함수추정의 응용 및 실습

Assignment #3

서울대학교 통계학과 2017-11362 박건도

2022년 05월 30일

1. CV for Ordinary Kriging.

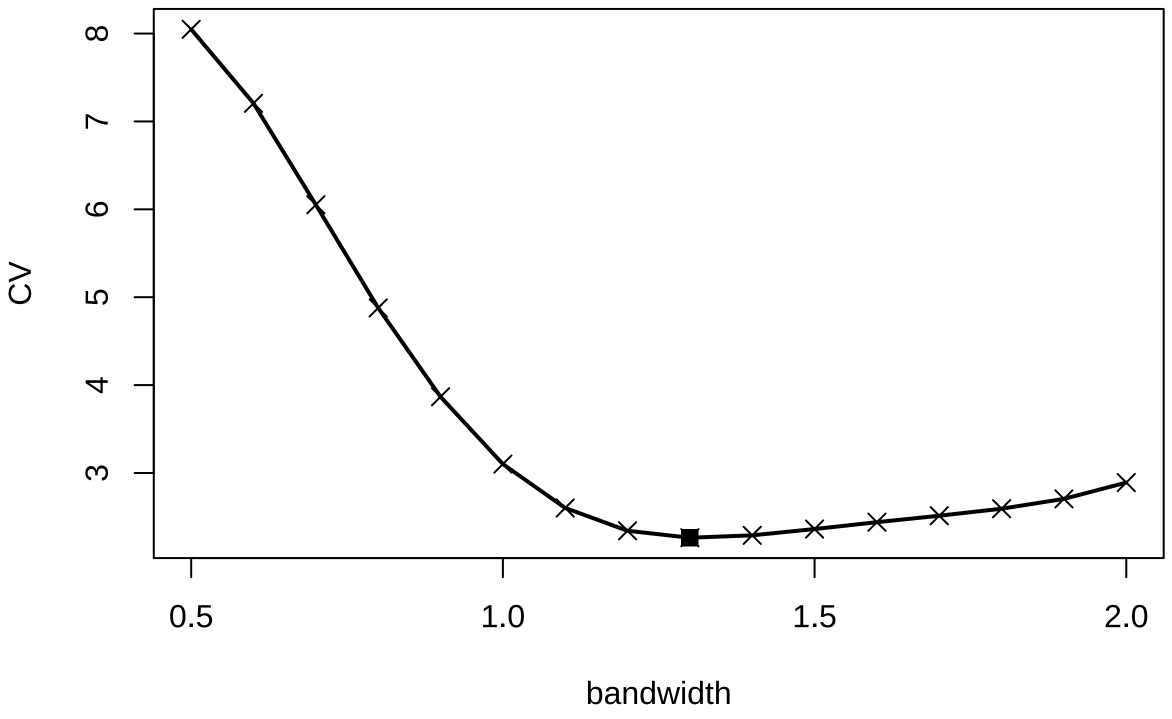
(a) CV

```r
# use (E) - mykrig1 for ordinary kriging
CV_delta <- function(nd, xx, yy, bws){
  calc_cv <- function(nd, xx, yy, bw){
    calc_res <- function(i, xx, yy, bw){
      xx1 <- xx[-i]
      yy1 <- yy[-i]
      ex <- xx[i]
      ey <- mykrig1(nd-1, 1, xx1, yy1, ex, bw)
      (ey - yy[i])^2
    }
    # delete data one by one except two ends
    res <- sapply(2:(nd-1), calc_res, xx=xx, yy=yy, bw=bw)
    mean(res)
  }
  cv <- sapply(bws, calc_cv, nd=nd, xx=xx, yy=yy)
  cv
}
```

(b) optimize δ.

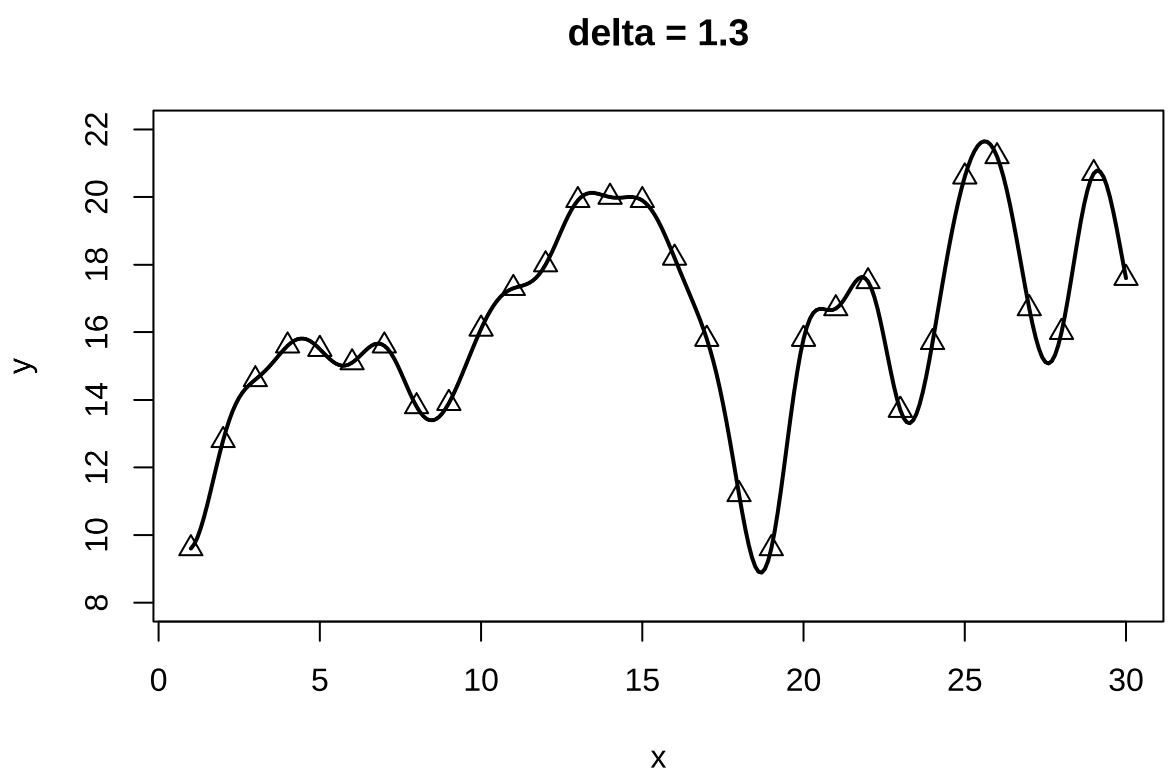
```r
X <- 1:30
Y <- c(9.6, 12.8, 14.6, 15.6, 15.5, 15.1, 15.6, 13.8, 13.9, 16.1, 17.3, 18,
       19.9, 20, 19.9, 18.2, 15.8, 11.2, 9.6, 15.8, 16.7, 17.5, 13.7, 15.7,
       20.6, 21.2, 16.7, 16, 20.7, 17.6)
```

```r
bandw <- seq(0.5, 2, 0.1)
cv <- CV_delta(30, X, Y, bandw)

# CV plot
plot(bandw, cv, type = "n", xlab = "bandwidth", ylab = "CV")
points(bandw, cv, cex = 1.2, pch = 4)
lines(bandw, cv, lwd = 2)
cvmin <- min(cv)
icvmin <- (1.:length(bandw))[cv == cvmin]
bandcv <- bandw[icvmin]
points(bandcv, cvmin, cex = 1.2, pch = 15)
```



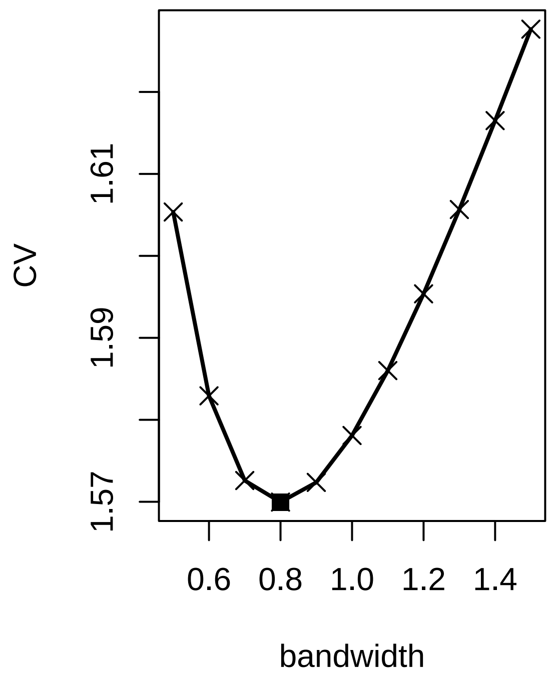
CV is minimized where $\delta = 1.3$.

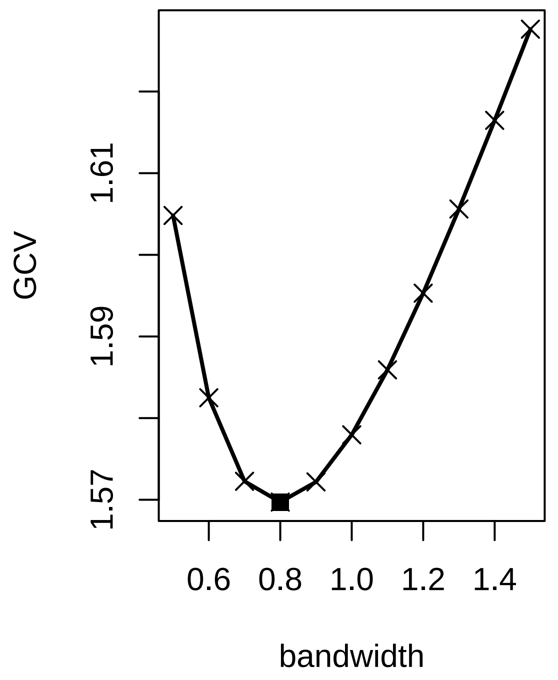
```r
ex <- seq(1, 30, by=0.1)
ne <- length(ex)
bw <- 1.3
ey <- mykrig1(30, ne, X, Y, ex, bw)
plot(X, Y, type="n", xlab="x", ylab="y", ylim=c(8,22), main="delta = 1.3")
points(X, Y, cex=1.2, pch=2)
lines(ex, ey, lwd=2)
```



2. CV-GCV for Simple Kriging.

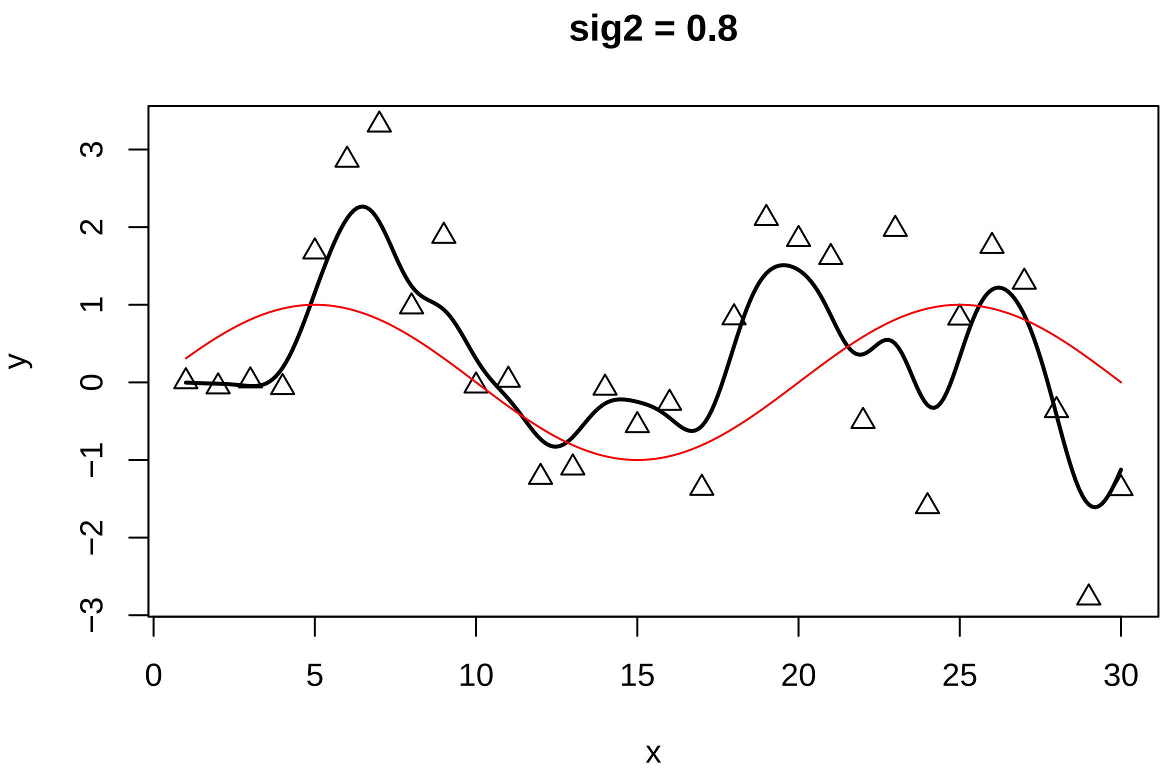
```r
# use (F) - mykrigs for simple kriging
krig_cvgcv <- function(nd, xx, yy, bw, sig2s){
  calc_cvgcv <- function (nd, xx, yy, bw, sig2){
    calc_res <- function(i, xx, yy, bw, sig2){
      xx1 <- xx[-i]
      yy1 <- yy[-i]

    }
    hat <- apply(diag(nd), 1, mykrigs, nd=nd, ne=nd,
                 xx=xx, ex=xx, bw=bw, sig2=sig2)
    ey <- hat %*% yy
    # except two ends
    res <- (ey - yy)[-c(1, nd)]
    dhat <- diag(hat)[-c(1, nd)]
    cv <- mean((res / (1-dhat))^2)
    gcv <- mean(res^2) / (1 - mean(dhat))^2
    list(cv=cv, gcv=gcv)
  }
  krig_out <- lapply(as.list(sig2s), calc_cvgcv, nd=nd, xx=xx, yy=yy, bw=bw)
  cvgcv <- unlist(krig_out)
  cv <- cvgcv[attr(cvgcv, "names") == "cv"]
  gcv <- cvgcv[attr(cvgcv, "names") == "gcv"]
  list(cv=cv, gcv=gcv)
}

# simulated with sig2 = 1
xx <- 1:30
Sigma <- sapply(1:30, function(x) exp(-((x-1:30)/1.3)^2))
# eps = eps_c + eps_u
eps <-mvrnorm(1, mu=rep(0, 30), Sigma=Sigma) + rnorm(30)
yy <- sin(0.1 * pi * xx) + eps

bandw <- seq(0.5, 1.5, by=0.1)
cvgcv <- krig_cvgcv(30, xx, yy, 1.3, bandw)
cv <- cvgcv$cv
gcv <- cvgcv$gcv

par(mfrow=c(1,2))
plot(bandw, cv, type = "n", xlab = "bandwidth", ylab = "CV")
points(bandw, cv, cex = 1.2, pch = 4)
lines(bandw, cv, lwd = 2)
cvmin <- min(cv)
icvmin <- (1.:length(bandw))[cv == cvmin]
bandcv <- bandw[icvmin]
points(bandcv, cvmin, cex = 1.2, pch = 15)
plot(bandw, gcv, type = "n", xlab = "bandwidth", ylab = "GCV")
points(bandw, gcv, cex = 1.2, pch = 4)
lines(bandw, gcv, lwd = 2)
gcvmin <- min(gcv)
igcvmin <- (1.:length(bandw))[gcv == gcvmin]
bandgcv <- bandw[igcvmin]
points(bandgcv, gcvmin, cex = 1.2, pch = 15)
```





CV와 GCV를 통해 구한 optimal $\widehat{\sigma^2} = 0.8$로, 실제 simulation에 적용된 $\sigma^2 = 1$과 유사한 값이다.

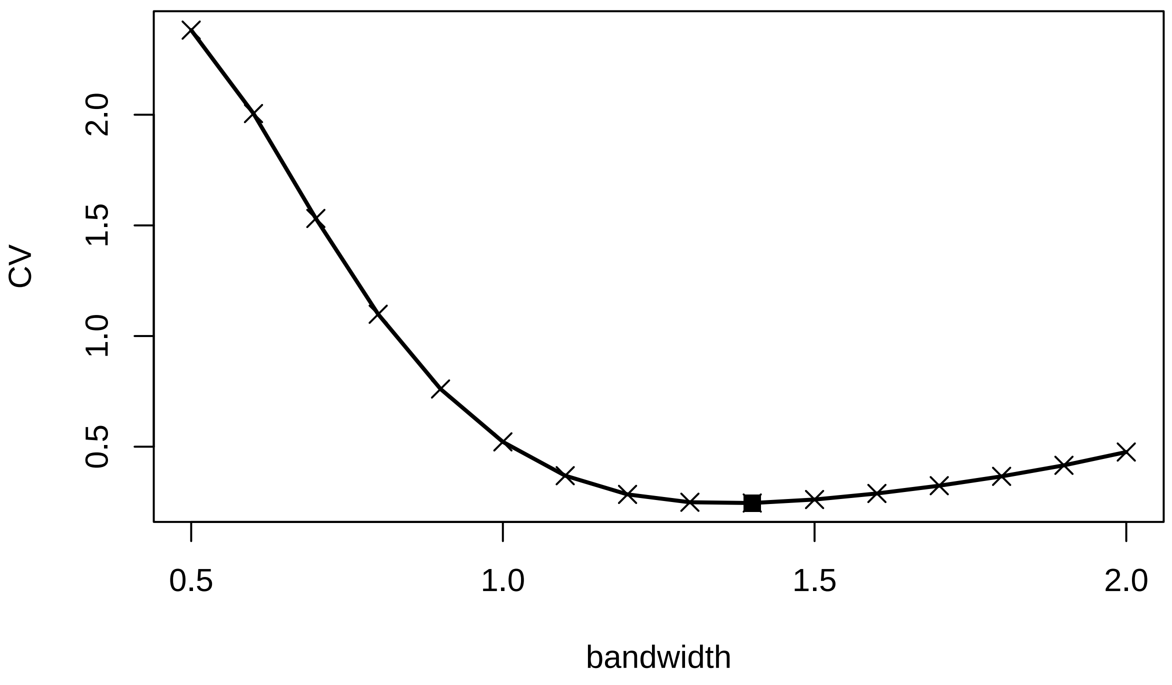
```r
ex <- seq(1, 30, by=0.1)
ne <- length(ex)
ey <- mykrigs(30, ne, xx, yy, ex, 1.3, 0.8)
plot(xx, yy, type="n", xlab="x", ylab="y", main="sig2 = 0.8")
points(xx, yy, cex=1.2, pch=2)
lines(ex, ey, lwd=2)
lines(ex, sin(0.1 * ex * pi), col="red")
```



3. Universal Kriging.

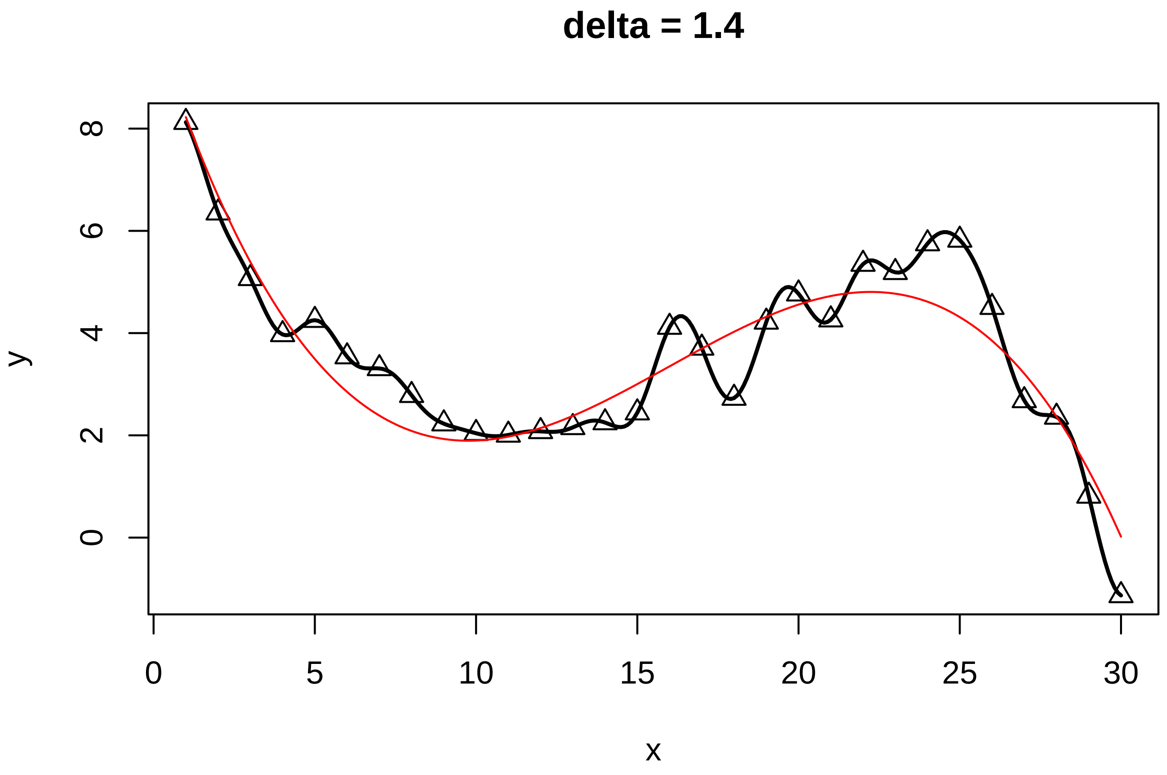
```
# Use (G) - mykrig2 for universal kriging
CV_delta2 <- function(nd, xx, yy, bws, np){
  calc_cv <- function(nd, xx, yy, bw, np){
    calc_res <- function(i, xx, yy, bw, np){
      xx1 <- xx[-i]
      yy1 <- yy[-i]
      ex <- xx[i]
      ey <- mykrig2(nd-1, 1, xx1, yy1, ex, bw, np)
      (ey - yy[i])^2
    }
    # delete data one by one except two ends
    res <- sapply(2:(nd-1), calc_res, xx=xx, yy=yy, bw=bw, np=np)
    mean(res)
  }
  cv <- sapply(bws, calc_cv, nd=nd, xx=xx, yy=yy, np=np)
  cv
}
```

```
xx <- 1:30
eps <- mvrnorm(n=1, mu=rep(0, 30), Sigma=Sigma)
yy <- 3 + 0.35 * (xx-15)  - 0.003 * (xx-16)^3 + eps

bandw <- seq(0.5, 2, 0.1)
cv <- CV_delta2(30, xx, yy, bandw, np=3)

# CV plot
plot(bandw, cv, type = "n", xlab = "bandwidth", ylab = "CV")
points(bandw, cv, cex = 1.2, pch = 4)
lines(bandw, cv, lwd = 2)
cvmin <- min(cv)
icvmin <- (1.:length(bandw))[cv == cvmin]
bandcv <- bandw[icvmin]
points(bandcv, cvmin, cex = 1.2, pch = 15)
```



CV를 통해, $\hat{\delta} = 1.4$의 값이 나왔고, 이를 대입하여 universal kriging을 하면 다음과 같다.

```
ex <- seq(1, 30, by=0.1)
ne <- length(ex)
ey <- mykrig2(30, ne, xx, yy, ex, 1.4, 3)
plot(xx, yy, type="n", xlab="x", ylab="y", main="delta = 1.4")
points(xx, yy, cex=1.2, pch=2)
lines(ex, ey, lwd=2)
lines(ex, 3 + 0.35 * (ex-15)  - 0.003 * (ex-16)^3, col="red")
```



4. Additive Model

(a) normal equation

eq(4.116): $y = m_1(x_1) + m_2(x_2) = c_0 + \sum_{j=1}^{p} c_j x_1^j + \sum_{j=1}^{q} d_j x_2^j + \varepsilon_i.$

Define X, y, b, e as below:

$$
\begin{aligned}
\mathrm{X} &= \begin{bmatrix} 1 & X_{11} & \cdots & X_{11}^p & X_{12} & \cdots & X_{12}^q \\ 1 & X_{21} & \cdots & X_{21}^p & X_{22} & \cdots & X_{22}^q \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & \cdots & X_{n1}^p & X_{n2} & \cdots & X_{n2}^q \end{bmatrix} \\
\mathrm{y} &= \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}^\top \\
\mathrm{b} &= \begin{bmatrix} c_0 & c_1 & \cdots & c_p & d_1 & \cdots & d_q \end{bmatrix}^\top \\
\mathrm{e} &= \begin{bmatrix} \varepsilon_1 & \cdots & \varepsilon_n \end{bmatrix}^\top
\end{aligned}
$$

Then, we can re-write eq(4.116) as follow:

$$\mathrm{y} = \mathrm{Xb} + \mathrm{e}$$

With simple regression, we get normal equation:

$$\mathrm{X}^\top \mathrm{Xb} = \mathrm{X}^\top \mathrm{y}$$

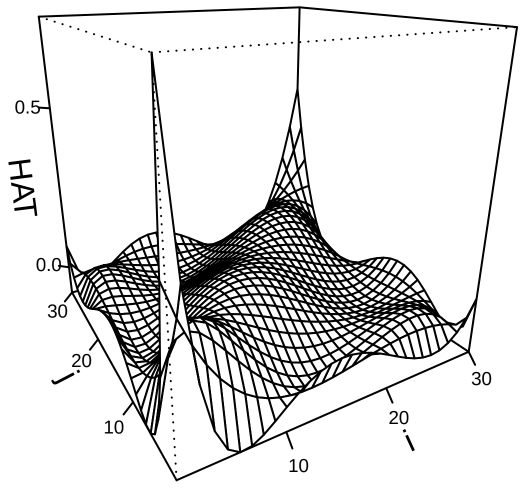
```r
normal_eq <- function(xx1, p, xx2, q, yy){
  X1 <- sapply(xx1, function(x) x^(0:p))
  X2 <- sapply(xx2, function(x) x^(1:q))
  X <- t(rbind(X1, X2))
  A <- t(X) %*% X
  b <- t(X) %*% yy
  list(A=A, b=b, X=X)
}

xx1 <- 1:30
xx2 <- (2:31)^1.1
yy <- -5 -0.03 * (xx1-15)^2 + 0.07 * (xx2-20)^2 + rnorm(30)
```

(b) Regression coefficients

```r
find_beta <- function(xx1, p, xx2, q, yy){
  Ab <- normal_eq(xx1, p, xx2, q, yy)
  beta <- solve(Ab$A, Ab$b)
  beta
}

find_beta(xx1, 2, xx2, 2, yy)
```

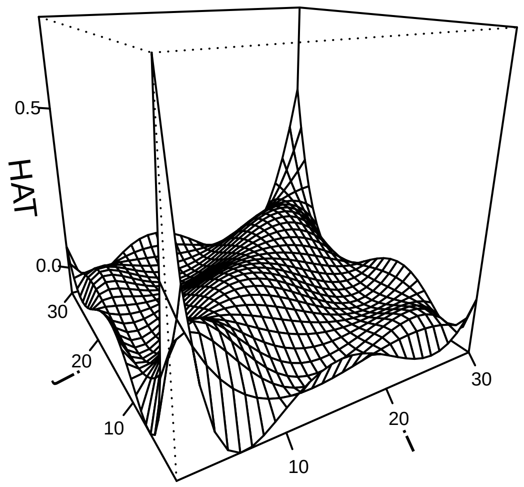
```
            [,1]
[1,]   2.0653343
[2,] -17.8555396
[3,]  -0.5474909
[4,]  12.6578709
[5,]   0.2623935
```

```r
additive_model <- function(xx1, p, xx2, q, yy){
  beta <- find_beta(xx1, p, xx2, q, yy)
  X <- normal_eq(xx1, p, xx2, q, yy)$X
  ey <- X %*% beta
  ey
}

hat_matrix <- function(xx1, p, xx2, q){
  nd <- length(xx1)
  hat <- apply(diag(nd), 1, additive_model, xx1=xx1, p=p, xx2=xx2, q=q)
  hat
}
hat <- hat_matrix(xx1, 2, xx2, 2)
persp(1:30, 1:30, hat, xlab = 'i', ylab='j', zlab='HAT',
        lab = c(3, 3, 3), theta = -30, phi = 20,
        ticktype = 'detailed', nticks=3, cex.lab = 1, cex.axis = 0.6)
```



(c) `poly()` and regression

```r
additive_model2 <- function(xx1, p, xx2, q, yy){
  formula.name <- yy ~ poly(xx1, degree=p) + poly(xx2, degree=q)
  data <- data.frame(xx1=xx1, xx2=xx2, yy=yy)
  lm.out <- lm(formula.name, data=data)
  ey <- lm.out$fitted.values
  ey
}

hat_matrix2 <- function(xx1, p, xx2, q){
  nd <- length(xx1)
  hat <- apply(diag(nd), 1, additive_model2, xx1=xx1, p=p, xx2=xx2, q=q)
  hat
}

hat2 <- hat_matrix(xx1, 2, xx2, 2)
persp(1:30, 1:30, hat2, xlab = 'i', ylab='j', zlab='HAT',
        lab = c(3, 3, 3), theta = -30, phi = 20,
        ticktype = 'detailed', nticks=3, cex.lab = 1, cex.axis = 0.6)
```



(d) identical hat matrix

```r
sum(hat - hat2) # 0 if all entries are identical
```

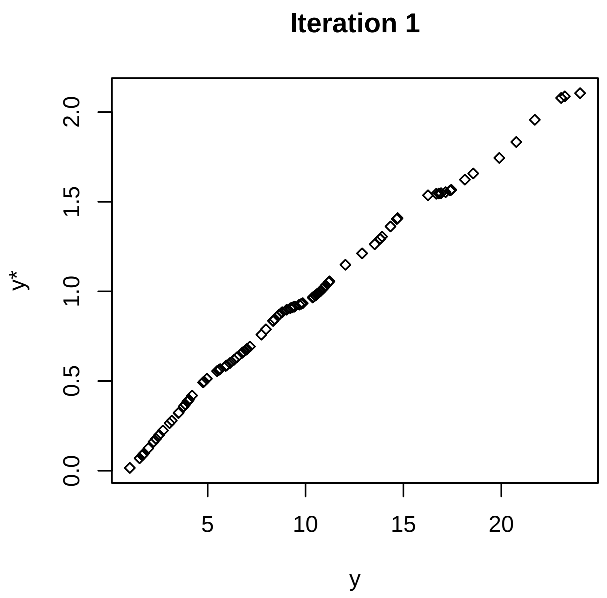
```
[1] 0
```

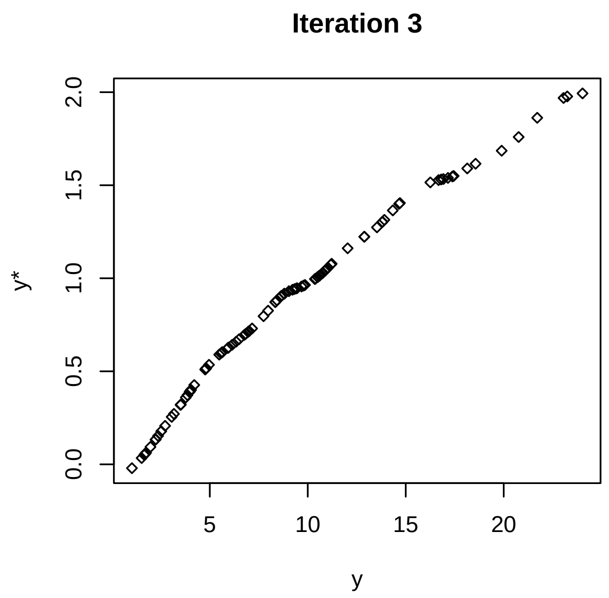
5. ACE with smoothing spline

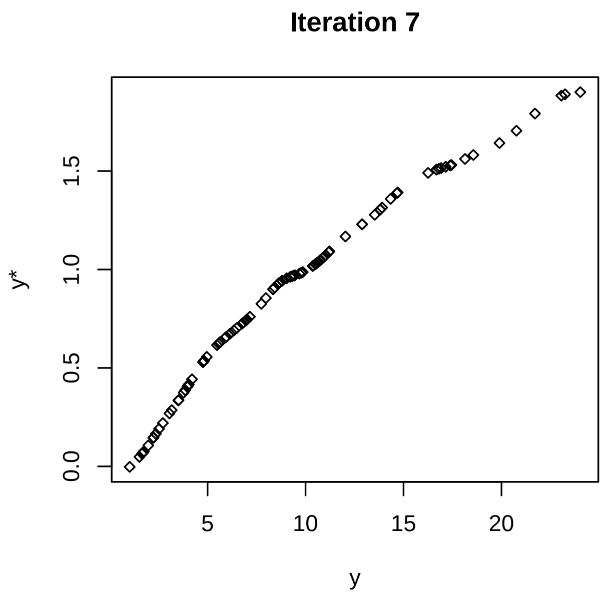
```r
aceit <- function(nd, it, lx1, lx2, ly, xx1, xx2, yy){
  yyst <- yy
  # smooth spline with y=m1(x1)+m2(x2)
  R1 <- sapply(xx1, function(x) abs(x-xx1)^3/12)
  R2 <- sapply(xx2, function(x) abs(x-xx2)^3/12)
  Q1 <- matrix(c(xx1^0, xx1), nrow=2, byrow=T)
  Q2 <- matrix(xx2, nrow=1)

  A <- matrix(0, nrow=2*nd+3, ncol=2*nd+3)
  A[1:nd,] <- cbind(R1 + lx1 * diag(nd), t(Q1), R2, t(Q2))
  A[(nd+1):(nd+2), 1:nd] <- Q1
  A[(nd+3):(2*nd+2),] <- cbind(R1, t(Q1), R2 + lx2 * diag(nd), t(Q2))
  A[2*nd+3, (nd+3):(2*nd+2)] <- Q2

  Hx <- cbind(R1, t(Q1), R2, t(Q2)) %*% solve(A)
  hatx <- Hx[,1:nd] + Hx[,(nd+3):(2*nd+2)]

  # smooth spline with y
  Ry <- sapply(yy, function(x) abs(x-yy)^3/12)
  Qy <- matrix(c(yy^0, yy), nrow=2, byrow=T)
  Ay <- matrix(0, nrow=nd+2, ncol=nd+2)
  Ay[1:nd,] <- cbind(Ry + ly * diag(nd), t(Qy))
  Ay[(nd+1):(nd+2), 1:nd] <- Qy

  Hy <- cbind(Ry, t(Qy)) %*% solve(Ay)
  haty <- Hy[,1:nd]

  hatyx <- haty %*% hatx
# (3)
  for(ii in 1:it) {
    yyst <- hatyx %*% yyst
    yyst <- (sqrt(nd) * yyst)/sqrt(sum(yyst^2))
  }
# (4)
  return(yyst)
}
```

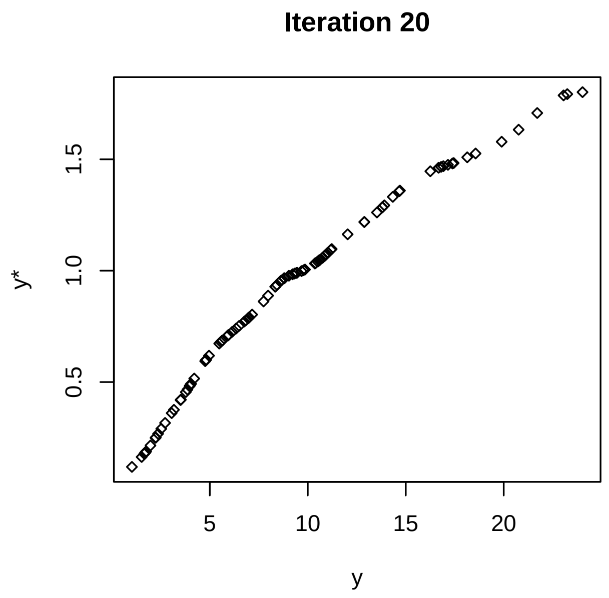
Here's an example.

```r
nd <- 100
xx1 <- runif(nd, min =0, max = 10)
xx2 <- runif(nd, min =0, max = 10)
yy <- (sin(0.2 * pi * xx1) + xx2^2 * 0.05 - xx2 * 0.4
  + 3 + rnorm(nd, mean =0, sd = 0.1))^2

par(mfrow = c(2, 2))

for(it in c(1,3,7,20)) {
  yyst <- aceit(nd, it, 0.3, 0.5, 0.4, xx1, xx2, yy)
  plot(yy, yyst, type = "n", xlab = "y", ylab = "y*", main = paste("Iteration", it))
  points(yy, yyst, cex = 0.7, pch =5)
  }
```


Iteration 1
y*
0.0
0.5
1.0
1.5
2.0
5
10
15
20
y


Iteration 3
y*
0.0
0.5
1.0
1.5
2.0
5
10
15
20
y


Iteration 7
y*
0.0
0.5
1.0
1.5
5
10
15
20
y


Iteration 20
y*
0.5
1.0
1.5
5
10
15
20
y