Solution to Homework 7

Jianliang He

2023/5/23

Problem 1

From definition, we have $L_{ij} = l_j(x_i)$ and for any x, $\sum_{j=1}^n l_j(x) = 1$

$$
\begin{aligned}
Y_i - \hat{r}_{(-i)}(x_i) &= Y_i - \Big(\sum_{j=1}^n \frac{l_j(x_i)}{\sum_{k\neq i} l_k(x_i)} Y_j - \frac{l_i(x_i)}{\sum_{k\neq i} l_k(x_i)} Y_i\Big) \\
&= \Big(1 + \frac{L_{ii}}{1-L_{ii}}\Big) Y_i - \sum_{j=1}^n \frac{L_{ij}}{1-L_{ii}} Y_j \\
&= \frac{Y_i - \hat{r}_n(x_i)}{1-L_{ii}}
\end{aligned}
$$

Therefore, we have proved Theorem (5.34)

$$
\hat{R}(h) = \frac{1}{n}\sum_{i=1}^n (Y_i - \hat{r}_{(-i)}(x_i))^2 = \frac{1}{n}\sum_{i=1}^n \Big(\frac{(Y_i - \hat{r}_n(x_i))}{(1-L_{ii})}\Big)^2
$$

■

Problem 2

```
glass <- read.table("https://www.stat.cmu.edu/~larry/all-of-nonpar/=dat
a/glass.dat")
Y <- glass$RI; X <- glass$Al
Y <- Y[order(X)]; X <- X[order(X)]
```

Regressogram

Fit Model

For simplicity, we order X and Y.

```
Y <- Y[order(X)]
X <- X[order(X)]
```

Here we use the below formula to calculate estimated risk.

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{r}_n(x_i)}{1 - L_{ii}}\right)^2$$

L_{ii} will change when our bandwidth of bins changes. We divide the interval $[a, b] = [\min X_i, \max X_i]$ into several equal spaced bins. For each space, we construct L_{ii} and calculate the estimated risk.

```
a <- min(X); b <- max(X); Y <- matrix(Y, ncol = 1)
n_try <- 2:15
record_risk <- c()
for (n in n_try){
  n_bins <- n
  cutoff <- seq(from = a, to = b, length.out = (n_bins+1))
  cutoff[1] <- cutoff[1] - 0.00001
  bins <- cut(X, cutoff)
  levels(bins) <- 1:n_bins
  L_matrix <- matrix(0, nrow = length(Y), ncol = length(Y))
  for (i in 1:length(Y)){
    L_matrix[i,bins == bins[i]] <- 1/sum(bins == bins[i])
  }
  r_hat <- L_matrix %*% Y
  diag_L <- diag(L_matrix)
  nominator_risk <- Y - r_hat
  denominator_risk <- 1 - diag_L
  risk <- (sum((nominator_risk / denominator_risk)^2, na.rm = T)) / (le
ngth(Y))
  record_risk <- c(record_risk, risk)
}
n_bins <- n_try[which.min(record_risk)]

cutoff <- seq(from = a, to = b, length.out = (n_bins+1))
cutoff[1] <- cutoff[1] - 0.00001
bins <- cut(X, cutoff)
levels(bins) <- 1:n_bins
L_matrix <- matrix(0, nrow = length(Y), ncol = length(Y))
for (i in 1:length(Y)){
  L_matrix[i,bins == bins[i]] <- 1/sum(bins == bins[i])
}
r_hat <- L_matrix %*% Y
```

Estimate Variance

We can estimate the variance by using the below formula.

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}\left(Y_i - \hat{r}(X_i)\right)^2}{n - 2v + \tilde{v}}$$

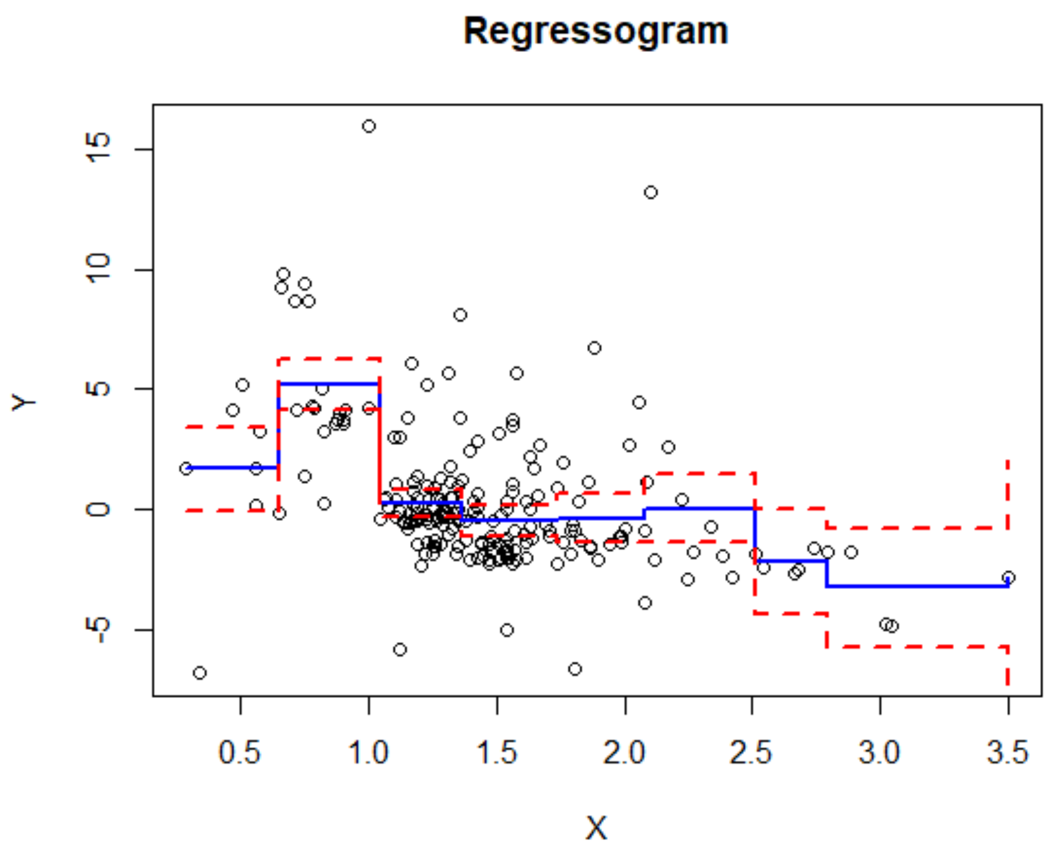
```
v <- sum(diag(L_matrix))
v_tilde <- sum(diag(t(L_matrix) %*% L_matrix))
sighat <- (sum((Y - r_hat)^2)) / (length(Y) - 2 * v + v_tilde)
```

Confidence Band

We use (5.100) to find c. Here we are only able to use numerical methodology to integrate $\kappa_0 = \int_a^b \| \quad |T'(x)| \quad \| dx$. In this way, we will only get $\kappa_0 = 0$ because $T_i{}'(x) = 0$. Therefore, we need to solve

$$2\big(1 - \Phi(c)\big) = 0.05$$

```
c <- qnorm(0.975)
upp_bond <- c()
low_bond <- c()
for (i in 1:length(Y)){
  upp_bond[i] <- r_hat[i,] + c * sqrt(sighat) * sqrt(sum((L_matrix[i,])
^2))
  low_bond[i] <- r_hat[i,] - c * sqrt(sighat) * sqrt(sum((L_matrix[i,])
^2))
}
plot(X, Y, main='Regressogram')
lines(X, r_hat, type = "s", lwd =2,  col = "blue")
lines(X, upp_bond, type = "s", lty =2, lwd =2,  col = "red")
lines(X, low_bond, type = "s", lty =2, lwd =2, col = "red")
```



Kernel

Fit Model

```
# dis stands for L1-norm; h is the tuning parameter
ker <- function(dis, h){
  return (1/(sqrt(2*pi)*h) * exp(-dis^2/(2*h^2)) )
}
ker <- Vectorize(ker)
```

Then similar as before we use the given formula to choose h.

```
h_all <- seq(from = 0.003, to = 0.3, length.out = 2000)
record_risk <- c()
for (h in h_all){
  L_matrix <- matrix(0, nrow = length(Y), ncol = length(Y))
  for (i in 1:length(Y)){
    dis <- X - X[i]
    L_matrix[i,] <- ker(dis = dis, h = h)
    L_matrix[i,] <- L_matrix[i,] / sum(L_matrix[i,])
  }
  r_hat <- L_matrix %*% Y
  diag_L <- diag(L_matrix)
```

```
  nominator_risk <- Y - r_hat
  denominator_risk <- 1 - diag_L
  risk <- (sum((nominator_risk / denominator_risk)^2, na.rm = T)) / (le
ngth(Y))
  record_risk <- c(record_risk, risk)
}
h <- h_all[which.min(record_risk)]
```

Using this h, we can fit our model:

```
x_grid <- seq(from = a, to = b, by = 0.001)
y_grid <- c()
for (k in 1:length(x_grid)){
  x_use <- x_grid[k]
  dis <- X - x_use
  L_krow <- ker(dis = dis, h = h)
  L_krow <- L_krow / sum(L_krow)
  y_grid[k] <- sum(L_krow * Y)
}
```

Estimate Variance

```
L_matrix <- matrix(0, nrow = length(Y), ncol = length(Y))
for (i in 1:length(Y)){
  dis <- X - X[i]
  L_matrix[i,] <- ker(dis = dis, h = h)
  L_matrix[i,] <- L_matrix[i,] / sum(L_matrix[i,])
}
r_hat <- L_matrix %*% Y
v <- sum(diag(L_matrix))
v_tilde <- sum(diag(t(L_matrix) %*% L_matrix))
sighat <- (sum((Y - r_hat)^2)) / (length(Y) - 2 * v + v_tilde)
```

Confidence Band

To begin with, we load a package to calculate the numerical derivative.

```
library(pracma)
options(warning = -1)
```

Next we write a function to calculate the κ_0.

```
Tprimenorm <- function(x){
  T_i <- c()
  for (i in 1:length(Y)) {
    get_l <- function(x){
      dis <- X - x
      l <- ker(dis = dis, h = h) / sum(ker(dis = dis, h = h))
      return(l[i])
    }
    T_i[i] <- fderiv(get_l,x)
  }
```

```
  return(sqrt(sum(T_i^2)))
}
Tprimenorm <- Vectorize(Tprimenorm)
int_result <- integrate(Tprimenorm,a,b)
print(int_result)

## 4.813632 with absolute error < 0.00048
```

Then we can solve the related c. (5.100)

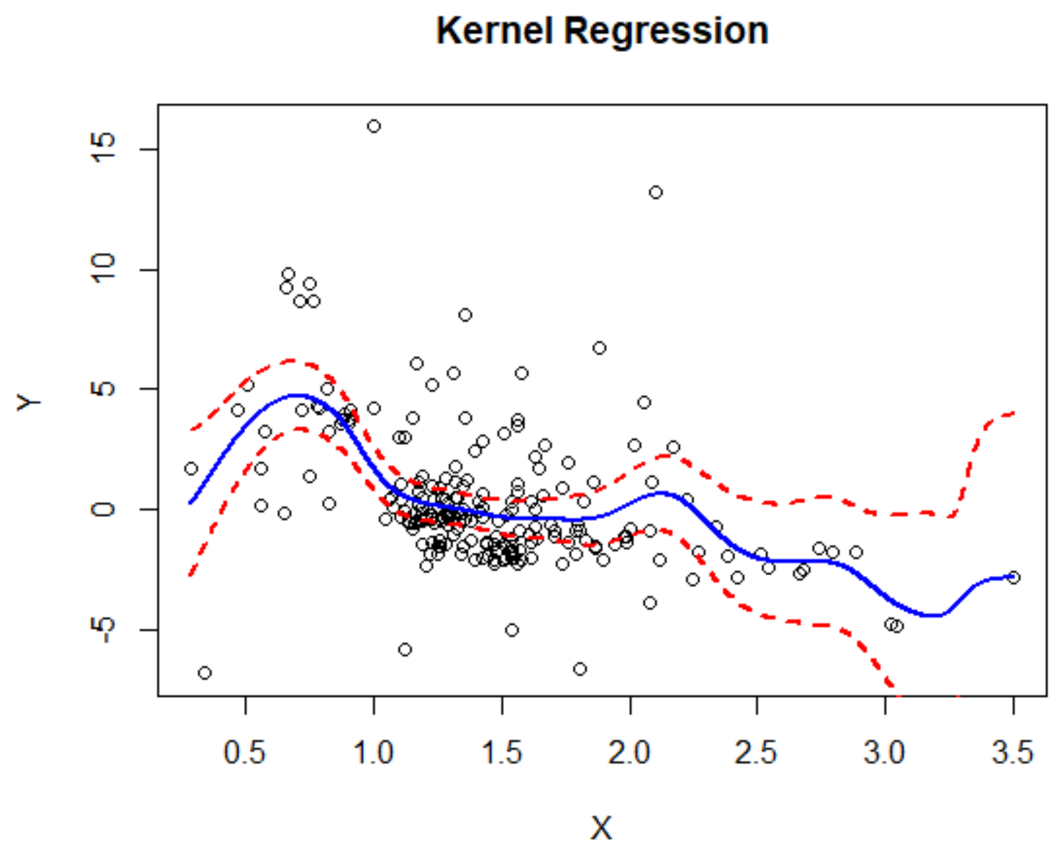
```
kappa0 <- int_result$value
obj_function <- function(c){
  return ((2 * (1 - pnorm(c)) + (kappa0 * exp(- (c^2) /2)) / pi - 0.05)
^2)
}
optimize(obj_function, c(2,3))

## $minimum
## [1] 2.677033
##
## $objective
## [1] 7.651029e-13

c_opt <- optimize(obj_function, c(2,3))$minimum
```

Finally, the confidence band is given by the following codes.

```
x_grid <- seq(from = a, to = b, by = 0.001)
y_grid <- c()
low_grid <- c()
upp_grid <- c()
for (k in 1:length(x_grid)){
  x_use <- x_grid[k]
  dis <- X - x_use
  L_krow <- ker(dis = dis, h = h)
  L_krow <- L_krow / sum(L_krow)
  y_grid[k] <- sum(L_krow * Y)
  low_grid[k] <- sum(L_krow * Y) - c_opt * sqrt(sighat) * sqrt(sum((L_k
row)^2))
  upp_grid[k] <- sum(L_krow * Y) + c_opt * sqrt(sighat) * sqrt(sum((L_k
row)^2))
}
plot(X, Y, main="Kernel Regression")
lines(x_grid, y_grid, type = "s", lwd =2,  col = "blue")
lines(x_grid, low_grid, type = "s", lty =2, lwd =2,  col = "red")
lines(x_grid, upp_grid, type = "s", lty =2, lwd =2, col = "red")
```



Local Linear

Fit Model

Find bandwidth: The i-th row of L matrix can be calculated as: given $x = x_i$, i.e., at the i-th data point. Then Calculate the i-th row of $(X_x^T W_x X_x)^{-1} X_x^T W_x$.

```
h_all <- seq(from = 0.1, to = 0.4, by = 0.001)
record_risk <- c()
for (h in h_all){
  L_matrix <- matrix(0, nrow = length(Y), ncol = length(Y))
  for (i in 1:length(Y)){K0=13.8
    Xx <- matrix(c(rep(1, length(Y)), X - X[i]), ncol = 2, byrow = F)
    dis <- X[i] - X
    Wx <- diag(ker(dis = dis, h = h), nrow = length(Y))
    hat_matrix <- solve(t(Xx) %*% Wx %*% Xx) %*% t(Xx) %*% Wx
    L_matrix[i,] <- hat_matrix[1,]
  }
  r_hat <- L_matrix %*% Y
  diag_L <- diag(L_matrix)
  nominator_risk <- Y - r_hat
  denominator_risk <- 1 - diag_L
```

```
  risk <- (sum((nominator_risk / denominator_risk)^2, na.rm = T)) / (le
ngth(Y))
  record_risk <- c(record_risk, risk)
}
h <- h_all[which.min(record_risk)]
```

The fitted model is

```
x_grid <- seq(from = a, to = b, by = 0.001)
y_grid <- c()
for (k in 1:length(x_grid)){
  Xx <- matrix(c(rep(1, length(Y)), X - x_grid[k]), ncol = 2, byrow = F
)
  dis <- X - x_grid[k]
  Wx <- diag(ker(dis = dis, h = h), nrow = length(Y))
  hat_matrix <- solve(t(Xx) %*% Wx %*% Xx) %*% t(Xx) %*% Wx
  y_grid[k] <- hat_matrix[1,] %*% Y
}
```

Estimate Variance

```
L_matrix <- matrix(0, nrow = length(Y), ncol = length(Y))
for (i in 1:length(Y)){
  Xx <- matrix(c(rep(1, length(Y)), X - X[i]), ncol = 2, byrow = F)
  dis <- X[i] - X
  Wx <- diag(ker(dis = dis, h = h), nrow = length(Y))
  hat_matrix <- solve(t(Xx) %*% Wx %*% Xx) %*% t(Xx) %*% Wx
  L_matrix[i,] <- hat_matrix[1,]
}
r_hat <- L_matrix %*% Y
v <- sum(diag(L_matrix))
v_tilde <- sum(diag(t(L_matrix) %*% L_matrix))
sighat <- (sum((Y - r_hat)^2)) / (length(Y) - 2 * v + v_tilde)
```

Confidence Band

Similarly we first calculate κ_0 (5.100)

```
Tprimenorm <- function(x){
  T_i <- c()
  for (i in 1:length(Y)) {
    get_l <- function(x){
      dis <- X - x
      Wx <- diag(ker(dis = dis, h = h), nrow=length(Y), ncol=length(Y))
      Xx <- matrix(c(rep(1, length(Y)), dis), ncol = 2, byrow = F)
      l = (solve(t(Xx)%*%Wx%*%Xx)%*%t(Xx)%*%Wx)[1,]
      l = l/sqrt(sum(l^2))
      return(l[i])
    }
    T_i[i] <- fderiv(get_l,x)
  }
  return(sqrt(sum(T_i^2)))
```

```
}
Tprimenorm <- Vectorize(Tprimenorm)
int_result <- integrate(Tprimenorm,a,b)
```

Similarly we calculate c by using optimize function.

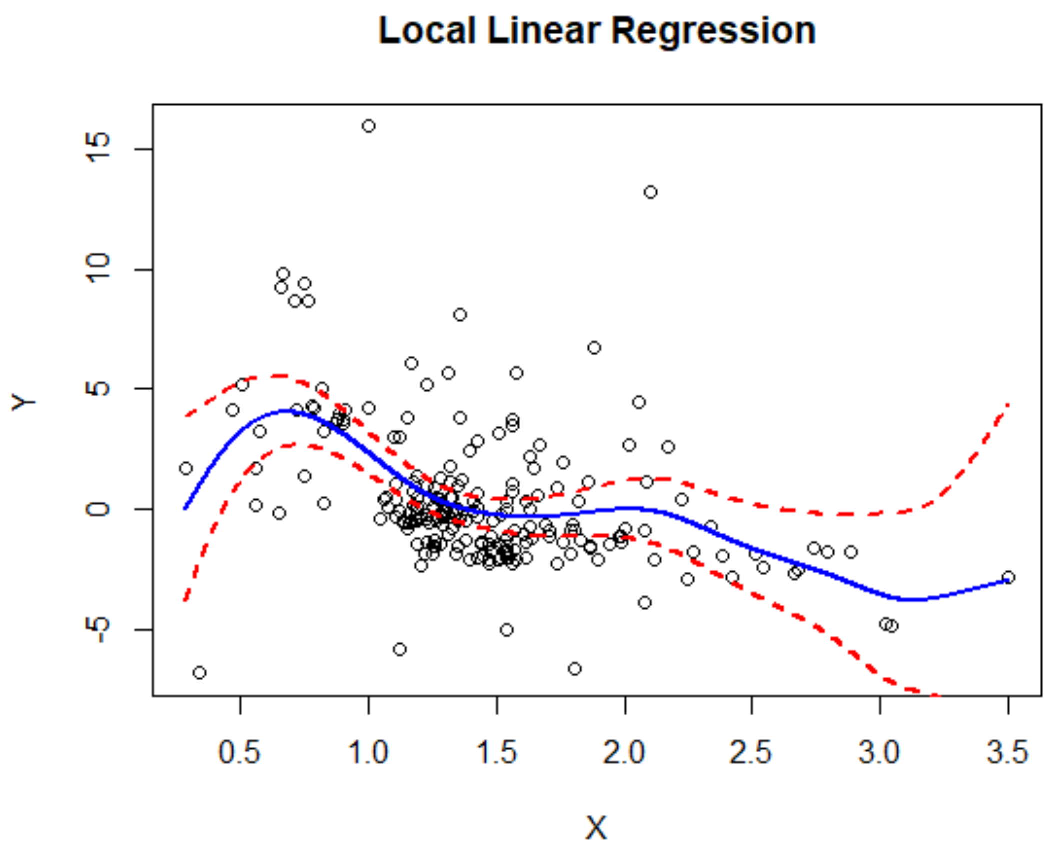
```
kappa0 <- int_result$value
obj_function <- function(c){
  return ((2 * (1 - pnorm(c)) + (kappa0 * exp(- (c^2) /2)) / pi - 0.05)
^2)
}
optimize(obj_function, c(2.7,3.2))

## $minimum
## [1] 2.910511
##
## $objective
## [1] 3.567086e-12

c_opt <- optimize(obj_function, c(2.7,3.2))$minimum
```

Finally the confidence band is as follows.

```
x_grid <- seq(from = a, to = b, by = 0.001)
y_grid <- c()
y_upp <- c()
y_low <- c()
for (k in 1:length(x_grid)){
  Xx <- matrix(c(rep(1, length(Y)), X - x_grid[k]), ncol = 2, byrow = F
)
  dis <- X - x_grid[k]
  Wx <- diag(ker(dis = dis, h = h), nrow = length(Y))
  hat_matrix <- solve(t(Xx) %*% Wx %*% Xx) %*% t(Xx) %*% Wx
  y_grid[k] <- hat_matrix[1,] %*% Y
  y_upp[k] <- hat_matrix[1,] %*% Y + c_opt * sqrt(sighat) * sqrt(sum((h
at_matrix[1,])^2))
  y_low[k] <- hat_matrix[1,] %*% Y - c_opt * sqrt(sighat) * sqrt(sum((h
at_matrix[1,])^2))
}
plot(X, Y, main="Local Linear Regression")
lines(x_grid, y_grid, type = "s", lwd =2,  col = "blue")
lines(x_grid, y_upp, type = "s", lty =2, lwd =2,  col = "red")
lines(x_grid, y_low, type = "s", lty =2, lwd =2, col = "red")
```



Spline

Fit Model

In the following codes, I will define matrix B and Ω step by step and finally find the optimal bandwidth h.

```
get_pos <- function(x){
  return(max(0, x))
}
get_pos <- Vectorize(get_pos)

n <- length(Y)
N <- length(Y) + 4

B <- matrix(0,length(Y),N)
B[,1:4] <- cbind(1,X,X^2,X^3)

for (i in 1:length(Y)) {
  ksai <- X[i]
  B[,i+4] <- get_pos((X-ksai)^3)
}
```

```r
xlist <- seq(a, b, length.out=500)

Bx <- matrix(0,500,N)
Bx[,1:4] <- cbind(1,xlist,xlist^2,xlist^3)

for (i in 1:n) {
  ksai <- X[i]
  Bx[,i+4] <- get_pos((xlist-ksai)^3)
}
get_B <- function(x){
  Bx <- matrix(0,length(x),N)
  Bx[,1:4]=cbind(1, x, x^2, x^3)
  for (i in 1:n) {
    ksai <- X[i]
    Bx[,i+4]=get_pos((x-ksai)^3)
  }
  return(Bx)
}

get_L_s <- function(lam){
  omega <- matrix(0,N,N)
  double_4 <- function(z){return(36*z^2)}
  double_34 <- function(z){return(12*z)}
  omega[3,3] <- 4*(b - a)
  omega[4,4] = integrate(double_4,a,b)$value
  omega[3,4] = integrate(double_34,a,b)$value
  omega[4,3] = omega[3,4]

  for(i in 5:N){
    ksaii = X[i-4]
    double_3i = Vectorize(function(z){
      return(max(6*(z-ksaii),0)*2)
    })
    double_4i = Vectorize(function(z){
      return(max(6*(z-ksaii),0)*6*z)
    })
    omega[3,i] = integrate(double_3i,a,b)$value
    omega[4,i] = integrate(double_4i,a,b)$value
    omega[i,3] = omega[3,i]
    omega[i,4] = omega[4,i]
  }
  for(i in 1:length(Y)){
    for(j in 1:i){
      ksai1 <- X[i]
      ksai2 <- X[j]
      double <- Vectorize(function(z){
        return(36*max((z-ksai1),0)*max((z-ksai2),0))
      })
```

```
      omega[i+4,j+4] = integrate(double,ksai1,b)$value
      omega[j+4,i+4] = omega[i+4,j+4]
    }
  }

  beta = solve(t(B)%*%B + lam*omega + diag(rep(0.0001,N)))%*%t(B)
  return(beta)
}
```

Calculate best lambda

```
risk_record <- c()
lamlist <- seq(0.01, 0.05, length.out=20)

get_Rh <- function(L){
  diag(L) <- 0
  for(i in 1:length(Y)){
    if(sum(L[i,1:length(Y)])){
      L[i,1:length(Y)] = L[i,1:length(Y)]/sum(L[i,1:length(Y)])
    }
  }
  rn <- as.vector(L%*%Y)
  return(mean((Y-rn)^2))
}

for(i in 1:20){
  risk_record[i] <- get_Rh(B%*%get_L_s(lamlist[i]))
}
```

Using the following lambda, we get the fitted model.

```
lam_best <- lamlist[which.min(risk_record)]
beta_best <- get_L_s(lam_best)
L <- B%*%beta_best
L_new <- Bx%*%beta_best
y_hat <- as.vector(L_new %*% Y)
y_hat2 <- as.vector(L %*% Y)
```

Variance Estimation

```
v <- sum(diag(L))
v_tilde <- sum(diag(t(L) %*% L))
sighat <- sum((Y - y_hat2)^2)/(n-2*v+v_tilde)
sighat

## [1] 6.462309
```

Confidence Band

```
get_Tprime <- function(x){
  T_p <- c()
  for (i in 1:length(Y)) {
    get_l <- function(x){
```

```
      l <- get_B(x)%*%beta_best
      li <- l[i]/sqrt(sum(l^2))
      return(li)
    }
    T_p[i] <- fderiv(get_l,x)
  }
  return(sqrt(sum(T_p^2)))
}
get_Tprime <- Vectorize(get_Tprime)
K0 <- integrate(get_Tprime,a,b)$value
```

The optimal c is then

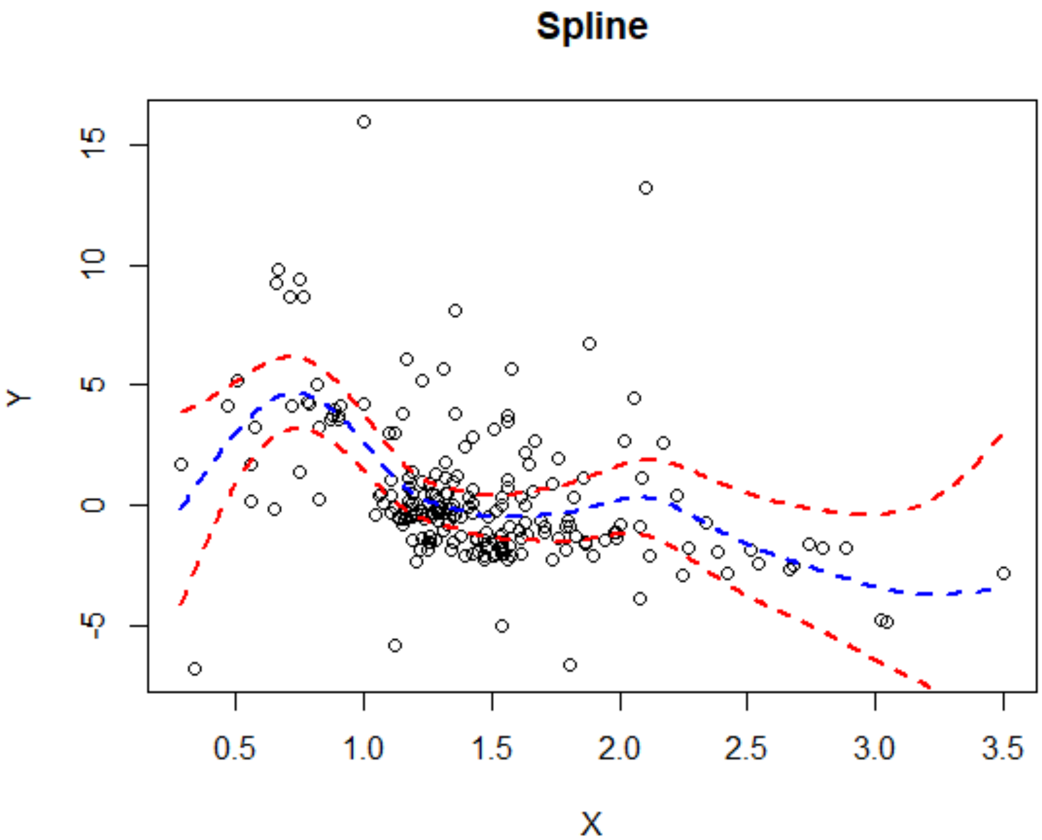
```
kappa0 <- K0
obj_function <- function(c){
  return ((2 * (1 - pnorm(c)) + (kappa0 * exp(- (c^2) /2)) / pi - 0.05)
^2)
}
optimize(obj_function, c(2.7,3.2))

## $minimum
## [1] 2.977185
##
## $objective
## [1] 2.687137e-12

c_opt <- optimize(obj_function, c(2.7,3.2))$minimum
```

Finally the confidence band is given by

```
upl= c()
lol= c()
for(i in 1:500){
  l_mode <- sqrt(sum(L_new[i,1:length(Y)]^2))
  upl[i] <- y_hat[i] + c_opt*sqrt(sighat)*l_mode
  lol[i] <- y_hat[i] - c_opt*sqrt(sighat)*l_mode
}
plot(X,Y,main="Spline")
lines(xlist[order(xlist)],y_hat[order(xlist)], lty =2, lwd =2,  col='bl
ue')
lines(xlist[order(xlist)],upl[order(xlist)], lty =2, lwd =2,  col='red'
)
lines(xlist[order(xlist)],lol[order(xlist)], lty =2, lwd =2,  col='red'
)
```


Spline
Y
X
15
10
5
0
-5
0.5
1.0
1.5
2.0
2.5
3.0
3.5