함수추정의 응용 및 실습

Project

서울대학교 통계학과 2017-11362 박건도

2022년 06월 19일

Q1. $\hat{\beta}(z)$, $\hat{m}(x,z)$, H.

(1) $\hat{\beta}(z)$.

z가 주어져 있을 때, Y, B, β, W를 다음과 같이 정의한다.

$$
\begin{aligned}
Y &= (Y_1,\, Y_2,\, \cdots,\, Y_n)^t \\
\beta &= \beta(z) = (\beta_1(z),\, \beta_2(z),\, \cdots,\, \beta_p(z))^t \\
W &= W(z) = diag(w_1,\, w_2,\, \cdots,\, w_n),\ where\, w_i = W(Z_i,\, z,\, \lambda) \\
B &= \begin{bmatrix} B_1(X_1) & B_2(X_1) & \cdots & B_p(X_1) \\ B_1(X_2) & B_2(X_2) & \cdots & B_p(X_2) \\ \vdots & \vdots & \ddots & \vdots \\ B_1(X_n) & B_2(X_n) & \cdots & B_p(X_n) \end{bmatrix}
\end{aligned}
$$

$\hat{\beta}(z) = \arg\min E_s(z)$라 하면 $E_s(z)$를 다음과 같이 쓸 수 있다.

$$
\begin{aligned}
E_s(z) &= \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{p} \beta_j B_j(X_i) \right)^2 W(Z_i,\, z,\, \lambda) \\
&= (Y - B\beta)^t W (Y - B\beta)
\end{aligned}
$$

위 식을 β에 대해 미분하여 0으로 놓고 풀면 $\hat{\beta}$를 구할 수 있다.

$$
\begin{aligned}
\hat{\beta}(z) &= (B^t W B)^{-1} B^t W Y \\
&= (B^t W(z) B)^{-1} B^t W(z) Y
\end{aligned}
$$

(2) $\hat{m}(x,z)$.

$\tilde{B} = (B_1(x),\, B_2(x),\, \cdots,\, B_p(x))^t$라 하자.

$$\begin{aligned}\hat{m}(x,z) &= \sum_{j=1}^{p} \hat{\beta}_j(z) B_j(x) \\ &= \tilde{B}\hat{\beta}(z) \\ &= \tilde{B}(B^tWB)^{-1}B^tWY\end{aligned}$$

(3) hat matrix H

먼저, $\hat{m}(X_i, Z_i)$를 구하면 다음과 같다.

$$\begin{aligned}\hat{m}(X_i,Z_i) &= \sum_{j=1}^{p} \hat{\beta}(Z_i) B_j(X_i) \\ &= \sum_{j=1}^{p} [B]_{ij} \left[(B^tW(Z_i)B)^{-1}B^tW(Z_i)Y\right]_j \\ &= \left[B(B^tW(Z_i)B)^{-1}B^tW(Z_i)Y\right]_i \\ &= \sum_{j=1}^{n} \left[B(B^tW(Z_i)B)^{-1}B^tW(Z_i)\right]_{ij} Y_j\end{aligned}$$

따라서, 우리는 $[H]_{ij}$를 구할 수 있다.

$$\begin{aligned}[H]_{ij} &= \left[B(B^tW(Z_i)B)^{-1}B^tW(Z_i)\right]_{ij} \\ &= \sum_{k=1}^{n} \left[B(B^tW(Z_i)B)^{-1}B^t\right]_{ik} [W(Z_i)]_{kj} \\ &= \left[B(B^tW(Z_i)B)^{-1}B^t\right]_{ij} [W(Z_i)]_{jj} \\ &= W(Z_i,\, Z_j,\, \lambda) \left[B(B^tW(Z_i)B)^{-1}B^t\right]_{ij}\end{aligned}$$

$j = i$를 대입하면 diagonal element 또한 구할 수 있다.

$$\begin{aligned}[H]_{ii} &= W(Z_i,\, Z_i,\, \lambda) \left[B(B^tW(Z_i)B)^{-1}B^t\right]_{ii} \\ &= \left[B(B^tW(Z_i)B)^{-1}B^t\right]_{ii}\end{aligned}$$

Q2. Bias, Variance of $\hat{m}(x, z)$.

(1) Bias

Bias를 구하기 전에, Y에 대한 정보가 주어졌으므로, $\hat{m}(x, z)$에 대해 다시 살펴보자.

$$
\begin{aligned}
m(x,z) &= \sum_{j=1}^{p} \beta_j(z) B_j(x) \\
m &= (m(X_1,\, Z_1),\, \cdots,\, m(X_n,\, Z_n))^t \\
\hat{m}(x,z) &= \tilde{B}(B^tWB)^{-1}B^tWY \\
&= \tilde{B}(B^tWB)^{-1}B^tW(m+\epsilon) \\
\mathbb{E}[\hat{m}(x,z)] &= \tilde{B}(B^tWB)^{-1}B^tWm
\end{aligned}
$$

따라서, bias는 다음과 같다.

$$
\begin{aligned}
Bias &= \mathbb{E}[\hat{m}(x,z)] - m(x,z) \\
&= \tilde{B}(B^tWB)^{-1}B^tWm - \tilde{B}\beta \\
&= \tilde{B}\left[(B^tWB)^{-1}B^tWm - \beta\right]
\end{aligned}
$$

Bias 식의 W는 z에 depend하고, m 안의 W는 Z_i에 depend하기 때문에, 더 간소화하기 어렵다.

(2) Variance

$$
\begin{aligned}
Variance &= \mathbb{E}\left[(\hat{m}(x,z) - \mathbb{E}\left[\hat{m}(x,z)\right])^2\right] \\
&= \mathbb{E}\left[\left(\tilde{B}(B^tWB)^{-1}B^tW(m+\epsilon) - \tilde{B}(B^tWB)^{-1}B^tWm\right)^2\right] \\
&= Var\left(\tilde{B}(B^tWB)^{-1}B^tW\epsilon\right) \\
&= \tilde{B}(B^tWB)^{-1}B^tW^2B(B^tWB)^{-1}\tilde{B}^t\sigma^2
\end{aligned}
$$

Q3. Cross Validation

```r
# Generate cubic B-spline functions with p bases
bs_gen <- function(X, p, boundary=c(0, 1)){
  knots <- seq(boundary[1], boundary[2], length=p-2) # equidistant knots
  bs(X, knots=knots[-c(1, p-2)], Boundary.knots=boundary, intercept=TRUE)
}

# Generate W(Zi, z, lambda)
W_gen <- function(Z, z, lambda, type="nominal"){
  if (type=="nominal"){
    diag(lambda^(Z != z))
  }else{
    diag(lambda^abs(Z-z))
  }
```

```
}
# Calculate mhat(x, z)
mhat <- function(x, Y, B, W, p){ # information of z is contained in W
  Bx <- bs_gen(x, p)
  Bx %*% ginv(t(B) %*% W %*% B) %*% t(B) %*% W %*% Y
}

# Caculate hatYi
hatYi <- function(i, Y, B, W){
  beta <- ginv(t(B) %*% W %*% B) %*% t(B) %*% W %*% Y
  sum(c(B[i,]) * beta)
}

# CV1
CV1 <- function(Y, X, Z, ps, lams, type="nominal"){
  n <- length(Y)
  cvs <- matrix(nrow=length(ps), ncol=length(lams), dimnames=list(ps, lams))
  for (xx in seq(ps)){
      p <- ps[xx]
      B <- bs_gen(X, p)
    for (yy in seq(lams)){
      lambda <- lams[yy]
      H <- matrix(nrow=n, ncol=n)
      for (zi in unique(Z)){
        W <- W_gen(Z, zi, lambda, type=type)
        H[Z==zi,] <- (B %*% ginv(t(B) %*% W %*% B) %*% t(B) %*% W)[Z==zi,]
      }
      hatY <- c(H %*% Y)
      cvs[xx, yy] <- sum((Y - hatY)^2/(1-diag(H))^2) / n
    }
  }
  cvs
}

CV2 <- function(Y, X, Z, ps, lams, type="nominal"){
  n <- length(Y)
  cvs <- matrix(nrow=length(ps), ncol=length(lams), dimnames=list(ps, lams))
  for (xx in seq(ps)){
```

```
      p <- ps[xx]
      B <- bs_gen(X, p)
    for (yy in seq(lams)){
      lambda <- lams[yy]
      cv_resid <- function(i){
        Bi <- B[-i,]
        Bx <- B[i,,drop=F]
        Wi <- W_gen(Z[-i], Z[i], lambda, type=type)
        mhati <- Bx %*% ginv(t(Bi) %*% Wi %*% Bi) %*% t(Bi) %*% Wi %*% Y[-i]
        (Y[i] - mhati)^2
      }
      cvs[xx,yy] <- mean(sapply(1:n, cv_resid))
    }
  }
  cvs
}
```

Q4. Simulation

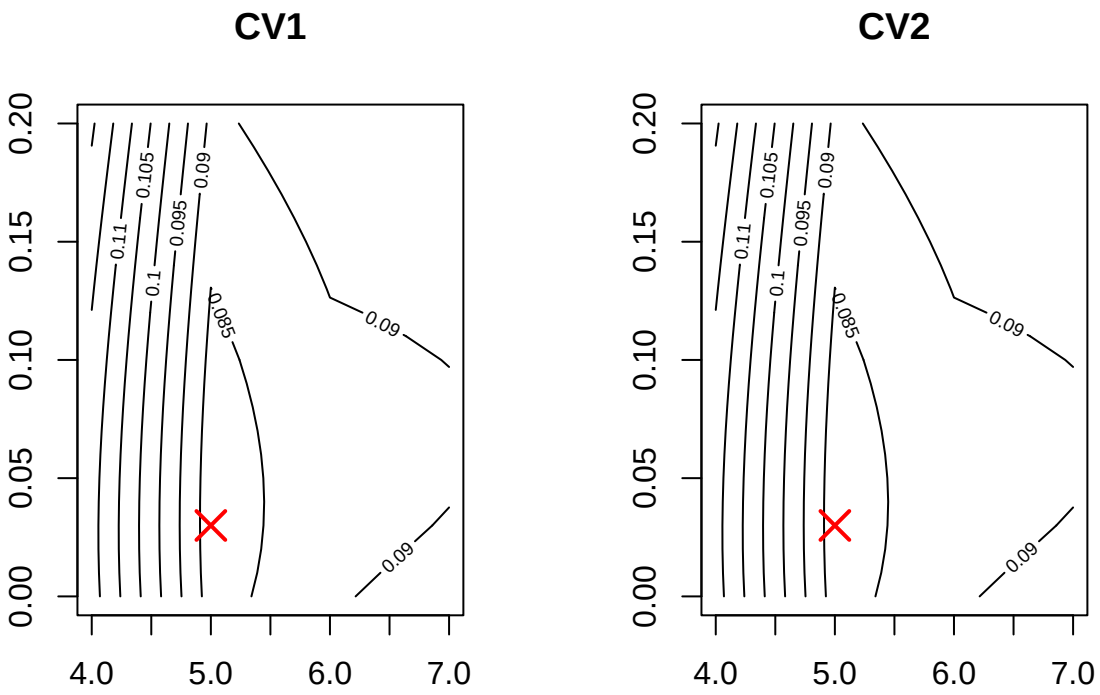
앞서 W를 정의할 때, categorical predictor Z가 nominal인지 ordinal인지에 따라 다른 값으로 두었는데, 해당 문제에서는 어떠한 정의를 써도 같은 값이 나오기 때문에, default로 설정한 nominal로 계산하였다.

```
# Data generation
n <- 100
set.seed(20221)
X <- runif(n)
set.seed(20222)
Z <- rbinom(n, 1, 0.5)
set.seed(20223)
eps1 <- rnorm(n, sd=0.25)
set.seed(20224)
eps2 <- rnorm(n, sd=0.5)
alpha <- c(0.5, 1)
```

(1) $\sigma = 0.25, \alpha = 0.5$.

```
# generate Y
Y <- cos(2 * pi * X) + alpha[1] * Z + eps1
```

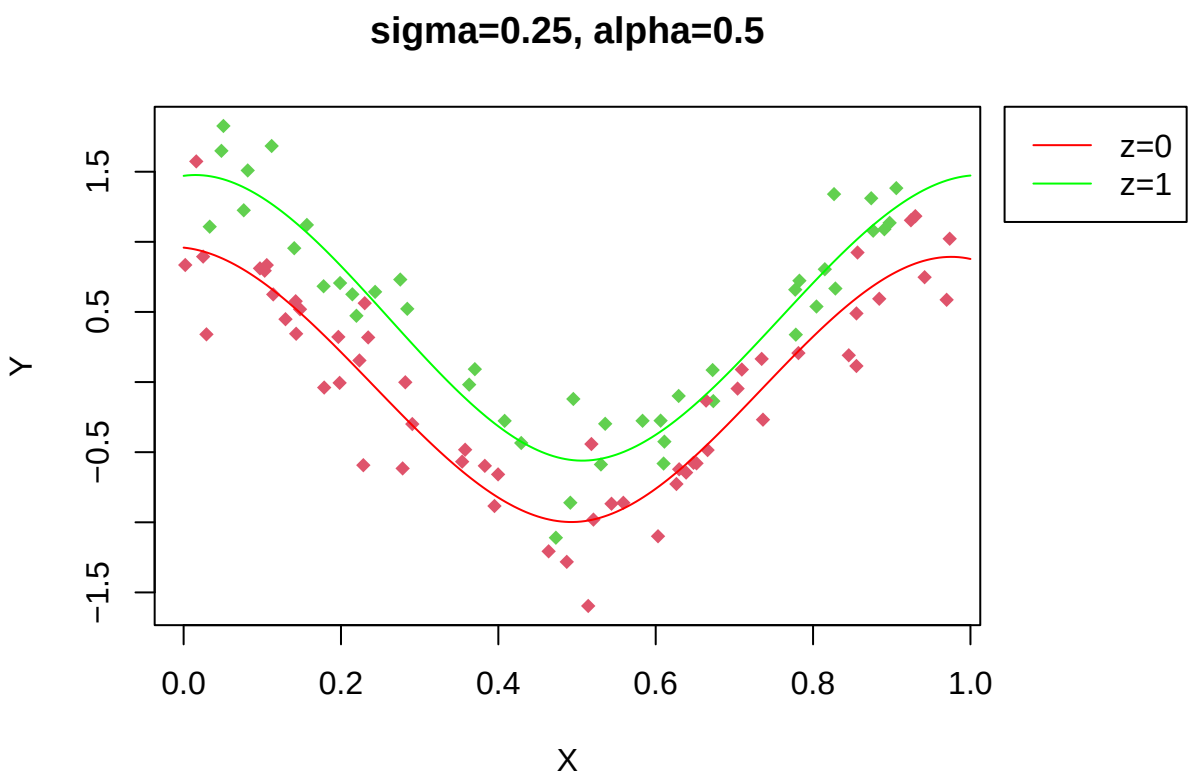
```r
# hyperparameter setting
ps <- c(4, 5, 6, 7)
lams <- seq(0, 0.2, by=0.01)

# Calculate CV
cvs1 <- CV1(Y, X, Z, ps, lams)
cvs2 <- CV2(Y, X, Z, ps, lams)

par(mfrow=c(1,2))
mincv1 <- min(cvs1)
ind <- which(cvs1==mincv1, arr.ind=TRUE)
contour(ps,lams, cvs1, main="CV1")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")

mincv2 <- min(cvs2)
ind <- which(cvs2==mincv2, arr.ind=TRUE)
contour(ps,lams, cvs2, main="CV2")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")
```



CV1과 CV2로 구한 결과가 같게 나왔으며, $p = 5, \lambda = 0.03$의 결과가 나왔다.

```r
par(mar = c(5, 5, 4, 6))
plot(X, Y, col=Z+2, pch=18, main="sigma=0.25, alpha=0.5")
p <- 5
```

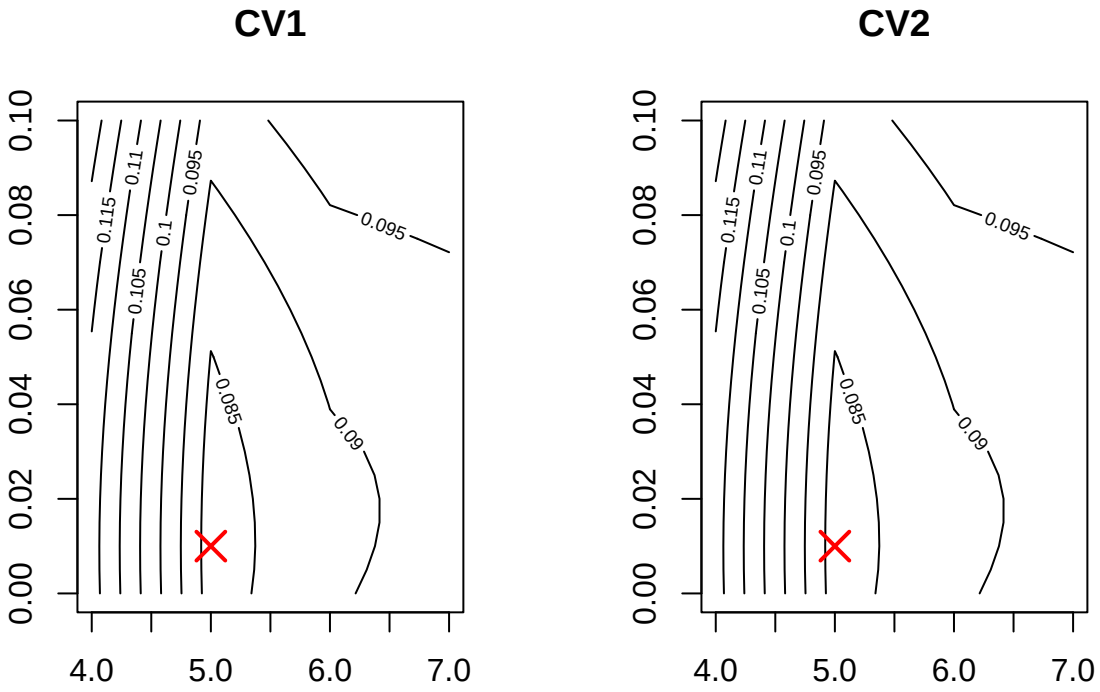
```
lambda <- 0.03
xx <- seq(0, 1, length=201)
B <- bs_gen(X, p)
W <- W_gen(Z, z=0, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="red")
W <- W_gen(Z, z=1, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="green")
legend(x="topright", legend=c("z=0", "z=1"), col=c("red", "green"),
       lty=1, inset=c(-0.25,0), xpd = TRUE)
```



(2) $\sigma = 0.25, \alpha = 0.1$**.**

```
# generate Y
Y <- cos(2 * pi * X) + alpha[2] * Z + eps1

# hyperparameter setting
ps <- c(4, 5, 6, 7)
lams <- seq(0, 0.1, by=0.005)
```

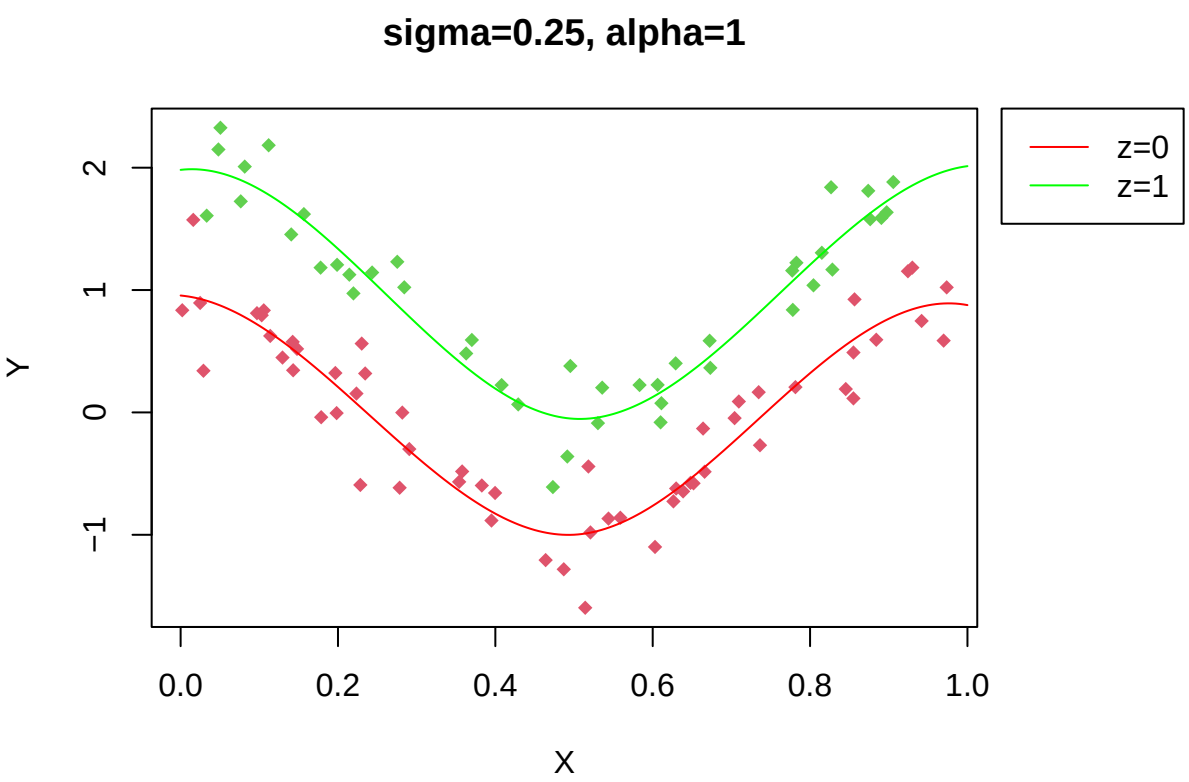
```
# Calculate CV
cvs1 <- CV1(Y, X, Z, ps, lams)
cvs2 <- CV2(Y, X, Z, ps, lams)

par(mfrow=c(1,2))
mincv1 <- min(cvs1)
ind <- which(cvs1==mincv1, arr.ind=TRUE)
contour(ps,lams, cvs1, main="CV1")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")

mincv2 <- min(cvs2)
ind <- which(cvs2==mincv2, arr.ind=TRUE)
contour(ps,lams, cvs2, main="CV2")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")
```



CV1과 CV2로 구한 결과가 같게 나왔으며, $p = 5, \lambda = 0.01$의 결과가 나왔다.

```
par(mar = c(5, 5, 4, 6))
plot(X, Y, col=Z+2, pch=18, main="sigma=0.25, alpha=1")
p <- 5
lambda <- 0.01
xx <- seq(0, 1, length=201)
B <- bs_gen(X, p)
W <- W_gen(Z, z=0, lambda=lambda)
```

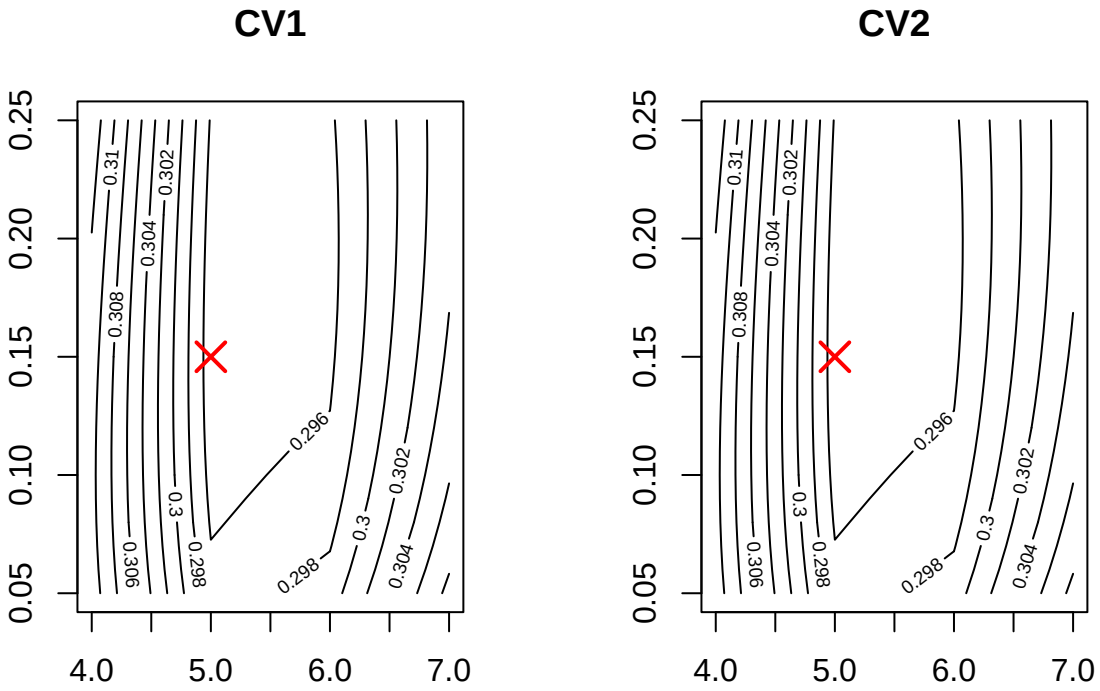
```
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="red")
W <- W_gen(Z, z=1, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="green")
legend(x="topright", legend=c("z=0", "z=1"), col=c("red", "green"),
       lty=1, inset=c(-0.25,0), xpd = TRUE)
```



(3) $\sigma = 0.5, \alpha = 0.5$**.**

```
# generate Y
Y <- cos(2 * pi * X) + alpha[1] * Z + eps2

# hyperparameter setting
ps <- c(4, 5, 6, 7)
lams <- seq(0.05, 0.25, by=0.01)

# Calculate CV
cvs1 <- CV1(Y, X, Z, ps, lams)
cvs2 <- CV2(Y, X, Z, ps, lams)
```

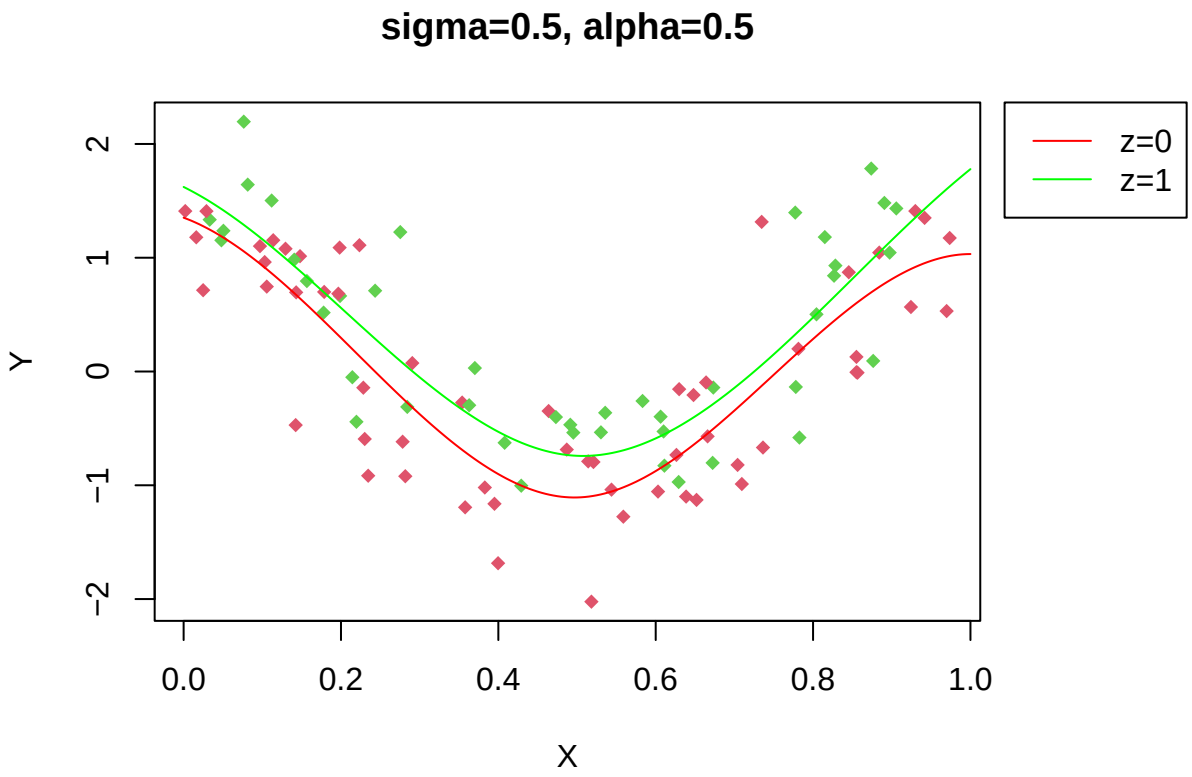
```
par(mfrow=c(1,2))
mincv1 <- min(cvs1)
ind <- which(cvs1==mincv1, arr.ind=TRUE)
contour(ps,lams, cvs1, main="CV1")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")

mincv2 <- min(cvs2)
ind <- which(cvs2==mincv2, arr.ind=TRUE)
contour(ps,lams, cvs2, main="CV2")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")
```



CV1과 CV2로 구한 결과가 같게 나왔으며, $p = 5, \lambda = 0.15$의 결과가 나왔다.

```
par(mar = c(5, 5, 4, 6))
plot(X, Y, col=Z+2, pch=18, main="sigma=0.5, alpha=0.5")
p <- 5
lambda <- 0.15
xx <- seq(0, 1, length=201)
B <- bs_gen(X, p)
W <- W_gen(Z, z=0, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="red")
W <- W_gen(Z, z=1, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
```

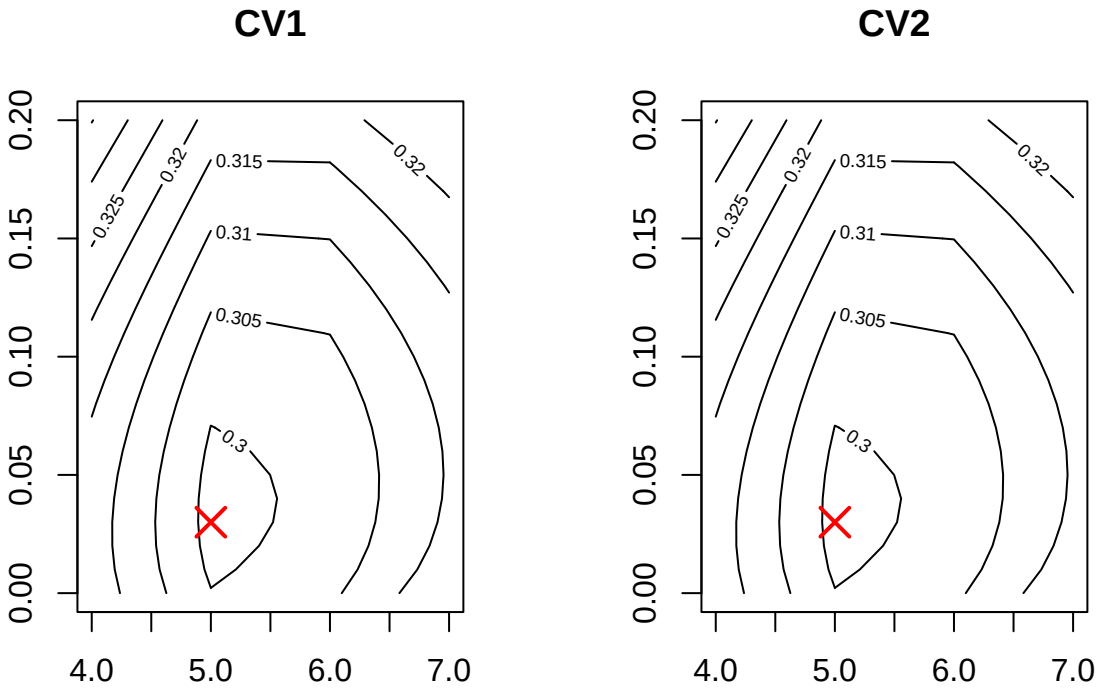
```
lines(xx, yy0, col="green")
legend(x="topright", legend=c("z=0", "z=1"), col=c("red", "green"),
       lty=1, inset=c(-0.25,0), xpd = TRUE)
```



(4) $\sigma = 0.5, \alpha = 1.$

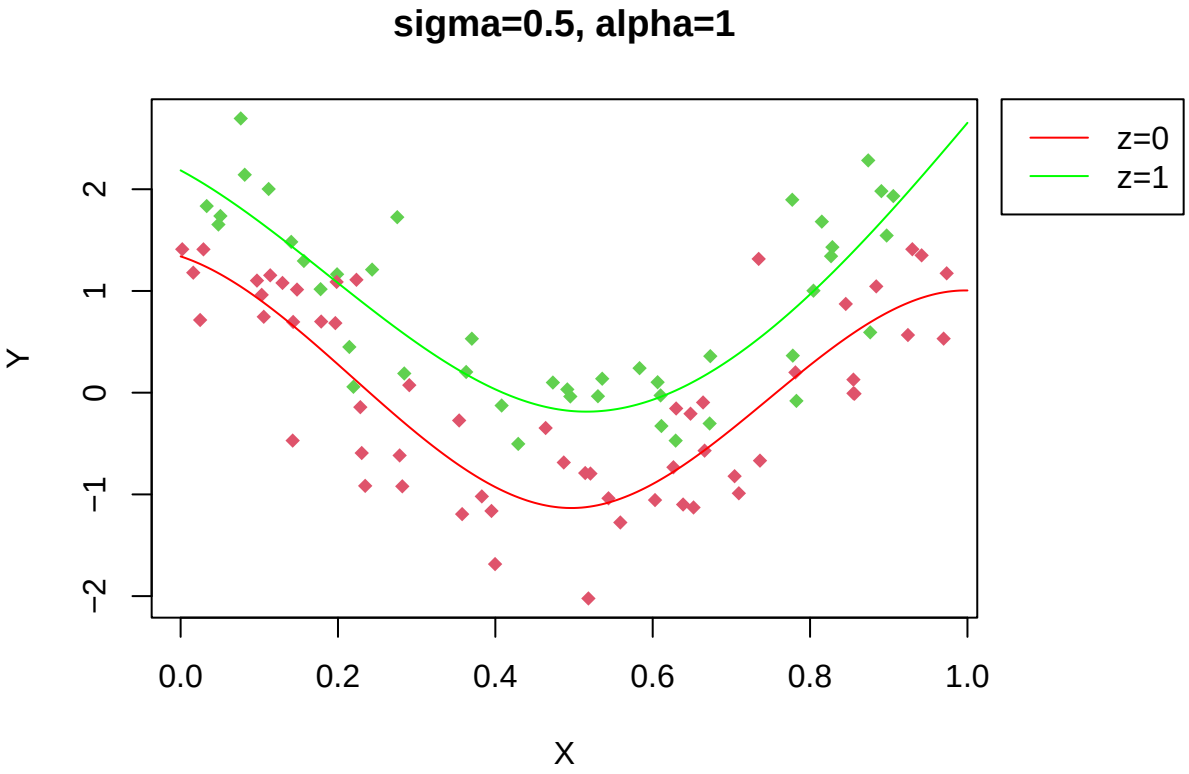
```
# generate Y
Y <- cos(2 * pi * X) + alpha[2] * Z + eps2

# hyperparameter setting
ps <- c(4, 5, 6, 7)
lams <- seq(0, 0.2, by=0.01)

# Calculate CV
cvs1 <- CV1(Y, X, Z, ps, lams)
cvs2 <- CV2(Y, X, Z, ps, lams)

par(mfrow=c(1,2))
mincv1 <- min(cvs1)
ind <- which(cvs1==mincv1, arr.ind=TRUE)
contour(ps,lams, cvs1, main="CV1")
```

```
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")

mincv2 <- min(cvs2)
ind <- which(cvs2==mincv2, arr.ind=TRUE)
contour(ps,lams, cvs2, main="CV2")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")
```



CV1과 CV2로 구한 결과가 같게 나왔으며, $p = 5, \lambda = 0.03$의 결과가 나왔다.

```
par(mar = c(5, 5, 4, 6))
plot(X, Y, col=Z+2, pch=18, main="sigma=0.5, alpha=1")
p <- 5
lambda <- 0.03
xx <- seq(0, 1, length=201)
B <- bs_gen(X, p)
W <- W_gen(Z, z=0, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="red")
W <- W_gen(Z, z=1, lambda=lambda)
yy0 <- sapply(xx, mhat, Y=Y, B=B, W=W, p=p)
lines(xx, yy0, col="green")
legend(x="topright", legend=c("z=0", "z=1"), col=c("red", "green"),
       lty=1, inset=c(-0.25,0), xpd = TRUE)
```



(1) ~ (4)의 결과를 종합해보자.

- CV1과 CV2로 구한 결과가 항상 일치하였다. CV1에서 사용한 식을 유도하는 과정에서 hat matrix H가 X에만 의존한다는 사실을 이용했었는데, Q1에서도 확인하였듯이 hat matrix가 predictor인 X, Z에만 의존하기 때문에 두 식이 같다는 결론을 내릴 수 있다.
- B-spline의 basis의 수는 $p = 5$로 일정하게 나왔다. 코사인 함수를 근사하는 데에 있어서 5개의 basis를 갖는 cubic B-spline이 충분해서 나온 결과로 추측된다.
- λ의 경우, 1이라면 categorical 데이터의 구분 없이 하나의 데이터로 사용해도 무방하다는 뜻이고, 0인 경우에는 해당 categorical 데이터 그룹만을 사용해서 추정할 수 있다는 뜻이다. 다시 말해서, λ가 0에 가까울 수록 데이터가 잘 분리되어있고, 서로가 서로를 설명하지 않는다는 뜻이다. $\sigma = 0.5, \alpha = 0.5$일 때 λ의 값이 가장 크게 나왔다. 그 이유는, 위에서 다룬 데이터 모형 중 분산이 가장 크고, category 항목 별 데이터가 서로 겹친 부분이 많아 데이터가 분리되어있다고 생각하기 어렵기 때문이다. 또한 $\sigma = 0.25, \alpha = 0.5$인 경우와 $\sigma = 0.5, \alpha = 1$인 경우의 λ 값이 같게 나왔다. 이 부분이 뜻하는 바는, σ가 커짐으로써 λ가 증가하도록 영향을 끼치는 것과 α가 커짐으로써 λ가 감소하도록 영향을 끼치는 것이 서로 상쇄되어 결국에는 λ가 유사한 값을 유지한 것으로 추측된다.

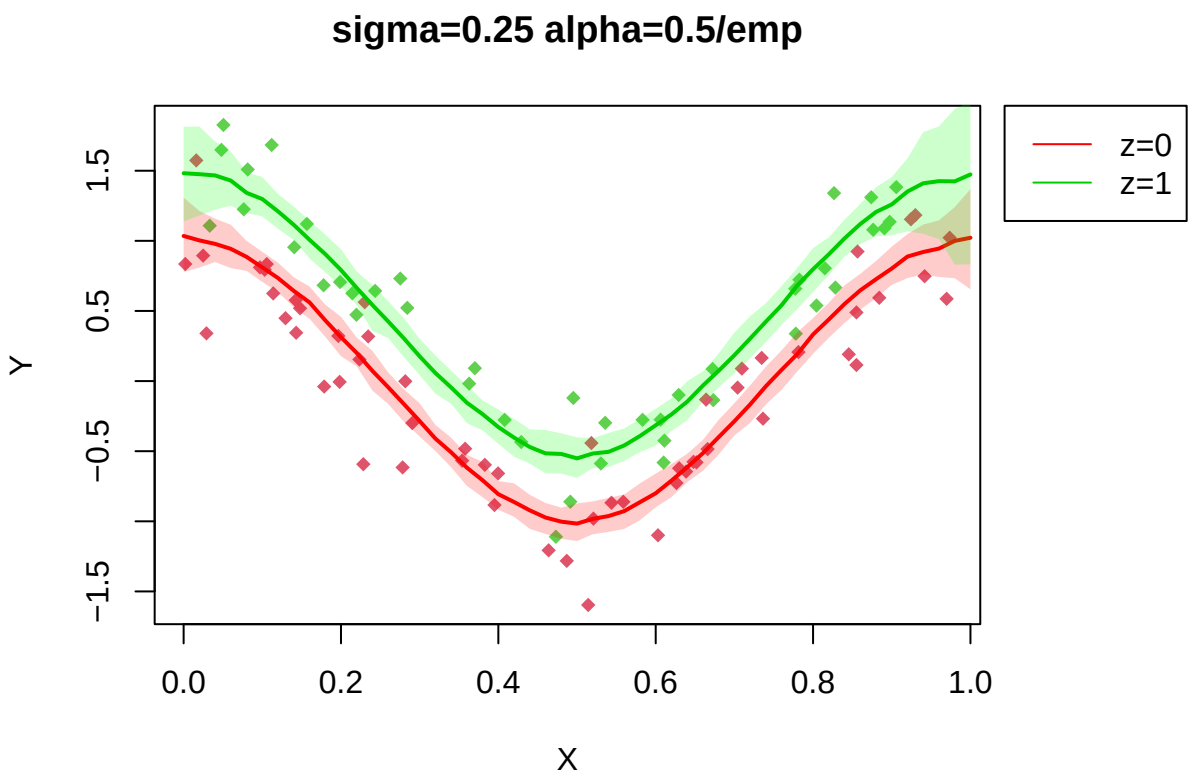
Q5. Confidence Interval

```
# Empirical generation(Monte Carlo)
CI_emp <- function(x, Y, B, W, p, sd){
  sim <- c()
  for (iter in 1:100){
```

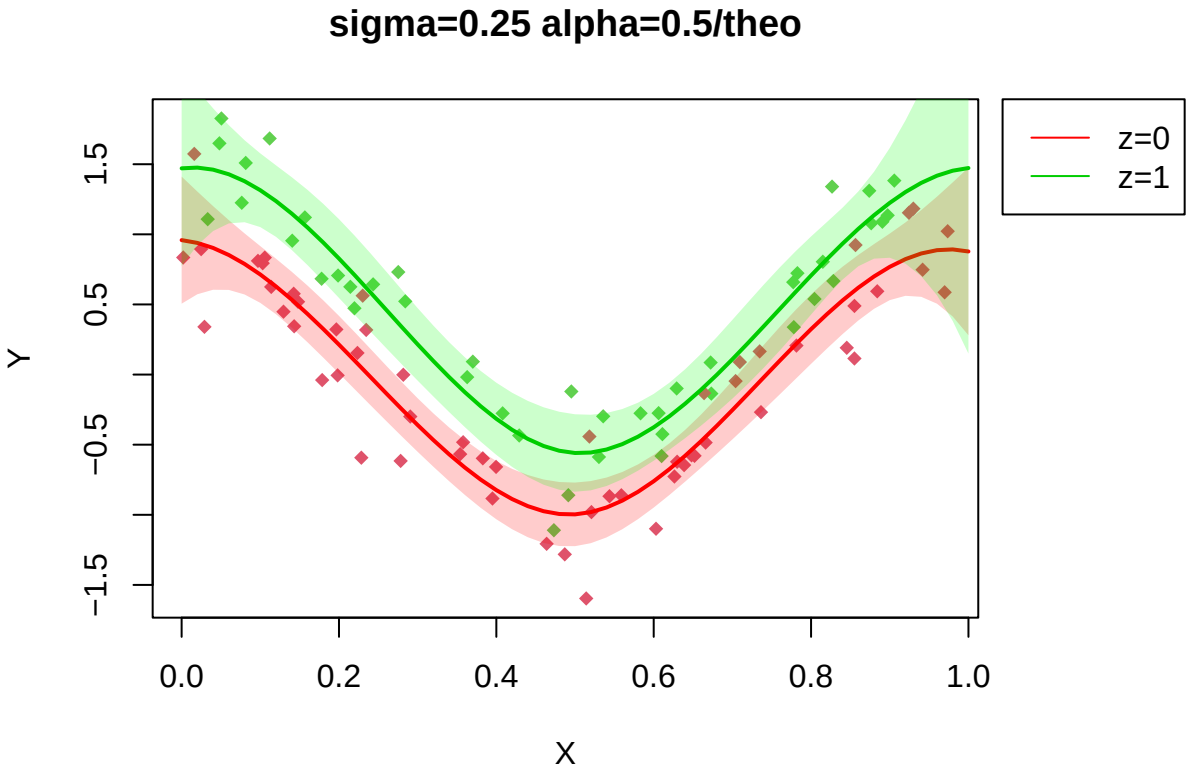
```
    Yi <- Y + rnorm(length(Y), sd=sd)
    sim[iter] <- mhat(x, Yi, B, W, p)
  }
  c(quantile(sim, 0.025), mean(sim), quantile(sim, 0.975))
}

# Theoretical approach
CI_theo <- function(x, Y, B, W, p, sd=NULL){
  n <- length(Y)
  Bx <- bs_gen(x, p)
  hatY <- sapply(1:n, function(i) hatYi(i, Y, B, W))
  sig2 <- mean((Y - hatY) ^ 2)
  inv <- ginv(t(B) %*% W %*% B)
  A <- Bx %*% inv %*% t(B) %*% W
  var <- sum(A^2) * sig2
  mhat <- A %*% Y
  err <- qnorm(0.975) * sqrt(var)
  c(mhat - err, mhat, mhat + err)
}

# Plot CI
colors = c(rgb(1,0,0,0.2), rgb(0,1,0,0.2), rgb(0,0,1,0.2), rgb(0.8,0.8,0,0.2))
line_colors=c("red", "green3", "blue", "yellow3")

CI_plot <- function(xx, Y, B, zz, p, lambda, sd, alpha, type, eps=0){
  CIf <- ifelse(type=="emp", CI_emp, CI_theo)
  par(mar = c(5, 5, 4, 6))
  plot(X, Y+eps, col=Z+2, pch=18, ylab="Y",
       main=paste0("sigma=", sd, " alpha=", alpha, "/", type))
  for (z in zz){
    Wz <- W_gen(Z, z=z, lambda=lambda)
    zi <- sapply(xx, function(x) CIf(x, Y, B, Wz, p, sd))
    polygon(c(xx, rev(xx)), c(zi[1,], rev(zi[3,])), col=colors[z+1], border=F)
    lines(xx, zi[2,], col=line_colors[z+1], lwd=2)
  }
  legend(x="topright", legend=sapply(zz, function(z) paste0("z=",z)),
         col=line_colors, lty=1, inset=c(-0.25, 0), xpd=T)
}
```

(1) $\sigma = 0.25, \alpha = 0.5.$

```
xx <- seq(0, 1, length = 51)
zz <- c(0, 1)
sd <- 0.25
alpha <- 0.5
Y <- cos(2 * pi * X) + alpha * Z
lambda <- 0.03
CI_plot(xx, Y, B, zz, p, lambda, sd, alpha, type="emp", eps=eps1)
```

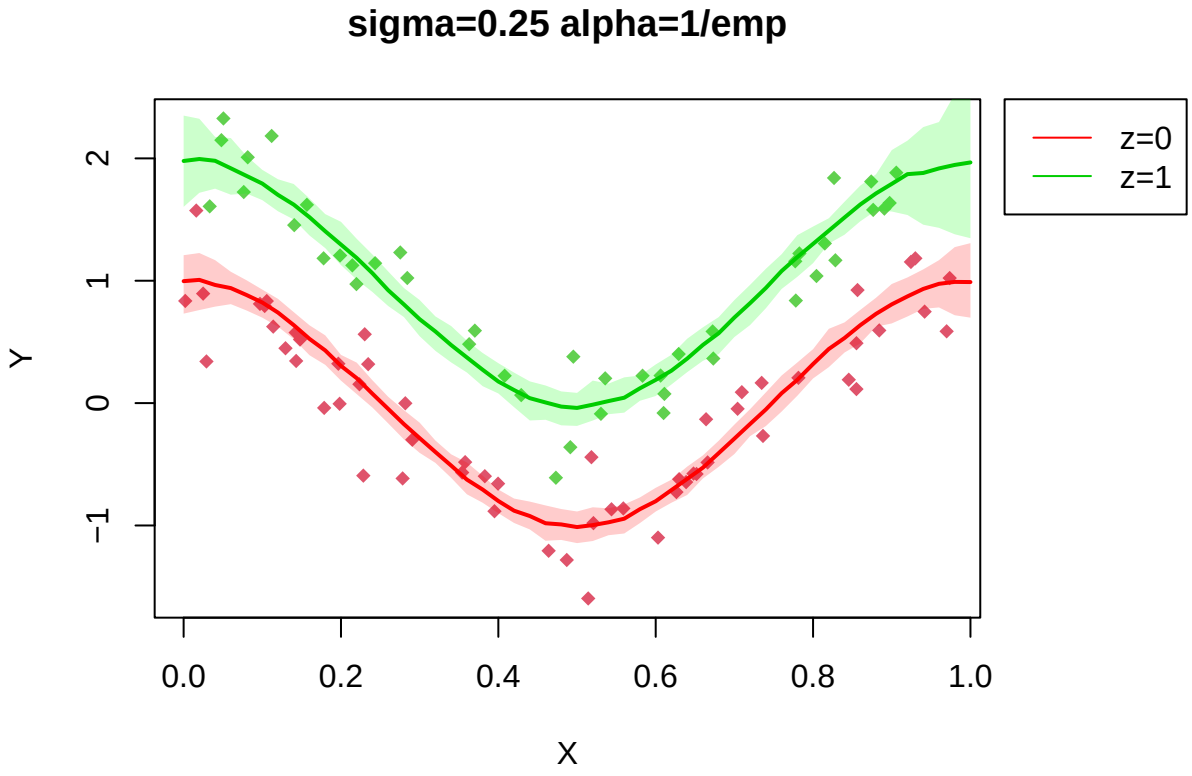



```
CI_plot(xx, Y+eps1, B, zz, p, lambda, sd, alpha, type="theo")
```

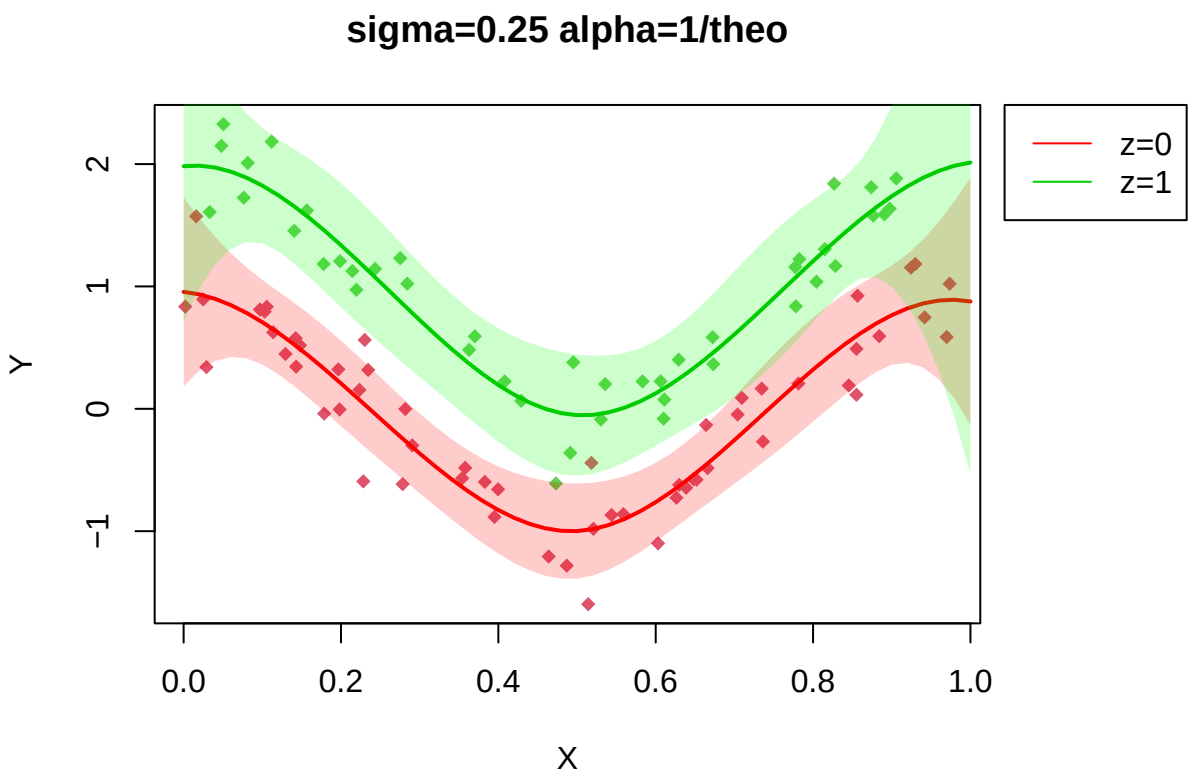


(2) $\sigma = 0.25, \alpha = 1.$

```
sd <- 0.25
alpha <- 1
Y <- cos(2 * pi * X) + alpha * Z
lambda <- 0.01
CI_plot(xx, Y, B, zz, p, lambda, sd, alpha, type="emp", eps=eps1)
```

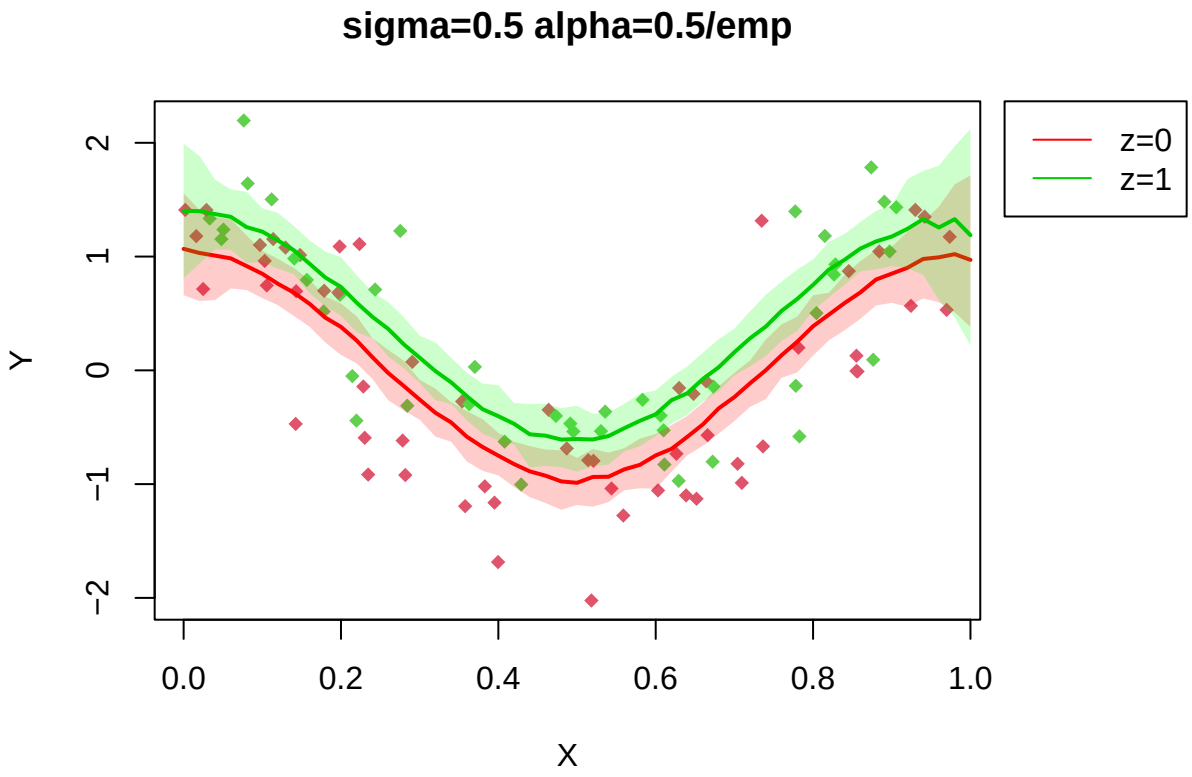



```
CI_plot(xx, Y+eps1, B, zz, p, lambda, sd, alpha, type="theo")
```

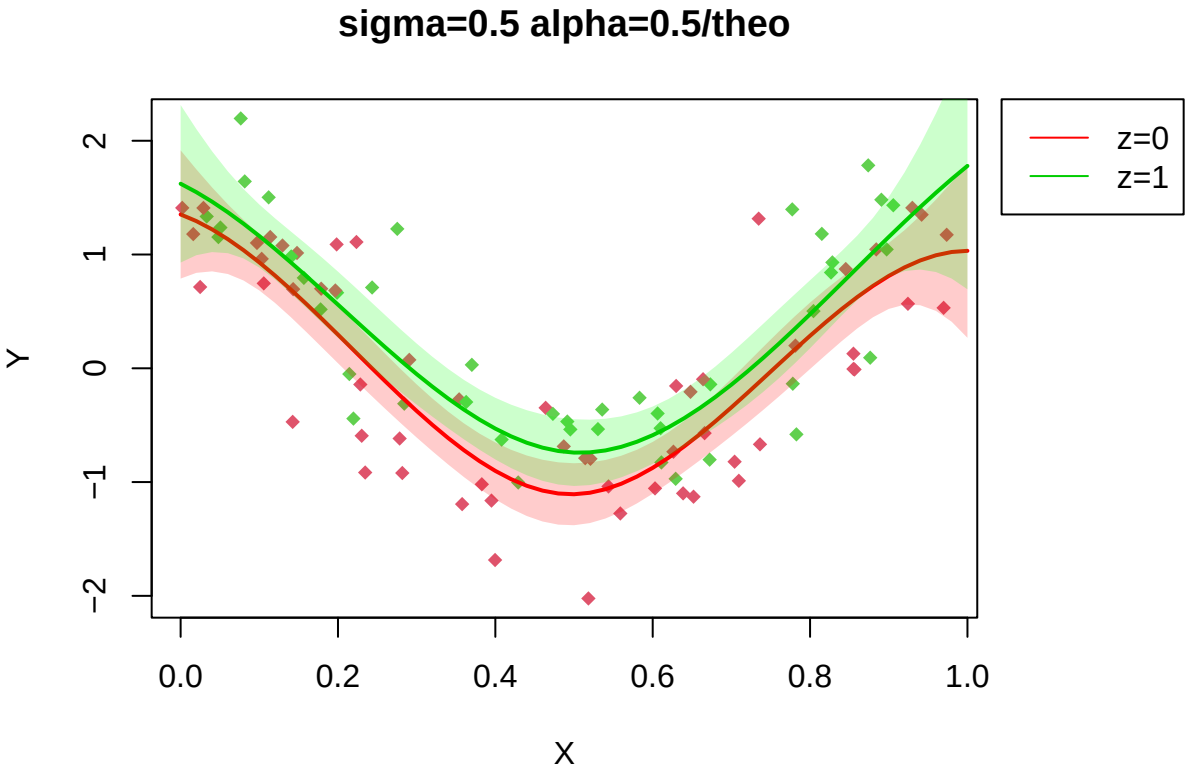


(3) $\sigma = 0.5, \alpha = 0.5$**.**

```
sd <- 0.5
alpha <- 0.5
Y <- cos(2 * pi * X) + alpha * Z
lambda <- 0.15
CI_plot(xx, Y, B, zz, p, lambda, sd, alpha, type="emp", eps=eps2)
```

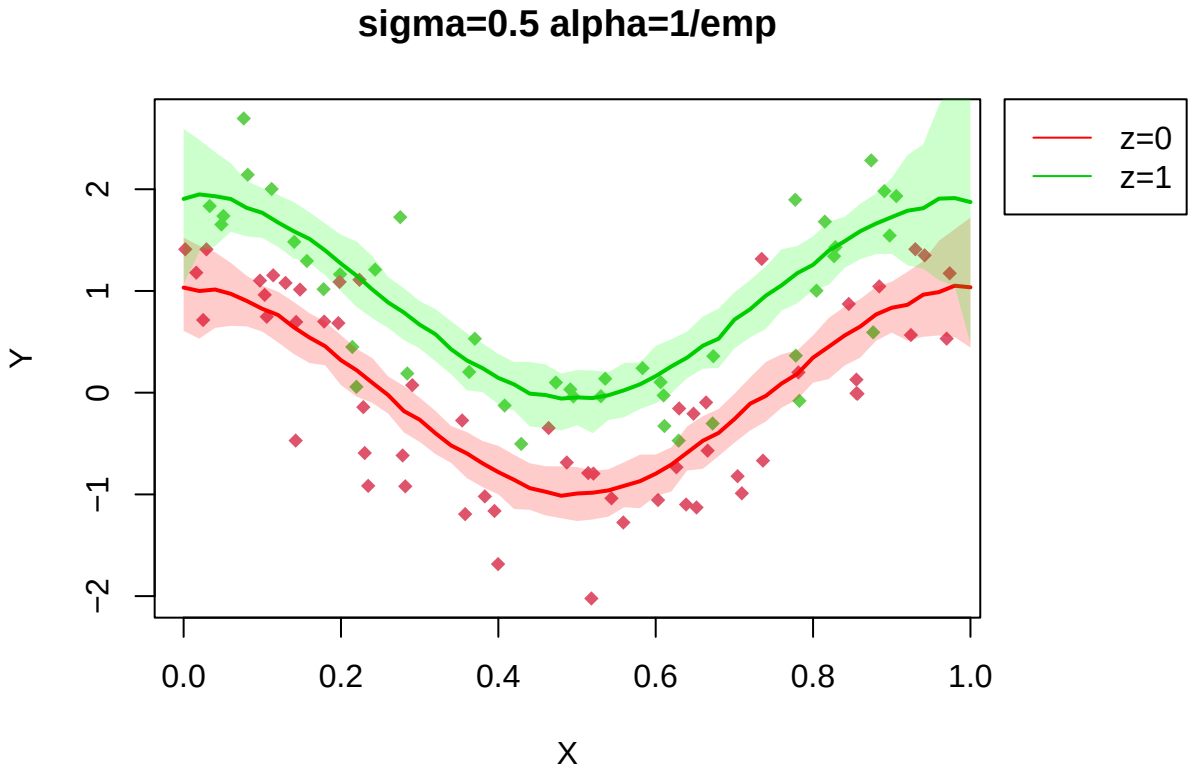



```
CI_plot(xx, Y+eps2, B, zz, p, lambda, sd, alpha, type="theo")
```

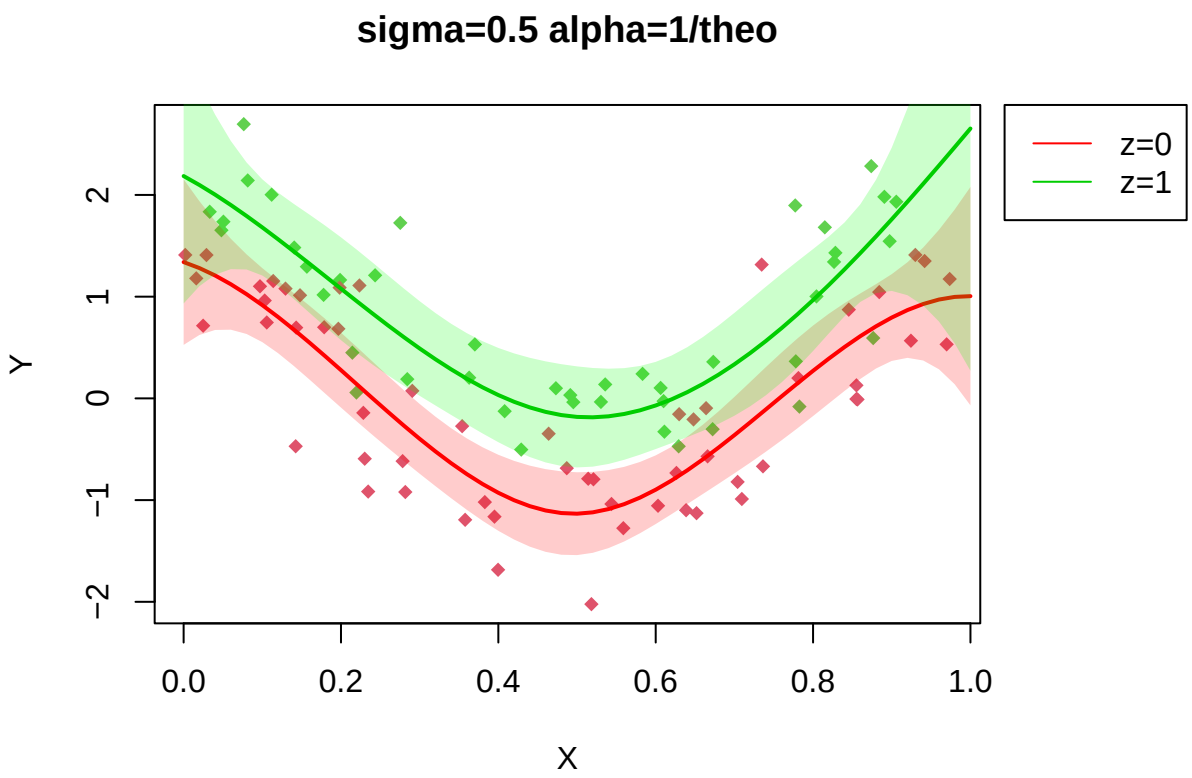


(4) $\sigma = 0.5, \alpha = 1.$

```
sd <- 0.5
alpha <- 1
Y <- cos(2 * pi * X) + alpha * Z
lambda <- 0.03
CI_plot(xx, Y, B, zz, p, lambda, sd, alpha, type="emp", eps=eps2)
```




```
CI_plot(xx, Y+eps2, B, zz, p, lambda, sd, alpha, type="theo")
```



(1) ~ (4)의 결과를 종합해 보면 다음과 같다.

λ의 값이 작을수록 이론적으로 구한 신뢰구간이 더 넓다. 이러한 이유는 $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$로 추정했기 때문이다.

실제 저 값이 무엇인지 생각해보면 다음과 같다.

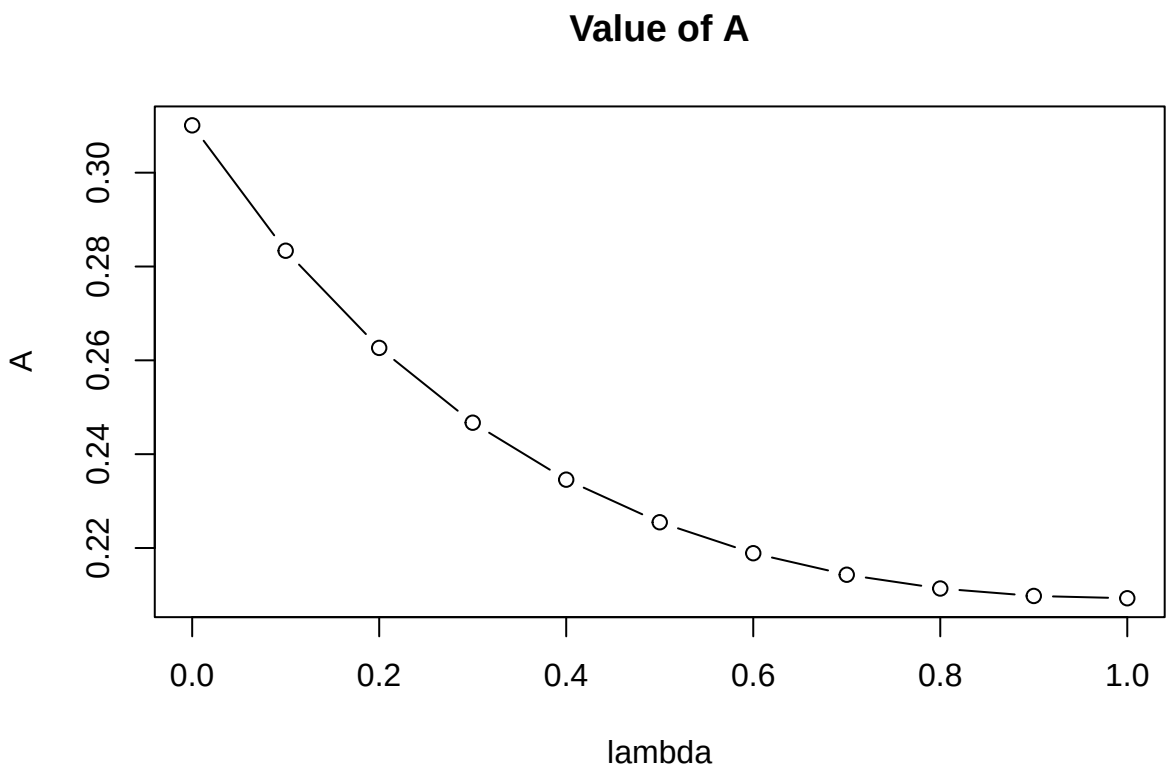
$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2 &\approx \mathbb{E}\left[(Y-\hat{Y})^2\right] \\ &= \mathbb{E}\left[(Y-\bar{Y}+\bar{Y}-\hat{Y})^2\right] \\ &= \mathbb{E}\left[(Y-\bar{Y})^2\right]+\mathbb{E}\left[(\hat{Y}-\bar{Y})^2\right] \\ &= Var(Y)+\mathbb{E}\left[(\hat{Y}-\mathbb{E}(\hat{Y}))^2\right] \\ &= Var(Y)+Var(\hat{Y})\end{aligned}$$

우리가 추정한 $\hat{\sigma}^2$ 이 실제의 분산에 $\hat{Y}$ 의 분산 만큼 더해진 biased estimator이기 때문에, 위와 같은 현상이 일어난다. 따라서 우리가 위에서 계산한 값은 다음과 같다.

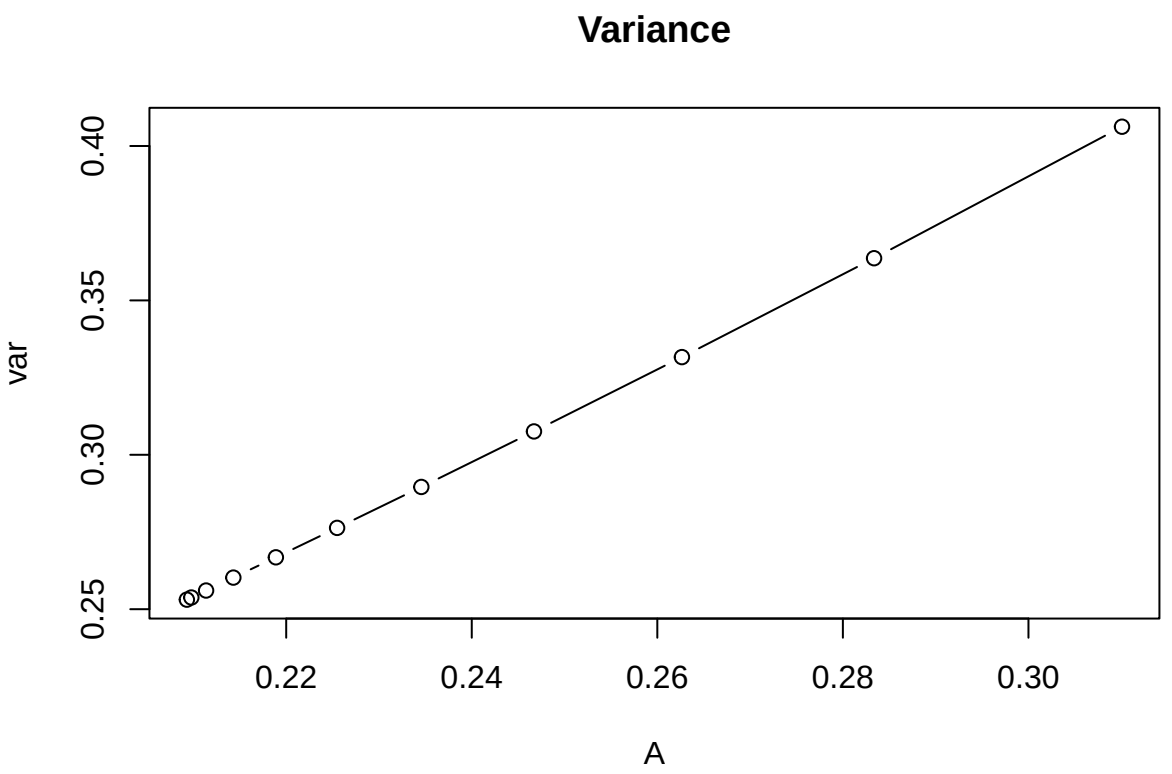
$$\begin{aligned}\hat{Var}(\hat{m}(x,z)) &\approx \tilde{B}(B^tWB)^{-1}B^tW^2B(B^tWB)^{-1}\tilde{B}^t(\sigma^2+Var(\hat{Y})) \\ &= A(\sigma^2+A\sigma^2), \quad where \quad A=\tilde{B}(B^tWB)^{-1}B^tW^2B(B^tWB)^{-1}\tilde{B}^t \\ &= A(1+A)\sigma^2\end{aligned}$$

여기서 λ가 작아질수록 A의 값은 커지고 $Var(\hat{Y})$ 또한 커진다. 실제로 그래프를 그려보면 아래와 같다.

```
Bx <- bs_gen(0, p)
B <- bs_gen(X, p)
A <- function(lambda){
  W <- W_gen(Z, 0, lambda)
  inv <- ginv(t(B) %*% W %*% B)
  H <- Bx %*% inv %*% t(B) %*% W
  c(H %*% t(H))
}
xx <- seq(0,1,by=0.1)
yy <- sapply(xx, A)
plot(xx, yy, type="b", xlab="lambda", ylab="A", main="Value of A")
```



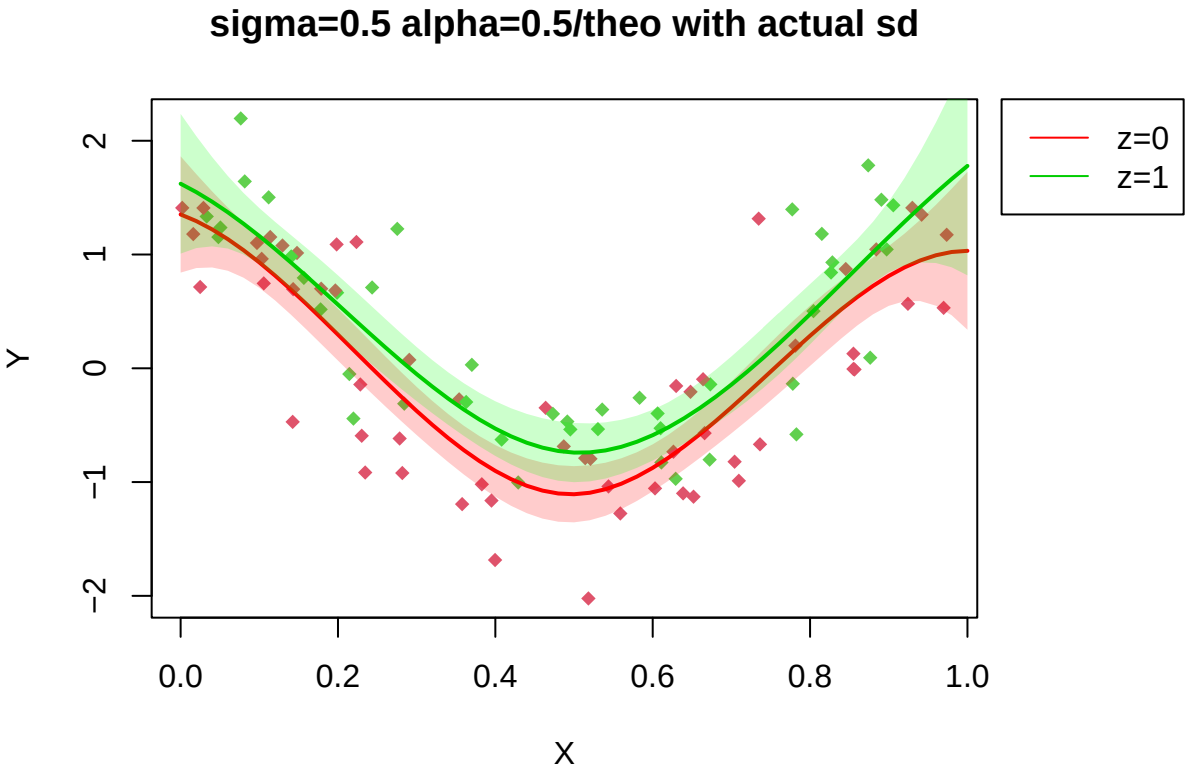

```
plot(yy, yy*(1+yy), type="b", xlab="A", ylab="var", main="Variance")
```



따라서 λ 값이 작아질수록 이론적으로 구한 신뢰구간의 너비가 실험적으로 구한 신뢰구간의 너비보다 점점 커지게 된다는 것을 알 수 있다.

실제 σ^2 의 값을 추정값 대신 넣어주게 되면, 신뢰구간의 너비가 거의 같음을 알 수 있다.

```
CI_theo2 <- function(x, Y, B, W, p, sd=NULL){
  Bx <- bs_gen(x, p)
  inv <- ginv(t(B) %*% W %*% B)
  A <- Bx %*% inv %*% t(B) %*% W
  var <- sum(A^2)
  mhat <- A %*% Y
  err <- qnorm(0.975) * sqrt(var) * sd
  c(mhat - err, mhat, mhat + err)
}

sd <- 0.5
alpha <- 0.5
Y <- cos(2 * pi * X) + alpha * Z + eps2
lambda <- 0.15
xx <- seq(0, 1, length = 51)
zz <- c(0, 1)

par(mar = c(5, 5, 4, 6))
plot(X, Y, col=Z+2, pch=18, ylab="Y",
     main=paste0("sigma=", sd, " alpha=", alpha, "/", "theo with actual sd"))
for (z in zz){
  Wz <- W_gen(Z, z=z, lambda=lambda)
  zi <- sapply(xx, function(x) CI_theo2(x, Y, B, Wz, p, sd))
  polygon(c(xx, rev(xx)), c(zi[1,], rev(zi[3,])), col=colors[z+1], border=F)
  lines(xx, zi[2,], col=line_colors[z+1], lwd=2)
}
legend(x="topright", legend=sapply(zz, function(z) paste0("z=",z)),
       col=line_colors, lty=1, inset=c(-0.25, 0), xpd=T)
```



Q6. PM2.5/PM10

서울의 풍향을 동, 서, 남, 북으로 categorical하게 나눠 각각의 경우에 대해 베이징의 미세먼지 수준을 predictor로 서울의 미세먼지 수준과 관련이 있는지 확인해 보고자 한다. 풍향 데이터는 기상청 기상자료개방포털에서 발췌하였고, 서울과 베이징의 미세먼지 관련 데이터는 'aqicn.org' 사이트에서 발췌하였다. 최근 1년간의 일별 미세먼지 평균 자료 및 최대풍속풍향 정보를 사용하였으며, 앞서 만든 함수들을 사용하기 위해 미세먼지 데이터를 [0, 1] 구간으로 scaling 하였다.

```
wind <- read.csv("wind.csv")
seoul <- read.csv("seoul-air-quality.csv")
beijing <- read.csv("beijing-air-quality.csv")
wind$date <- as.Date(wind$date)
seoul$date <- as.Date(seoul$date)
beijing$date <- as.Date(beijing$date)
wind <- subset(wind, date>as.Date("2021-06-17"))
wind$date <- as.Date(wind$date)
seoul <- subset(seoul, pm10 < 200, select=c(date, pm25, pm10))
beijing <- subset(beijing, pm25 < 300 & pm10 < 200, select=c(date, pm25, pm10))
df <- merge(beijing, seoul, by="date")
df <- merge(wind, df, by="date")
df$direction <- ((df$direction-45) %% 360) %/% 90
```

```r
df <- na.omit(df)
head(df) # x for Seoul, y for Beijing
```

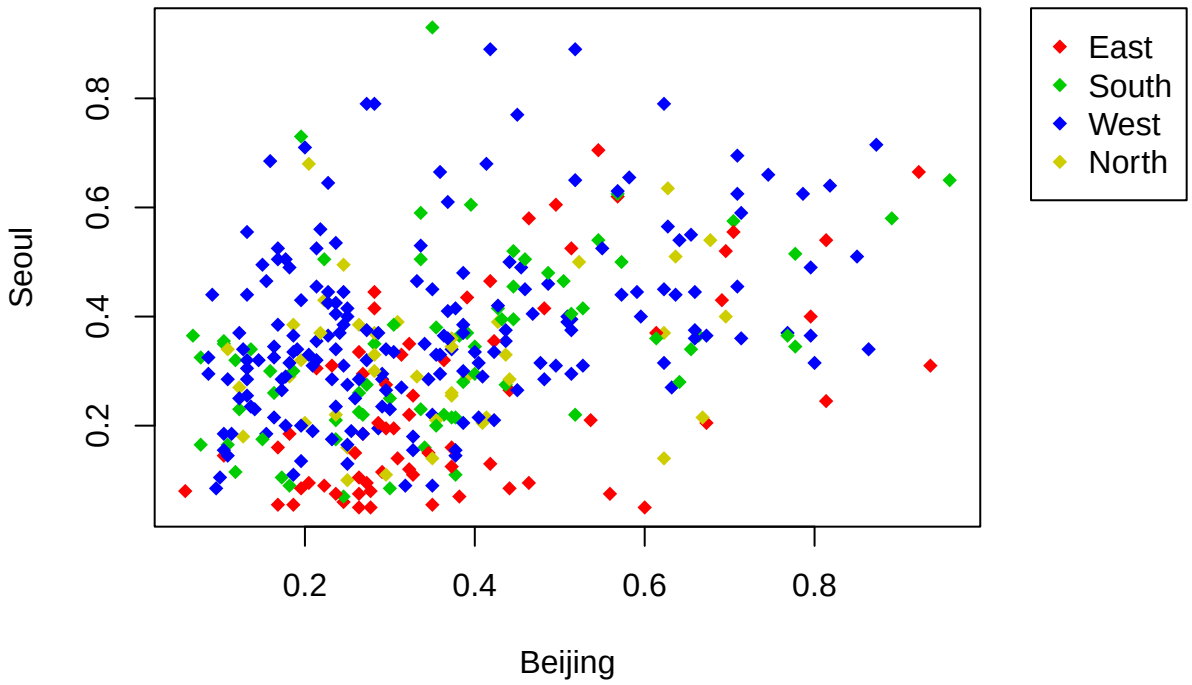
date	direction	pm25.x	pm10.x	pm25.y	pm10.y
2021-06-18	1	27	35	46	47
2021-06-19	2	39	30	101	49
2021-06-20	2	47	65	105	44
2021-06-21	2	73	38	93	22
2021-06-22	1	58	34	52	19
2021-06-23	0	71	47	44	33

```r
# scaling
coef <- c(220, 160, 200, 120)
for (i in 1:4){
  df[i+2] <- df[i+2] / coef[i]
}

head(df)
```

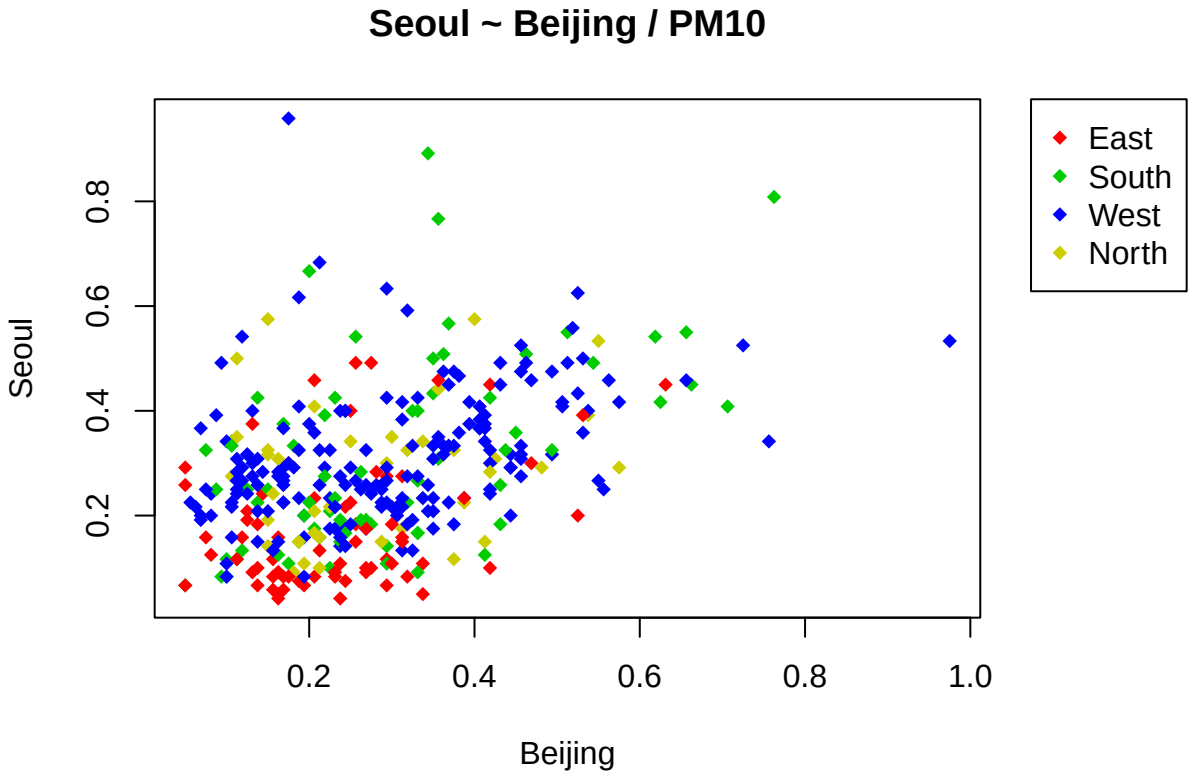
date	direction	pm25.x	pm10.x	pm25.y	pm10.y
2021-06-18	1	0.1227273	0.21875	0.230	0.3916667
2021-06-19	2	0.1772727	0.18750	0.505	0.4083333
2021-06-20	2	0.2136364	0.40625	0.525	0.3666667
2021-06-21	2	0.3318182	0.23750	0.465	0.1833333
2021-06-22	1	0.2636364	0.21250	0.260	0.1583333
2021-06-23	0	0.3227273	0.29375	0.220	0.2750000

우선 방향별 산점도를 그려보면 다음과 같다.

```r
par(mar=c(5, 5, 4, 6))
plot(df$pm25.x, df$pm25.y, col=line_colors[df$direction+1], pch=18,
     main="Seoul ~ Beijing / PM2.5", xlab="Beijing", ylab="Seoul")
legend(x="topright", legend=c("East","South","West","North"),
       col=line_colors, pch=18, inset=c(-0.25, 0), xpd=T)
```

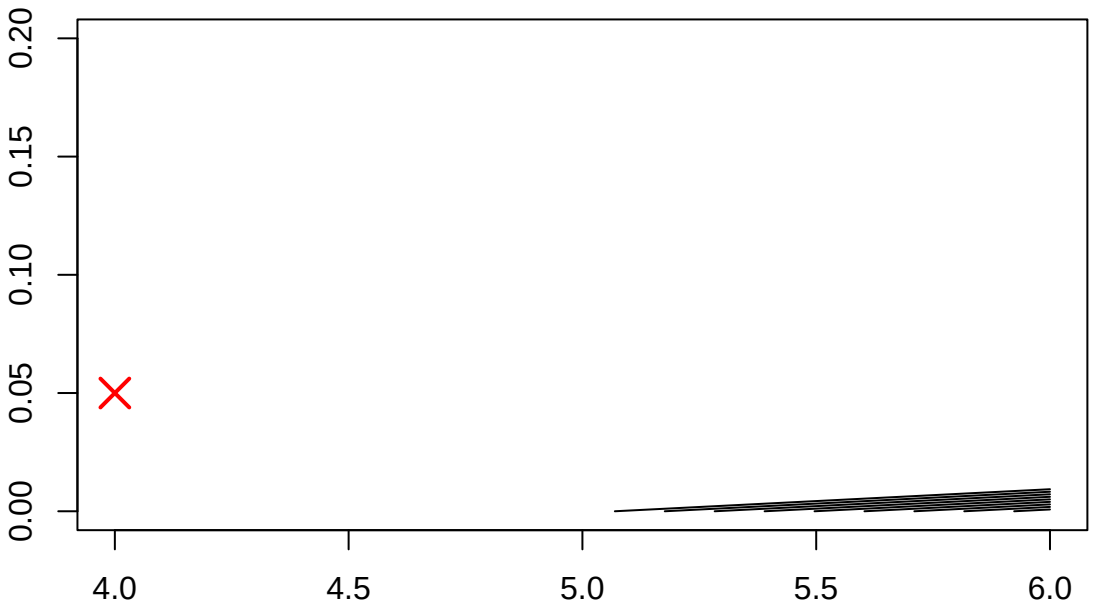



```
plot(df$pm10.x, df$pm10.y, col=line_colors[df$direction+1], pch=18,
     main="Seoul ~ Beijing / PM10", xlab="Beijing", ylab="Seoul")
legend(x="topright", legend=c("East","South","West","North"),
       col=line_colors, pch=18, inset=c(-0.25, 0), xpd=T)
```

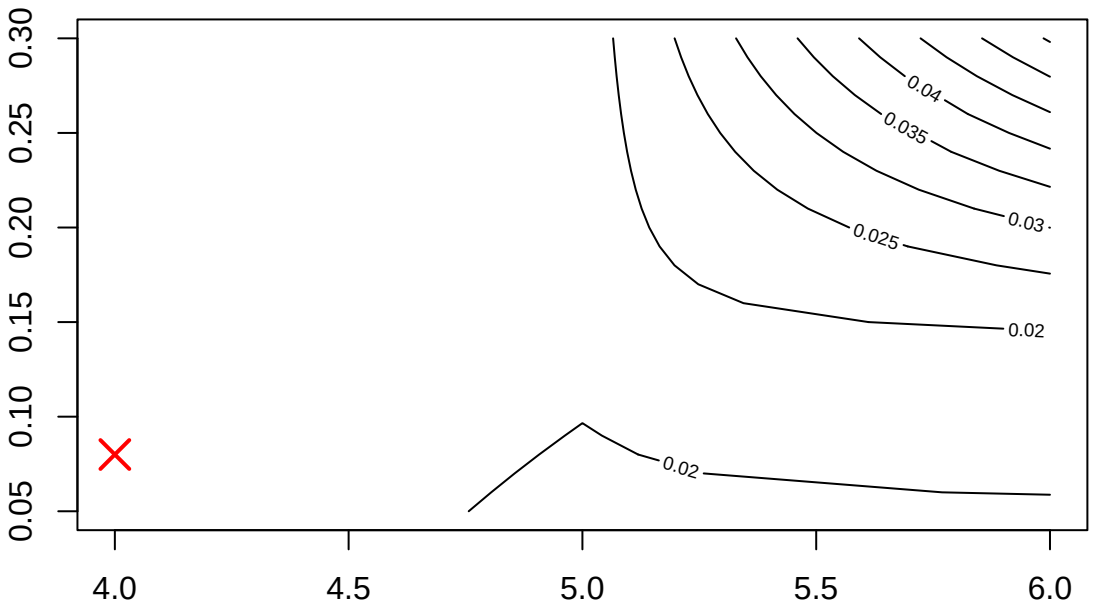


먼저 적절한 p와 λ를 찾기 위해, CV를 실시한다. 참고로, 방위 데이터는 ordinal의 성격보다 nominal의 성격이 더 강하므로 W의 type으로 nominal을 사용하였다.

```
# hyperparameter setting
ps <- c(4, 5, 6)
lams <- seq(0, 0.2, by=0.01)

# Calculate CV
cvs25 <- CV1(df$pm25.y, df$pm25.x, df$direction, ps, lams)
mincv25 <- min(cvs25)
ind <- which(cvs25==mincv25, arr.ind=TRUE)
contour(ps,lams, cvs25, main="CV for PM2.5")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")
```

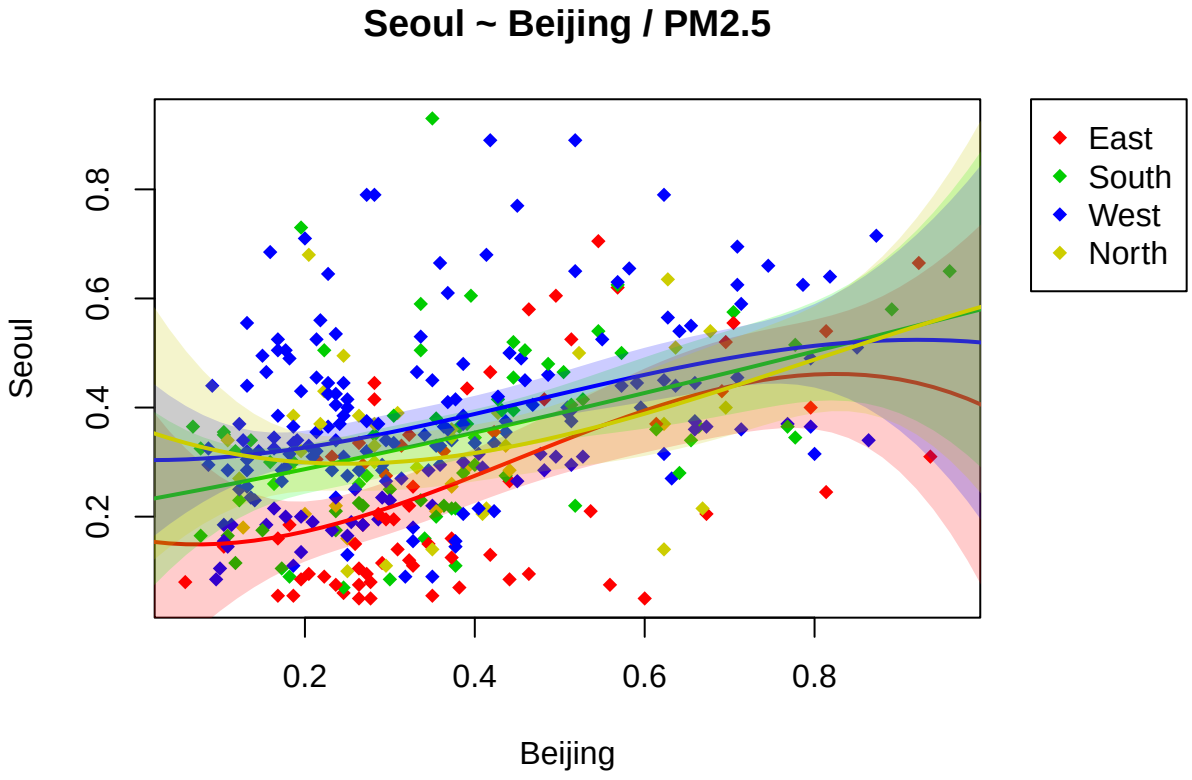



```
# hyperparameter setting
ps <- c(4, 5, 6)
lams <- seq(0.05, 0.3, by=0.01)

# Calculate CV
cvs10 <- CV1(df$pm10.y, df$pm10.x, df$direction, ps, lams)
mincv10 <- min(cvs10)
ind <- which(cvs10==mincv10, arr.ind=TRUE)
contour(ps,lams, cvs10, main="CV for PM10")
points(ps[ind[1]], lams[ind[2]], pch=4, cex=2, lwd=2, col="red")
```

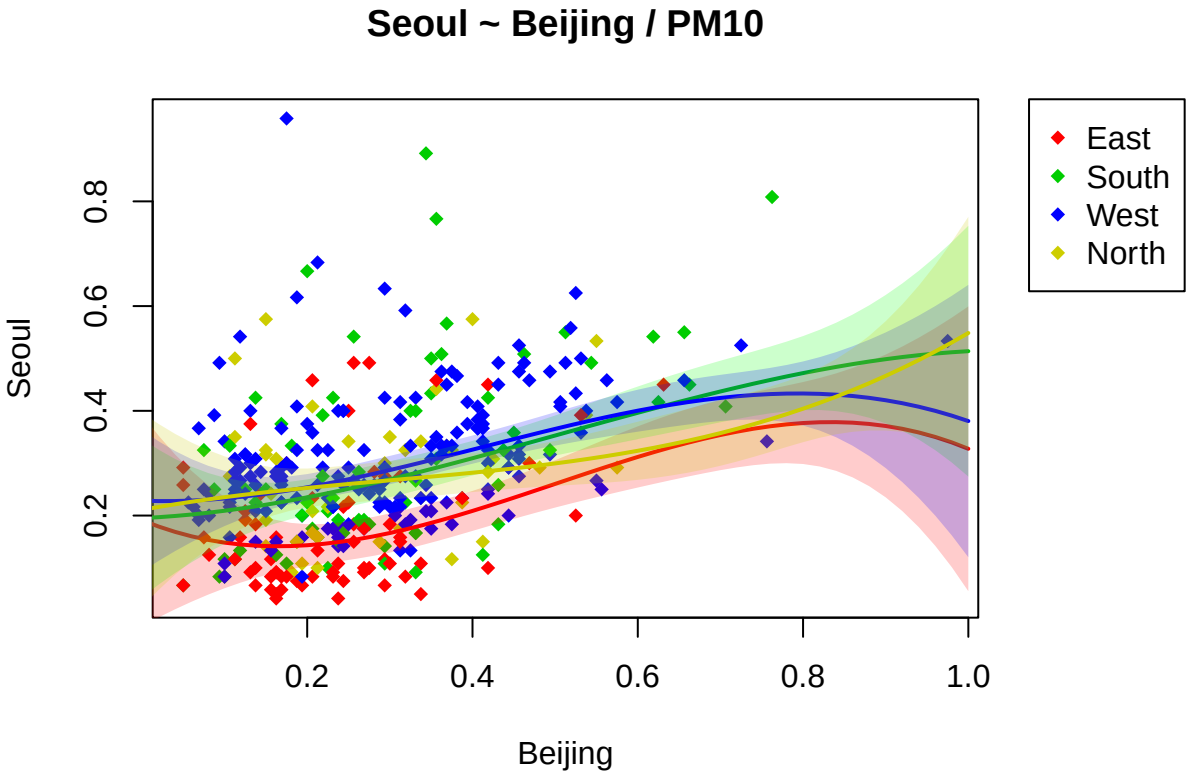


PM2.5 데이터에 대해서는 $p = 4, \lambda = 0.04$의 값이 최적이라는 결론을 내릴 수 있고, PM10 데이터에 대해서는 $p = 4$, $\lambda = 0.075$이 최적이라는 결론을 내릴 수 있다. 이를 대입하여 함수를 추정해 보았다.

```
CI_plot <- function(xx, X, Y, zz, Z, p, lambda, pm){
  B <- bs_gen(df$pm25.x, p)
  par(mar = c(5, 5, 4, 6))
  plot(X, Y, col=line_colors[Z+1], pch=18, ylab="Seoul", xlab="Beijing",
       main=paste0("Seoul ~ Beijing / PM", pm))
  for (z in zz){
    Wz <- W_gen(Z, z=z, lambda=lambda)
    zi <- sapply(xx, function(x) CI_theo(x, Y, B, Wz, p))
    polygon(c(xx, rev(xx)), c(zi[1,], rev(zi[3,])), col=colors[z+1], border=F)
    lines(xx, zi[2,], col=line_colors[z+1], lwd=2)
  }
  legend(x="topright", legend=c("East", "South", "West", "North"),
         col=line_colors, pch=18, inset=c(-0.25, 0), xpd=T)
}

p <- 4
lambda <- 0.04
zz <- 0:3
CI_plot(xx, df$pm25.x, df$pm25.y, zz, df$direction, p, lambda, pm=2.5)
```




```
p <- 4
lambda <- 0.075
zz <- 0:3
CI_plot(xx, df$pm10.x, df$pm10.y, zz, df$direction, p, lambda, pm=10)
```



먼저 PM2.5 데이터부터 살펴보자. 베이징의 미세먼지 농도가 0.2 ~ 0.5 정도 (원래 스케일에서는 44 ~ 110) 에서, 서풍과 북/남풍, 동풍이 불 때 차이가 구별됨을 알 수 있다. 베이징의 미세먼지 농도를 고정시켜두고, 서울의 미세먼지 농도를 풍향별로 생각해보자.

- 서풍 : 서울 미세먼지 농도가 가장 높다. 베이징은 서울의 서쪽에 있음에도 불구하고, 서풍이 불 때 서울의 미세먼지 농도가 높은 이유는 다음과 같이 추측할 수 있다. 베이징 미세먼지가 확산될 때, 바람이 부는 방향으로 퍼질 것이다. 베이징에서 동쪽으로 바람이 불고, 서울에서 서쪽으로 바람이 분다면 미세먼지는 가운데에서 정체될 것이다. 이러한 경우에는 서울의 미세먼지 농도가 높게 나올 것으로 예상된다. 반면에 베이징에서도 서풍이 분다면 서울의 미세먼지가 서쪽으로 사라지는 경향이 관찰될 수 있다. 실제로 파란 점들의 분포를 관찰하면 다른 점들에 비해 분산이 크게 나타남을 알 수 있다. 평균이 높은 이유는 베이징에서 동풍이 불 때 증가하는 미세먼지 농도가 워낙 크기 때문이라고 생각된다.
- 동풍 : 서울 미세먼지 농도가 가장 낮다. 서풍이 불 때와 반대되는 효과로 볼 수 있다. 만약 베이징에서 동풍이 불어 미세먼지가 서울로 넘어온다 해도 서울에서의 동풍에 의해 강원도 쪽으로 이동될 것이고, 베이징에서 서풍이 분다면 서울의 미세먼지는 더욱 줄어들 것이다.
- 북/남풍 : 북풍과 남풍의 효과는 거의 비슷하였다. 서풍과 동풍의 효과를 절반만 받는다고 생각할 수 있으므로, 그러한 평균 미세먼지 농도 또한 둘의 평균이라 예측할 수 있다.

다음으로, PM10 데이터를 살펴보자. PM2.5와 마찬가지로, 베이징의 미세먼지 농도가 0.2 ~ 0.5 정도 (원래 스케일에서는 32 ~ 80) 에서, 동풍과 북/남/서풍의 차이가 구별된다. 추정된 함수만 보면 북/남/서풍도 미세하게 구분되지만, 신뢰구간 안에서 구분이 되지 않으므로 효과의 차이가 난다고 하기 어렵다.

- 동풍 : 서울 미세먼지 농도가 가장 낮다. 이러한 이유는 PM2.5 데이터와 마찬가지로, 서울의 미세먼지를 다른 곳으로 분산시켜주는 역할을 하기 때문에 다른 풍향에 비해 낮은 농도가 나왔다고 생각할 수 있다.

- 북/남/서풍 : 서울 미세먼지 농도가 비교적 높다. 이러한 이유는, PM10 미세먼지의 입자가 PM2.5 미세먼지의 입자보다 크기가 커서 생기는 현상이라고 생각할 수 있다. 그렇게 되면 미세먼지가 이동하기 위해서는 더 강한 바람이 불어야하고, 북/남풍으로 상쇄되는 절반의 효과로는 미세먼지를 이동시킬만한 강한 힘이 제공되지 않아 서풍과 비슷한 수준의 미세먼지 농도를 가질 수 있다고 생각할 수 있다.

종합하면, 동풍이 불 때 서울의 전반적인 미세먼지 농도는 낮으며, 북/남풍이 불면 서울의 PM2.5 미세먼지 농도가 소폭 감소하고, 서풍이 불 때 서울의 미세먼지 농도가 가장 높다고 결론지을 수 있다.

위 분석의 한계를 꼽자면, 일별 최대풍속풍향이 미세먼지의 흐름을 지배하지 않을 수 있다. 또한 서울의 풍향만 가지고 분석한 점이 아쉽고, 중국의 풍향까지 고려하여 케이스를 세세하게 나눈다면 변화가 있을 수 있다고 생각한다. 해당 데이터를 보면, 분산이 매우 크므로 함수의 추정이 어려웠던 점도 꼽을 수 있다.