```
# Modified on 29/Nov/2004.
# This is for an empirical example (quantile = 0.5).
# Revised version for JASA
# Choose kappa by BIC
# Choose h by a rule-of-thumb by Fan and Gijbels
# Use a truncated normal K

rm(list = ls()) # Initialization
options(digits = 4) # Set digits to be printed

sink("ej-aqr3.txt", append=TRUE)
cat(paste("The program started:",date()), "\n")
sink()

library(quantreg)
library(splines)

################################
### User-Defined Function   ###
################################

# This routine is to compute a truncated, normalized standard normal density (truncated at 3 times sigma).

"dtnorm" <- function(u) {
          tmp1 <- dnorm(u)*(abs(u)<3.5)
          tmp2 <- 1 - 2*pnorm(3.5, mean=0, sd=1, lower.tail = FALSE, log.p = FALSE)
            fv <- tmp1/tmp2
          fv
}

# This routine is to compute the local linear quantile regression estimator.

"lprq" <- function(x, y, h, xx, tau) {
        fv <- xx
        for (i in 1:length(xx)){
            z1 <- x - xx[i]
            wx <- dtnorm(z1/h)
             r <- rq(y ~ z1, weights = wx, tau = tau, ci = FALSE)
          fv[i] <- r$coef[1]
       }
       fv
}

# This routine is to compute the kernel density estimator when X is one-dimensional.

"kde1" <- function(x, h, xx) {
        fv <- xx
        for (i in 1:length(xx)){
            z1 <- x - xx[i]
            wx <- dtnorm(z1/h)
             r <- sum(wx)/(length(x)*h)
          fv[i] <- r
       }
```

```
      fv
}

# This routine is to compute the kernel density estimator when X is two-dimensional.

"kde2" <- function(x1, x2, h1, h2, xx1, xx2) {
        fv <- xx1
        for (i in 1:length(xx1)){
            z1 <- x1 - xx1[i]
            z2 <- x2 - xx2[i]
            wx <- dtnorm(z1/h1)*dtnorm(z2/h2)
             r <- sum(wx)/(length(x1)*h1*h2)
          fv[i] <- r
       }
       fv
}


################################
###    Data Loading         ###
################################

d<-read.table(file="ej.data",header=TRUE)
bl<-data.frame(d)
attach(bl)

  y <- mh5
xx1 <- top10
xx2 <- log(size)
xx3 <- age
xx4 <- leverage


x1 <- (xx1-mean(xx1))/sd(xx1)
x2 <- (xx2-mean(xx2))/sd(xx2)
x3 <- (xx3-mean(xx3))/sd(xx3)
x4 <- (xx4-mean(xx4))/sd(xx4)


################################
### Model Specification     ###
################################


   n <- length(y) # sample size
tau0 <- 0.50      # quantile of interest
 deg <- 3         # cubic splines



########################################
### Choosing a best kappa using QBIC ###
########################################
```

```
   kk <- c(3,4,5,6,7,8,9,10)
qbic_r <- rep(0,length(kk))

for (ki in 1:length(kk)){

 k <- kk[ki] # number of approximating functions

z1 <- bs(x1, df = k, degree = deg)  # B-splines with kappa_n = k
for (i in 1:k){                  # Normalization (sample mean zero assumption)
z1[,i] <- z1[,i] - mean(z1[,i])
}

z2 <- bs(x2, df = k, degree = deg)
for (i in 1:k){
z2[,i] <- z2[,i] - mean(z2[,i])
}

z3 <- bs(x3, df = k, degree = deg)
for (i in 1:k){
z3[,i] <- z3[,i] - mean(z3[,i])
}

z4 <- bs(x4, df = k, degree = deg)
for (i in 1:k){
z4[,i] <- z4[,i] - mean(z4[,i])
}

#####################################################
### Preliminary First-Stage Series Estimation  ###
#####################################################

fit <- rq(y ~ z1+z2+z3+z4, tau = tau0, ci = FALSE)

 b <- coef(fit)
b0 <- b[1]           # Intercept term
 b <- b[2:length(b)]
b1 <- b[1:k]           # Coefficients for f1
b2 <- b[(k+1):(2*k)]    # Coefficients for f2
b3 <- b[(2*k+1):(3*k)]  # Coefficients for f3
b4 <- b[(3*k+1):(4*k)]  # Coefficients for f4

f1h <- z1%*%b1       # Series estimates of f1
f2h <- z2%*%b2       # Series estimates of f2
f3h <- z3%*%b3       # Series estimates of f3
f4h <- z4%*%b4       # Series estimates of f4

       uh <- y - b0 - f1h - f2h - f3h - f4h # estimated residuals
    qloss <- abs(uh) + (2*tau0 - 1)*uh
   qbic_k <- n*log(sum(qloss)) + log(n)*k*2    # QBIC

qbic_r[ki] <- qbic_k
}

best_k <- kk[rank(qbic_r)==1]     # kappa that minimizes QBIC
```

```
#############################################################################
### Choosing bandwidths using a rule-of-thumb (Fan and Gijbels, 1996, p.202) ###
#############################################################################

   x1_2 <- x1^2
   x1_3 <- x1^3
   x1_4 <- x1^4
   x2_2 <- x2^2
   x2_3 <- x2^3
   x2_4 <- x2^4
   x3_2 <- x3^2
   x3_3 <- x3^3
   x3_4 <- x3^4
   x4_2 <- x4^2
   x4_3 <- x4^3
   x4_4 <- x4^4


    pqr <- rq(y ~ x1+x1_2+x1_3+x1_4+x2+x2_2+x2_3+x2_4+x3+x3_2+x3_3+x3_4+x4+x4_2+x4_3+x4_4, tau = tau0,
ci = FALSE)
     pb <- coef(pqr)
   pqre <- resid(pqr)
    qre <- quantile(pqre, probs = tau0)
  h_kde <- 1.06*sd(pqre)*n^(-1/5)
    qrd <- kde1(pqre,h_kde, qre)

dev2_x1 <- pb[3]*2 + pb[4]*(6*x1) + pb[5]*(12*x1^2)
dev2_x2 <- pb[7]*2 + pb[8]*(6*x2) + pb[9]*(12*x2^2)
dev2_x3 <- pb[11]*2 + pb[12]*(6*x3) + pb[13]*(12*x3^2)
dev2_x4 <- pb[15]*2 + pb[16]*(6*x4) + pb[17]*(12*x4^2)

   w_x1 <- (abs(x1) <= 2)
   w_x2 <- (abs(x2) <= 2)
   w_x3 <- (abs(x3) <= 2)
   w_x4 <- (abs(x4) <= 2)

    c_x <- tau0*(1-tau0)*4
    num <- c_x/(qrd^2)
    den <- (dev2_x1^2)*w_x1
 den_x1 <- sum(den)
    den <- (dev2_x2^2)*w_x2
 den_x2 <- sum(den)
    den <- (dev2_x3^2)*w_x3
 den_x3 <- sum(den)
    den <- (dev2_x4^2)*w_x4
 den_x4 <- sum(den)

     h1 <- 0.776*((num/den_x1)^(1/5))  # Bandwidth
     h2 <- 0.776*((num/den_x2)^(1/5))  # Bandwidth
     h3 <- 0.776*((num/den_x3)^(1/5))  # Bandwidth
```

```
    h4 <- 0.776*((num/den_x4)^(1/5))  # Bandwidth



################################
###    Linear Models       ###
################################

mr <- lm(y ~ xx1+xx2+xx3+xx4)
qr <- rq(y ~ xx1+xx2+xx3+xx4, tau0)

############################################
### Construction Approximating functions ###
############################################

 k <- best_k + 1                 # overfitting

z1 <- bs(x1, df = k, degree = deg)  # B-splines with kappa_n = k
for (i in 1:k){                 # Normalization (sample mean zero assumption)
z1[,i] <- z1[,i] - mean(z1[,i])
}

z2 <- bs(x2, df = k, degree = deg)
for (i in 1:k){
z2[,i] <- z2[,i] - mean(z2[,i])
}

z3 <- bs(x3, df = k, degree = deg)
for (i in 1:k){
z3[,i] <- z3[,i] - mean(z3[,i])
}

z4 <- bs(x4, df = k, degree = deg)
for (i in 1:k){
z4[,i] <- z4[,i] - mean(z4[,i])
}




#####################################
### First-Stage Series Estimation  ###
#####################################

fit <- rq(y ~ z1+z2+z3+z4, tau = tau0, ci = FALSE)

 b <- coef(fit)
b0 <- b[1]              # Intercept term
 b <- b[2:length(b)]
b1 <- b[1:k]            # Coefficients for f1
b2 <- b[(k+1):(2*k)]    # Coefficients for f2
b3 <- b[((2*k)+1):(3*k)]  # Coefficients for f3
```

```
b4 <- b[((3*k)+1):(4*k)]  # Coefficients for f4


f1h <- z1%*%b1       # Series estimates of f1
f2h <- z2%*%b2       # Series estimates of f2
f3h <- z3%*%b3       # Series estimates of f3
f4h <- z4%*%b4       # Series estimates of f4



##############################################
### Second-Stage Local Linear Estimation ###
##############################################

  ys1 <- y - b0 - f2h - f3h - f4h


f1hat <- lprq(x1, ys1, h1, x1, tau0)

   u1 <- ys1 - f1hat
  hu1 <- sd(u1)*(n^(-1/6))
  hx1 <- n^(-1/6)
   hx <- n^(-1/5)
  u00 <- (1:n)*0
 tmp1 <- kde2(u1, x1, hu1, hx1, u00, x1)
 tmp2 <- kde1(x1, hx, x1)
 tmp3 <- (tmp1^2)/tmp2

  se1 <- sqrt(0.1407*tau0*(1-tau0)/(tmp3*n*h1))
  lb1 <- f1hat - 1.645*se1
  ub1 <- f1hat + 1.645*se1

##############################################

  ys2 <- y - b0 - f1h - f3h - f4h


f2hat <- lprq(x2, ys2, h2, x2, tau0)

   u2 <- ys2 - f2hat
  hu2 <- sd(u2)*(n^(-1/6))
  hx2 <- n^(-1/6)
   hx <- n^(-1/5)
  u00 <- (1:n)*0
 tmp1 <- kde2(u2, x2, hu2, hx2, u00, x2)
 tmp2 <- kde1(x2, hx, x2)
 tmp3 <- (tmp1^2)/tmp2

  se2 <- sqrt(0.1407*tau0*(1-tau0)/(tmp3*n*h2))
  lb2 <- f2hat - 1.645*se2
  ub2 <- f2hat + 1.645*se2

##############################################
```

```
  ys3 <- y - b0 - f1h - f2h - f4h


f3hat <- lprq(x3, ys3, h3, x3, tau0)

   u3 <- ys3 - f3hat
  hu3 <- sd(u3)*(n^(-1/6))
  hx3 <- n^(-1/6)
   hx <- n^(-1/5)
  u00 <- (1:n)*0
 tmp1 <- kde2(u3, x3, hu3, hx3, u00, x3)
 tmp2 <- kde1(x3, hx, x3)
 tmp3 <- (tmp1^2)/tmp2

  se3 <- sqrt(0.1407*tau0*(1-tau0)/(tmp3*n*h3))
  lb3 <- f3hat - 1.645*se3
  ub3 <- f3hat + 1.645*se3


##############################################

  ys4 <- y - b0 - f1h - f2h - f3h


f4hat <- lprq(x4, ys4, h4, x4, tau0)

   u4 <- ys4 - f4hat
  hu4 <- sd(u4)*(n^(-1/6))
  hx4 <- n^(-1/6)
   hx <- n^(-1/5)
  u00 <- (1:n)*0
 tmp1 <- kde2(u4, x4, hu4, hx4, u00, x4)
 tmp2 <- kde1(x4, hx, x4)
 tmp3 <- (tmp1^2)/tmp2

  se4 <- sqrt(0.1407*tau0*(1-tau0)/(tmp3*n*h4))
  lb4 <- f4hat - 1.645*se4
  ub4 <- f4hat + 1.645*se4

##############################################

##############################################
### Residuals after two-step estimation  ###
##############################################

resid <- y - b0 - f1hat - f2hat - f3hat - f4hat


#####################################
###    Estimation Results        ###
#####################################

sink("ej-aqr3.txt", append=TRUE)
cat(" Quantile = ", tau0, "\n")
```

```
cat(" Dimension of X = ", 4, "\n")
cat(" Sample Size = ", n, "\n")
cat(" Kappa = ", k, "\n")
cat(" h1 = ", h1, "\n")
cat(" h2 = ", h2, "\n")
cat(" h3 = ", h3, "\n")
cat(" h4 = ", h4, "\n")
sink()


#########################################################
###    Drawing Figures of Estimated Functions     ###
#########################################################

tmp1 <- order(xx1)
tmp2 <- xx1[tmp1]
tmp3 <- f1hat[tmp1]
tlb1 <- lb1[tmp1]
tub1 <- ub1[tmp1]

tmp4 <- order(xx2)
tmp5 <- xx2[tmp4]
tmp6 <- f2hat[tmp4]
tlb2 <- lb2[tmp4]
tub2 <- ub2[tmp4]

tmp7 <- order(xx3)
tmp8 <- xx3[tmp7]
tmp9 <- f3hat[tmp7]
tlb3 <- lb3[tmp7]
tub3 <- ub3[tmp7]

tmp10 <- order(xx4)
tmp11 <- xx4[tmp10]
tmp12 <- f4hat[tmp10]
tlb4 <- lb4[tmp10]
tub4 <- ub4[tmp10]



postscript("fig-med.eps", horizontal=FALSE)
split.screen(c(2,2))

screen(1)
plot(tmp2, tmp3, ylim = c(min(tlb1),max(tub1)), type="l", col = "blue", xlab = " TOPTEN ", ylab = " ", cex = 0.7,
cex.axis = 0.85)
lines(tmp2, tlb1,  lty=2, col = "dark grey")
lines(tmp2, tub1,  lty=2, col = "dark grey")

screen(2)
plot(tmp5, tmp6, ylim = c(min(tlb2),max(tub2)), type="l", col = "blue", xlab = " Log(Assets) ", ylab = " ", cex = 0.7,
cex.axis = 0.85)
lines(tmp5, tlb2,  lty=2, col = "dark grey")
lines(tmp5, tub2,  lty=2, col = "dark grey")
```

```
screen(3)
plot(tmp8, tmp9, ylim = c(min(tlb3),max(tub3)), type="l", col = "blue", xlab = " Age ", ylab = " ", cex = 0.7, cex.axis = 0.85)
lines(tmp8, tlb3,  lty=2, col = "dark grey")
lines(tmp8, tub3,  lty=2, col = "dark grey")

screen(4)
plot(tmp11, tmp12, ylim = c(min(tlb4),max(tub4)), type="l", col = "blue", xlab = " Leverage ", ylab = " ", cex = 0.7, cex.axis = 0.85)
lines(tmp11, tlb4,  lty=2, col = "dark grey")
lines(tmp11, tub4,  lty=2, col = "dark grey")
lines(tmp11, tub4,  lty=2, col = "dark grey")
lines(tmp11, tub4,  lty=2, col = "dark grey")

close.screen(all = TRUE)

dev.off

##########################################
###    Drawing Figures of Rediduals     ###
##########################################


tmp1 <- order(xx1)
tmp2 <- xx1[tmp1]
tmp3 <- resid[tmp1]


tmp4 <- order(xx2)
tmp5 <- xx2[tmp4]
tmp6 <- resid[tmp4]

tmp7 <- order(xx3)
tmp8 <- xx3[tmp7]
tmp9 <- resid[tmp7]


tmp10 <- order(xx4)
tmp11 <- xx4[tmp10]
tmp12 <- resid[tmp10]




postscript("fig-resid.eps", horizontal=FALSE)
split.screen(c(2,2))

screen(1)
plot(tmp2, tmp3, ylim = c(min(tmp3),max(tmp3)), type="p", col = "blue", xlab = " TOPTEN ", ylab = " ", cex = 0.7, cex.axis = 0.85)
abline(h=0)

screen(2)
```

```
plot(tmp5, tmp6, ylim = c(min(tmp6),max(tmp6)), type="p", col = "blue", xlab = " Log(Assets) ", ylab = " ", cex = 0.7, cex.axis = 0.85)
abline(h=0)

screen(3)
plot(tmp8, tmp9, ylim = c(min(tmp9),max(tmp9)), type="p", col = "blue", xlab = " Age ", ylab = " ", cex = 0.7, cex.axis = 0.85)
abline(h=0)

screen(4)
plot(tmp11, tmp12, ylim = c(min(tmp12),max(tmp12)), type="p", col = "blue", xlab = " Leverage ", ylab = " ", cex = 0.7, cex.axis = 0.85)
abline(h=0)
plot(tmp11, tmp12, ylim = c(min(tmp12),max(tmp12)), type="p", col = "blue", xlab = " Leverage ", ylab = " ", cex = 0.7, cex.axis = 0.85)

close.screen(all = TRUE)

dev.off



cat(paste("The program ended:", date()), "\n")
sink()
```