함수추정의 응용 및 실습

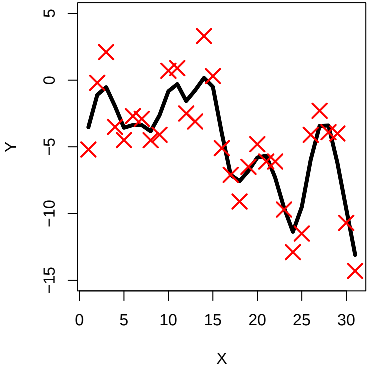
2. Smoothing for data with an equispaced predictor
2.7

Chae Young Lim

Seoul National University

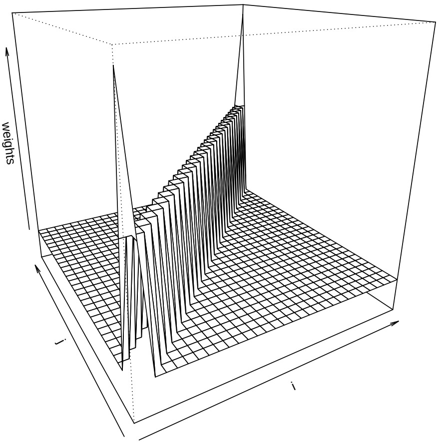
```r
##(A) Smoothing by the moving average
move1<-function(yy, mm){
#(1)
  nd <- length(yy)
#(2)
  yyr <- yy[(nd):(nd - mm + 1)]
  yyl <- yy [mm:1]
  y2 <- c(yyl, yy, yyr)
#(3)
  ey <- rep(0, length = nd)
#(4)
for(ii in 1:nd) {
    ey[ii] <- mean(y2[ii:(ii + 2 * mm)])
  }
#(5)
  return(ey)
}
```

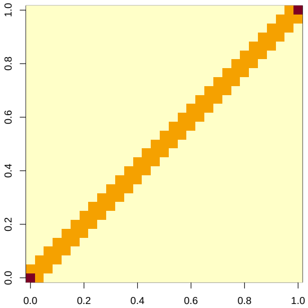
```r
## Figure 2.1 (left)

#(1)
mm <- 1
#(2)
xx <- seq(from = 1, by = 1, length =31)
#(3)
yy <- scan("wak2.csv")
#(4)
ey <- move1(yy, mm)
#(5)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5), oma = c(5, 5, 5, 5))
#(6)
plot(xx, ey, type='n' , ylim = c(-15, 5), xlab = "X", ylab = "Y")
lines(xx, ey, lwd =4)
#(7)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```


Y
5
0
−5
−10
−15
0
5
10
15
20
25
30
X

```r
##(B) Weights of the moving average
moveh1<-function(nd, mm2) {
#(1)
  iden <- diag(nd)
#(2)
  ww <- apply(iden, 2, move1, mm = mm2)
#(3)
  return(ww)
}
```

```r
## Figure 2.1 (right)
#(1)
nd <- 31
#(2)
mm <- 1
#(3)
ww <- moveh1(nd, mm)
#(4)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 1.5, 1.5), oma = c(5, 5, 5, 5))
#(5)
persp(ww, zlim =c(-0.1, 0.7), xlab = "i", ylab = "j", zlab = "weights",
      lab = c(3, 3, 3),theta=-30,phi=20)
#(6)
#invisible()

image(ww)
```


weights
j
i

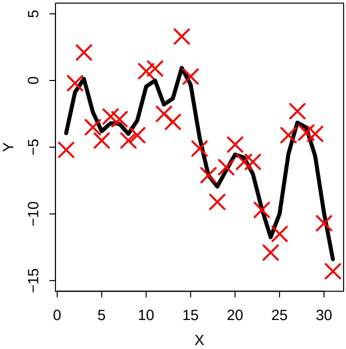


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

```r
##(C) Smoothing by the binomial filter
binom1<-function(yy, mm)
{
#(1)
  nd <- length(yy)
#(2)
  mm2 <- mm * 0.5
#(3)
  yywl <- yy; yyw2 <- yy; rlim <- mm2
  yyr <- NULL; count <- 0
  while(rlim > nd) {
    yyw1 <- rev(yyw1)
    yyr <- c(yyr, yyw1)
    rlim <- rlim - nd
    count <- count + 1
  }
  switch(count %% 2 + 1,
         yyr <- c(yyr, yy[nd:(nd - rlim + 1)]),
         yyr <- c(yyr, yy[1:rlim]))
  llim <- mm2; yyl <- NULL
  while (llim > nd) {
    yyw2 <- rev(yyw2)
    yyl <- c(yyw2, yyl)
    llim <- llim - nd
  }
```

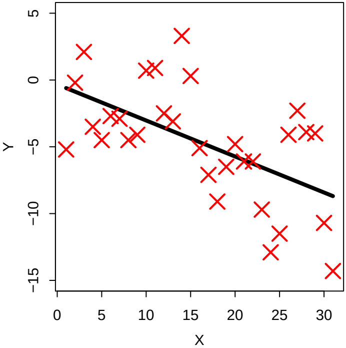
```r
  switch(count %% 2 + 1,
         yyl <- c(yy[llim:1], yyl),
         yyl <- c(yy[(nd - llim + 1):nd], yyl))
  y2 <- matrix(c(yyl, yy, yyr), ncol = 1)
#(4)
  ww <- matrix(0, ncol = nd + mm, nrow = nd + mm)
#(5)
  imat <- row(ww)
  jmat <- col(ww)
#(6)
  check <- 0 <= (mm2 + imat - jmat) & (mm2 + imat - jmat) <= mm
#(7)
  ww[check] <- exp(lgamma(mm +1) -
                   lgamma(mm2 + imat[check] - jmat[check] + 1) -
                   lgamma(mm2 - imat[check] + jmat[check] + 1) -
                   mm * logb(2))
#(8)
  ey <- ww %*% y2
  ey <- as.vector(ey[(mm2 + 1):(nd + mm2)])
#(9)
  return(ey)
}
```

```r
## Figure 2.4 (left)
#(1)
mm <- 2
#(2)
xx <- seq(from = 1, by = 1, length =31)
#(3)
yy <- scan("wak2.csv")
#(4)
ey <- binom1(yy, mm)
#(5)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
  oma = c(5, 5, 5, 5))
#(6)
plot(xx, ey, type = "n", ylim = c(-15, 5),
  xlab = "X", ylab = "Y")
lines(xx, ey, lwd =4)
#(7)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```



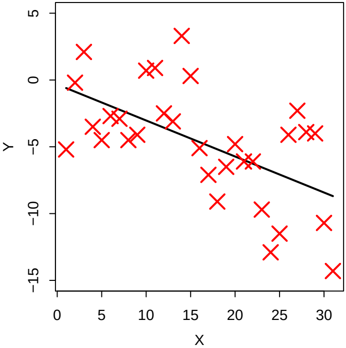
5
0
−5
−10
−15
Y
0
5
10
15
20
25
30
X

```r
##(D) Fitting of polynomials and Figure 2.8 (left)
#(1)
nd <- 31
xx <- seq(from = 1, by = 1, length = nd)
yy <- scan("wak2.csv")
datal <- data.frame(x = xx, y = yy)
#(2)
fit.lm <- lm(y ~ poly(x, degree = 1), data = datal)
#(3)
ey <- fitted.values(fit.lm)
#(4)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
  oma = c(5, 5, 5, 5))
plot(xx, ey, type = "n", ylim = c(-15, 5), xlab = "X",
  ylab = "Y")
lines(xx, ey, lwd =4)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```


5
0
−5
−10
−15
Y
0
5
10
15
20
25
30
X

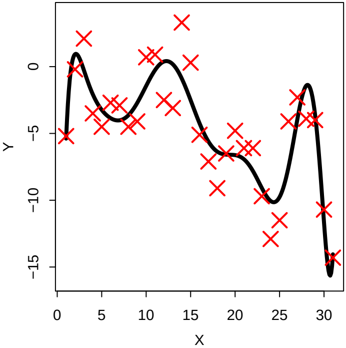
```r
## #(2) can be replaced with the below using substitute()
## the degree of polynomial is treated as an object (ndim)
#Object: substitute()
ndim <- 1
formula.name <- substitute(y ~ poly(x, degree = degreel),
                           list(degreel = ndim))
fit.lm <- lm(formula.name, data = datal)
```

```r
## Alternative way of fitting polynomials
#(1)
nd <- 31
xx <- seq(from = 1, by = 1, length = nd)
yy <- scan("wak2.csv")
ndim <- 1
#(2)
xm <- matrix(rep(0, length = nd * ndim), nrow = nd)
for(i in 1:ndim) {
    xm[, i] <- xx^i
}
#(3)
datal <- data.frame(x = xm, y = yy)
fit.lm <- lm(y ~ ., data = datal)
ey <- fitted.values(fit.lm)
#(4)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
    oma = c(5, 5, 5, 5))
plot(xx, ey, type = "n", ylim = c(-15, 5), xlab = "X",
    ylab = "Y")
lines(xx, ey, lwd =2)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```


Y
X
5
0
−5
−10
−15
0
5
10
15
20
25
30

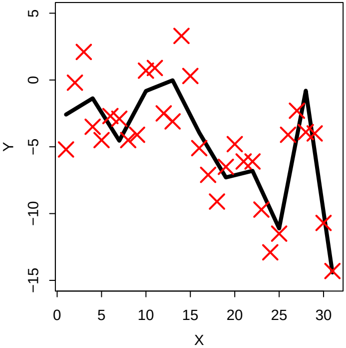
```r
## Another alternatives by replacing (2) to create xm
## with the below
xx <- 1:nd
power <- rep(1:ndim, rep(nd, ndim))
xm <- matrix(xx^power, nrow = nd)


## Also, the following can be used to derive xm
powerf <- function(jj, x1)
{
  pw <- x1^jj
  return(pw)
}
xm <- apply(matrix(c(1:ndim), nrow =1), 2, powerf, x1 = xx)
## When design matrix is obtained, (3) can be replaced with the below
fit.lin <- lsfit(xm, yy)
ey <- yy - fit.lin$residuals
```

```r
## (E)Fitting of a polynomial, and calculation of estimates
## that are placed at the positions where no data exist
## Figure 2.15 (left)
#(1)
nd <- 31
xx <- seq(from = 1, by = 1, length = nd)
ll <- list(r1 =0)
mm <- scan("wak2.csv", ll, sep = ",")
#(2)
yy <- mm$r1
#(3)
assign("data1", data.frame(x = xx, y = yy))
ex <- seq(from = 1, by = 0.1, length = nd * 10 - 9)
#(4)
data2 <- data.frame(x = ex)
fit.lm <- lm(y ~ poly(x, degree = 11), data = data1)
ey <- predict(fit.lm, newdata = data2)
#(5)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
    oma = c(5, 5, 5, 5))
plot(ex, ey, type = "n", ylim = c(-16, 4), xlab = "X",
      ylab = "Y")
lines(ex, ey, lwd =4)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```



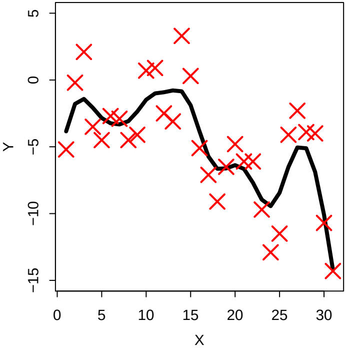
0
−5
−10
−15
Y
0
5
10
15
20
25
30
X

```r
##(F) Fitting of the spline function and Figure 2.19 (left)
library(splines)
#(1)
nd <- 31
xx <- seq(from = 1, by = 1, length = nd)
yy <- scan("wak2.csv")
#(2)
data1 <- data.frame(x = xx, y = yy)
#(3)
fit.lm <- lm(y ~ bs(x, knots = c(4, 7, 10, 13, 16, 19, 22, 25, 28),
                    degree =1), data = data1)
#(4)
ey <- fitted.values(fit.lm)
#(5)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
    oma = c(5, 5, 5, 5))
plot(xx, ey, type = "n", ylim = c(-15, 5),
      xlab = "X", ylab = "Y")
lines(xx, ey, lwd =4)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```


5
0
−5
−10
−15
Y
0
5
10
15
20
25
30
X

```r
##(G)Local linear regression for equispaced predictor
lline<-function(yy, hh)
{
#(1)
  llin <- function(ex1, xdata, ydata, band)
  {
#(2)
    wts <- exp((-0.5 * (ex1 - xdata)^2)/band^2)
#(3)
    data1 <- data.frame(x = xdata, y = ydata, www = wts)
#(4)
    fit.lm <- lm(y ~ x, data = data1, weights = www)
#(5)
    est <- fit.lm$coef[1] + fit.lm$coef[2] * ex1
#(6)
    return(est)
  }
#(7)
  nd <- length(yy)
  xx <- seq(from = 1, by = 1, length = nd)
#(8)
  xxmat <- matrix(xx, ncol = 1)
#(9)
  ey <- apply(xxmat, 1, llin, xdata = xx, ydata = yy,
              band = hh)
  ey <- as.vector(ey)
#(10)
  return(ey)
}
```

```r
## Figure 2.23 (left)
#(1)
yy <- scan("wak2.csv")
nd <- length(yy)
xx <- seq(from = 1, by = 1, length = nd)
#(2)
hh <- 1.5
#(3)
ey <- lline(yy, hh)
#(4)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
    oma = c(5, 5, 5, 5))
plot(xx, ey, type = "n", ylim = c(-15, 5), xlab = "X",
      ylab = "Y")
lines(xx, ey, lwd =4)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```

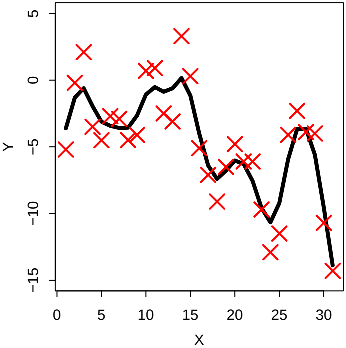


5
0
−5
−10
−15
Y
0
5
10
15
20
25
30
X

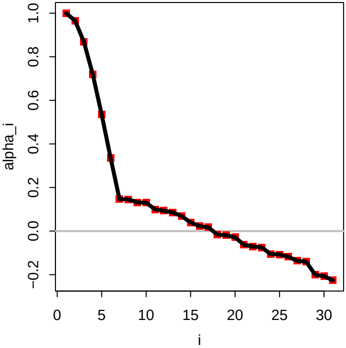
```r
##(H)Smoothing spline for an equispaced predictor
smspe<-function(yy, lambda)
{
#(1)
  nd <- length(yy)
#(2)
  ss <- c(1, -2, 1, rep(0, nd - 3))
  ss <- rbind(ss, c(-2, 5, -4, 1, rep(0, length = nd - 4)))
  for(ii in 1:(nd - 4)) {
    ss <- rbind(ss, c(rep(0, ii - 1), 1, -4, 6, -4, 1,
                      rep(0, nd - ii - 4)))
  }
  ss <- rbind(ss, c(rep(0, length = nd - 4), 1, -4, 5, -2))
  ss <- rbind(ss, c(rep(0, length = nd - 3), 1, -2, 1))
#(3)
  ssi <- diag(nd) + lambda * ss
#(4)
  ey <- solve(ssi, yy)
  ey <- as.vector(ey)
#(5)
  return(ey)
}
```

```r
## Alternative way of replacing (2)
#(2)
ss <- matrix(rep(0, length = nd * nd), ncol = nd)
ss[1, 1:3] <- c(1, -2, 1)
ss[2, 1:4] <- c(-2, 5, -4, 1)
for(ii in 3:(nd - 2)) {
  ss[ii, (ii - 2):(ii + 2)] <- c(1, -4, 6, -4, 1)
}
ss[(nd - 1), (nd - 3):nd] <- c(1, -4, 5, -2)
ss[nd, (nd - 2):nd] <- c(1, -2, 1)
```

```r
## Figure 2.28(left)
#(1)
yy <- scan("wak2.csv")
#(2)
lambda <- 1
#(3)
ey <- smspe(yy, lambda)
#(4)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
    oma = c(5, 5, 5, 5))
xx <- seq(from = 1., by = 1., length = length(yy))
plot(xx, ey, type = "n", ylim = c(-15, 5),
      xlab = "X", ylab = "Y")
lines(xx, ey, lwd =4)
points(xx, yy, pch =4, lwd=2, col='red',cex=2)
```


Y
X
5
0
−5
−10
−15
0
5
10
15
20
25
30

```r
## Eigenvalues of a hat matrix from moving average with m=4
## Figure 2.44
#(1)
nd <- 31
mm <- 4
#(2)
iden <- diag(nd)
#(3)
ww <- apply(iden, 2, move1, mm = mm)
#(4)
eigen1 <- eigen(ww, symmetric = T)
value1 <- eigen1$values
#(5)
par(mfrow = c(1, 1), mai = c(1.5, 1.5, 0.5, 0.5),
    oma = c(5, 5, 5, 5))
xx <- seq(from = 1, to = 31, by = 1)
plot(xx, value1, type = "n", xlab = "i", ylab = "alpha_i")
abline(h=0,col='gray',lwd=2)
points(xx, value1, pch = 15,  lwd=2, col='red')
lines(xx, value1, lwd = 4)
```



1.0
0.8
0.6
0.4
0.2
0.0
−0.2
alpha_i
0
5
10
15
20
25
30
i