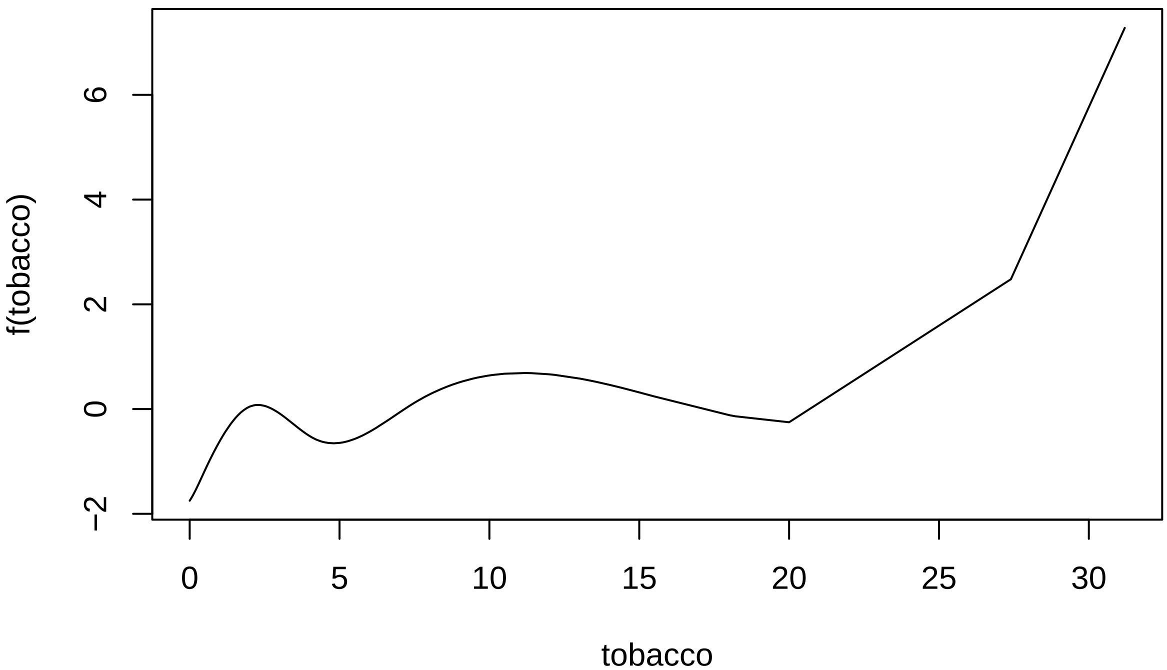
```
## # weights:  10 (9 variable)
## initial  value 225.272834
## iter  10 value 190.375280
## final  value 189.626304
## converged
```

```
tob_bs_coe <- coef(tob_model_bs)
```

```r
#predict train
pred.bs <- as.matrix(cbind(rep(1,nrow(train_sa)),tob_bs)) %*% tob_bs_coe

pred.bs.logi <- exp(pred.bs)/(1+exp(pred.bs))
pred.bs.logi01 <- ifelse(pred.bs.logi > 0.5, 1, 0)
table(pred.bs.logi01,train_sa$chd)
```

```
##
## pred.bs.logi01   0   1
##              0 175  69
##              1  33  48
```

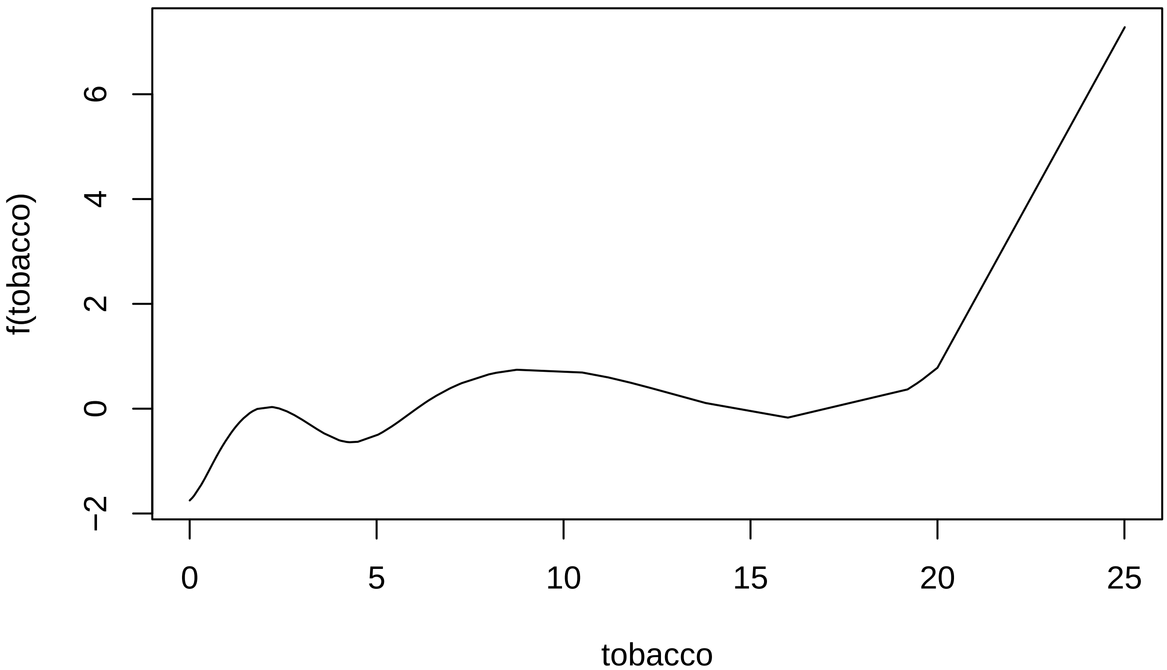
```r
vv=order(train_tob)
plot(sort(train_tob),pred.bs[vv],type='l',pch=19,xlab='tobacco',ylab='f(tobacco)')
```




```r
#predict test
test_tob_bs <- bs(test_tob, df = 8)
pred.bs.test <- as.matrix(cbind(rep(1,nrow(test_sa)),test_tob_bs)) %*% tob_bs_coe

pred.bs.logi.test <- exp(pred.bs.test)/(1+exp(pred.bs.test))
pred.bs.logi01.test <- ifelse(pred.bs.logi.test > 0.5, 1, 0)
table(pred.bs.logi01.test,test_sa$chd)
```

```
##
## pred.bs.logi01.test  0  1
##                   0 84 26
##                   1 11 17
```

```r
vv=order(test_tob)
plot(sort(test_tob),pred.bs.test[vv],type='l',pch=19,xlab='tobacco',ylab='f(tobacco)')
```




```
#truncated polynomial spline##########################################################################

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)
}


#tobacco
tob_tps <- truncpolyspline(train_tob, df = 7)
```

```
## Loading required package: Matrix
```

```
#logistic regression
tob_model_tps <- multinom(train_sa$chd~tob_tps)
```

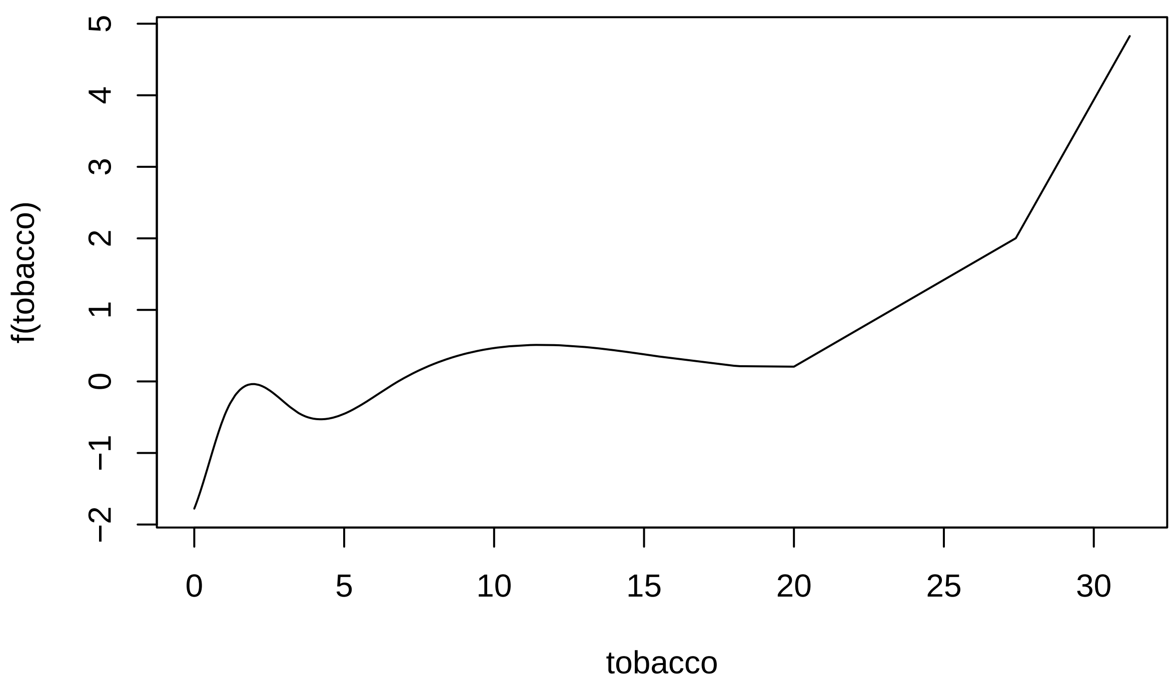
```
## # weights:  9 (8 variable)
## initial  value 225.272834
## iter  10 value 190.894718
## final  value 190.880707
## converged
```

```
tob_tps_coe <- coef(tob_model_tps)
```

```r
#predict train
pred.tps <- as.matrix(cbind(rep(1,nrow(train_sa)),tob_tps)) %*% tob_tps_coe

pred.tps.logi <- exp(pred.tps)/(1+exp(pred.tps))
pred.tps.logi01 <- ifelse(pred.tps.logi > 0.5, 1, 0)
table(pred.tps.logi01,train_sa$chd)
```

```
##
## pred.tps.logi01   0   1
##               0 183  78
##               1  25  39
```

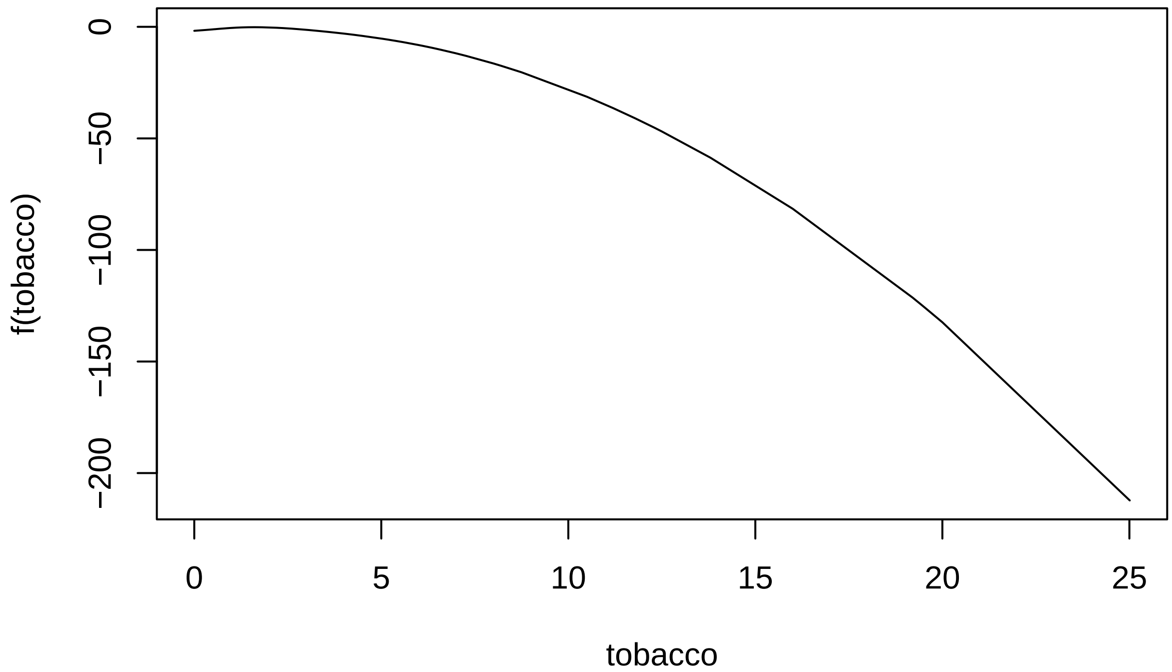
```r
vv=order(train_tob)
plot(sort(train_tob),pred.tps[vv],type='l',pch=19,xlab='tobacco',ylab='f(tobacco)')
```




```r
#predict test
test_tob_tps <- truncpolyspline(test_tob, df = 7)
pred.tps.test <- as.matrix(cbind(rep(1,nrow(test_sa)),test_tob_tps)) %*% tob_tps_coe

pred.tps.logi.test <- exp(pred.tps.test)/(1+exp(pred.tps.test))
pred.tps.logi01.test <- ifelse(pred.tps.logi.test > 0.5, 1, 0)
table(pred.tps.logi01.test,test_sa$chd)
```

```
##
## pred.tps.logi01.test  0  1
##                    0 95 43
```

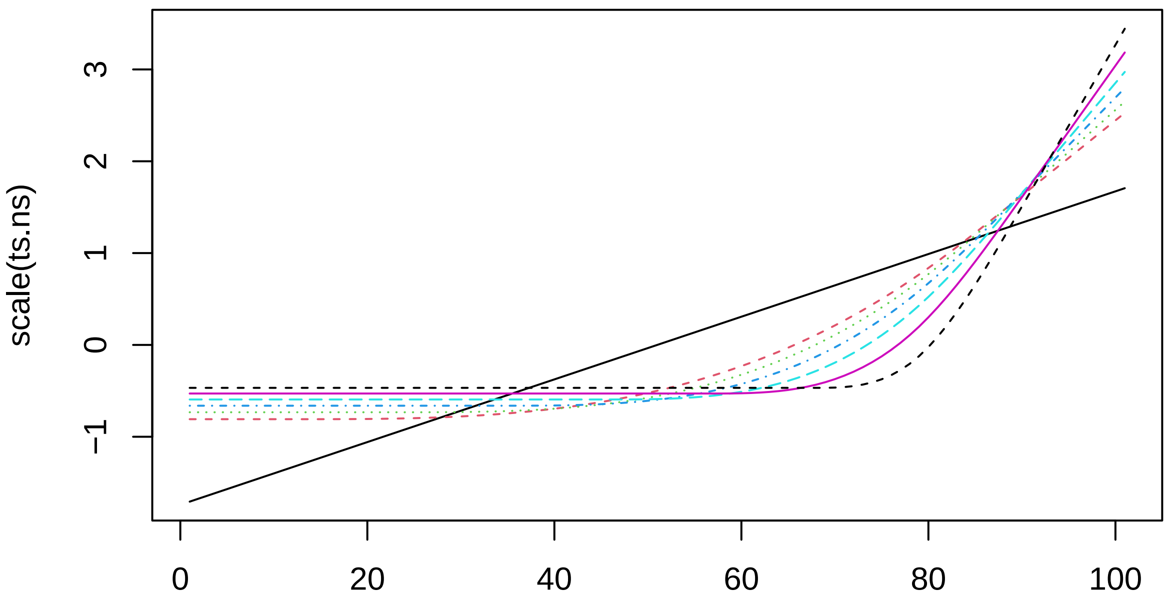
```r
vv=order(test_tob)
plot(sort(test_tob),pred.tps.test[vv],type='l',pch=19,xlab='tobacco',ylab='f(tobacco)')
```



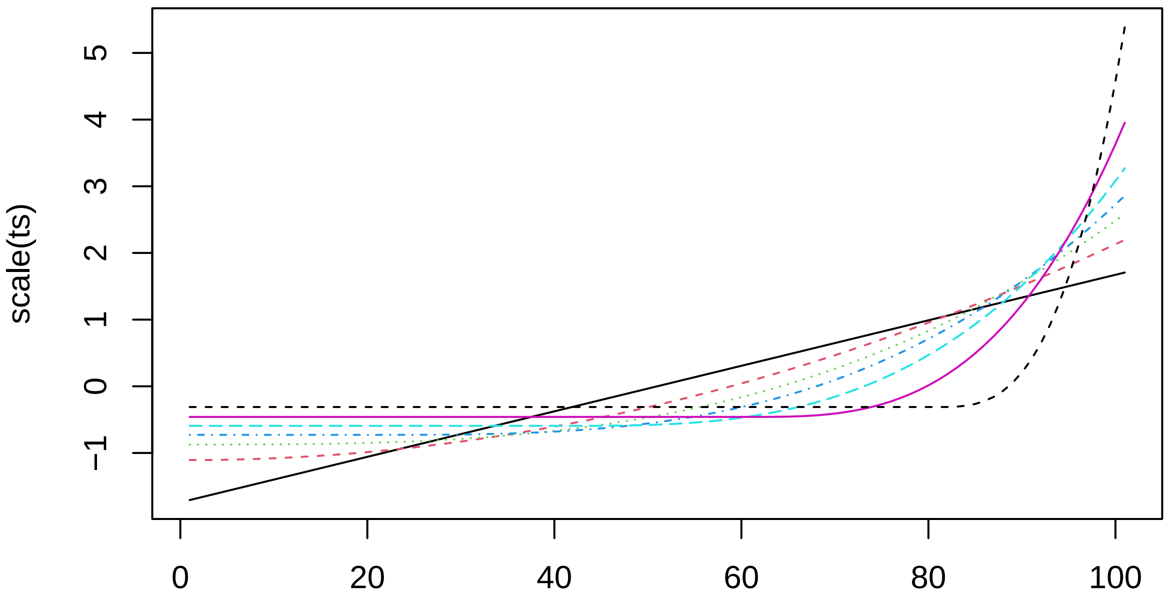
Question 3

```
library(Matrix)
truncpolyspline <- function(x, df,natrual) {
  if (!require("Matrix")) stop("need Matrix package")

# if natural cubic spline
if (natrual == TRUE){

  knots <- quantile(x, seq(0, 1, length = df+3))
  trunc_fun <- function(k) (x>=k)*(x-k)^3

  numberKnots <- df+2

  xiK <- knots[df+2] #last region

  SN <- x


  end <- df - 1
  for (i in 1:end){
    #print(j)
    dk <- (trunc_fun(knots[i+1])-trunc_fun(xiK))/(xiK-knots[i+1])
    dKm1 <- (trunc_fun(knots[df+1])-trunc_fun(xiK))/(xiK-knots[df+1])   #K-1
    SN <- cbind(SN,dk-dKm1)
  }
  return(SN)

}else{# cubic spline

  ## should probably use seq() instead of `:`
  ## dim: n x (df-2)
```

```r
  knots <- quantile(x, seq(0, 1, length = df - 1))
  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)
}

}




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

ts.ns <- truncpolyspline(xvec, df = 7,natrual = TRUE)
ts <- truncpolyspline(xvec, df = 7,natrual = FALSE)

matplot(scale(ts.ns), type = "l")
```

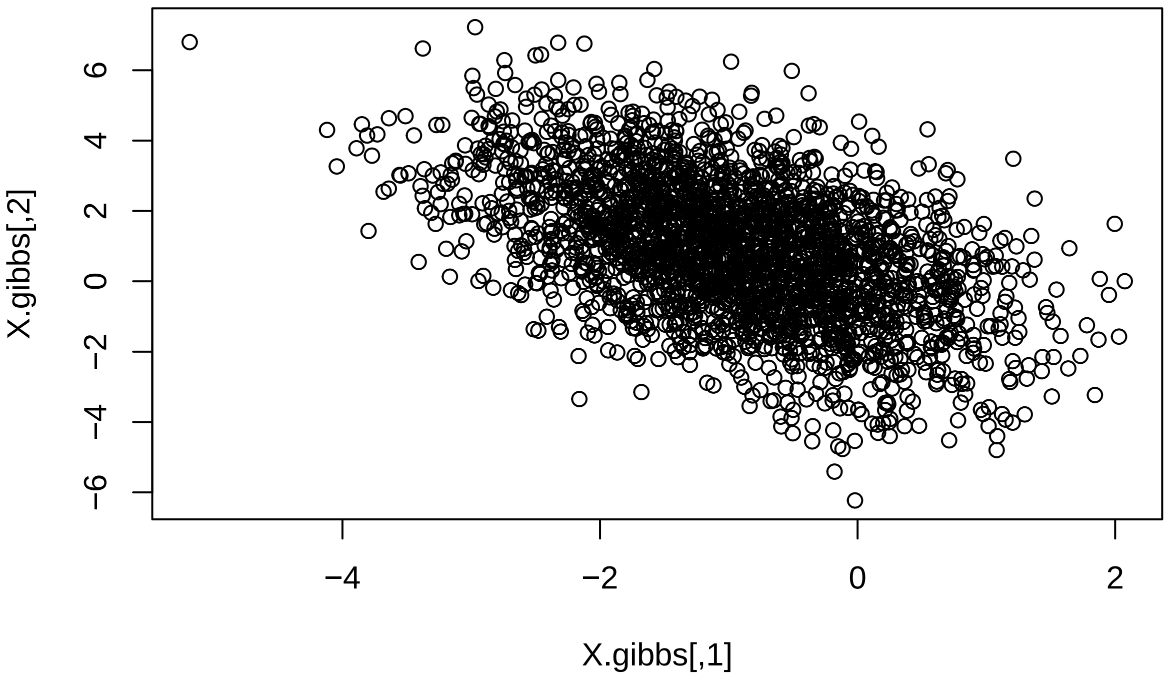



```r
matplot(scale(ts), type = "l")
```



Question 4

```
rbinorm.gibbs <- function(N, mu1, mu2, s1, s2, rho, burnin=0) {
  # Function to generate bivariate normals by Gibbs sampling algorithm(From Stats4CI3)

  # Arguments:
  # N - the required length of the chain
  # mu1, mu2 - the marginal means
  # s1, s2 - the marginal standard deviations
  # rho - the correlation between the components
  # burnin - the number of burn-in iterations

  x0 <- rnorm(2, c(mu1, mu2), c(s1,s2))
  # calculate the conditional standard deviations
  s1c <- s1*sqrt(1-rho^2)
  s2c <- s2*sqrt(1-rho^2)
  X <- matrix(NA, nrow=N+1+burnin, ncol=2)
  X[1,] <- x0
  for (i in 1:(N+burnin)) {
    X[i+1,1] <- rnorm(1, mu1+rho*s1/s2*(X[i,2]-mu2), s1c)
    X[i+1,2] <- rnorm(1, mu2+rho*s2/s1*(X[i+1,1]-mu1), s2c)
  }
  X[-(1:(burnin+1)),]
}

X.gibbs <- rbinorm.gibbs(2500, -1, 1, 1, 2, -0.5, 2000)

plot(X.gibbs)
```


X.gibbs[,2]
6
4
2
0
−2
−4
−6
−4
−2
0
2
X.gibbs[,1]