Estimating the conditional variance by local linear regression

Daniel Fuentes, Marc Valentí

Aircraft Data

Calling library **sm**.

```
library(sm)
library(latex2exp)
library(dplyr)
library(KernSmooth)
```

Reading the data and taking logarithms.

```
dt<-aircraft %>%
  select(Yr,Weight) %>%
  mutate(lgWeight=log(Weight)) %>%
  arrange(Yr)
attach(dt)
```

Estimating the conditional variance

```
locpolreg <- function(x,
                      y,
                      h,
                      q=1,
                      r=0,
                      tg=NULL,
                      type.kernel="normal",
                      nosubplot=FALSE,
                      doing.plot=TRUE, ...){
   if (is.null(tg)){tg<-x}
   aux <- sort(tg,index.return=T)
   sorted.tg <- tg[aux$ix]
   sorted.tg.ix <- aux$ix

   n <- length(x);
   m <- length(tg);
   mtgr <- numeric(m);
   S <- matrix(0,nrow=m,ncol=n)

   for (i in seq(1,m)){
     #estima un kernel normal para cada x
      aux <- kernel((x-tg[i])/h,type=type.kernel);
      #valores positivos
      Ih <- (aux>0);
      #numero de valores positivos (aunque en un kernel normal siempre lo son todos)
      ni <- sum(Ih);
```

```
      #diferencia entre los
      xh <- x[Ih]-tg[i];

      Dq <- matrix(1,nrow=ni,ncol=q+1);
      if (q>0){for (j in 1:q) Dq[,j+1] <- xh^j}
      Wx <- kernel(xh/h,type=type.kernel)/h;
      Wm <- Wx%*%ones(1,q+1);
      Dqq <- Wm*Dq;
      Si <- solve(t(Dq)%*%Dqq)%*%t(Dqq);
      beta <- Si%*%y[Ih];
      #Estimated values of the r-th derivative of the regression function at points in vector tg
      mtgr[i] <- factorial(r)*beta[r+1];
      #The Ssmoothing matrix
      S[i,Ih] <- Si[r+1,]
   }

   if (doing.plot){
      if (r==0){
        if (nosubplot) par(mfrow=c(1,1))
        plot(x,y,col="grey",...)
        lines(sorted.tg,mtgr[sorted.tg.ix],col=1,lwd=2)
      }
      else{
         par(mfrow=c(2,1))
         aux <- locpolreg(x,y,h,q,0,tg,nosubplot=F,type.kernel,...)
         plot(sorted.tg,mtgr[sorted.tg.ix],type="n",
              xlab="x",ylab="Estimated derivative")
         abline(h=0,col=4)
         lines(sorted.tg,mtgr[sorted.tg.ix],col=1,lwd=2)
      }
   }
return(list(mtgr=mtgr,S=S))
}

epan <- function(x){pmax(.75*(x+1)*(1-x))}
kernel <- function(x,type=c("normal","epan","rs.epan","unif")){
   switch(type[1],
          epan = pmax(.75*(x+1)*(1-x),0),
          rs.epan = pmax(.75*(x/sqrt(5)+1)*(1-x/sqrt(5))/sqrt(5),0),
          unif = as.numeric( (abs(x)<=1) )/2,
          dnorm(x))
}
ones <- function(n,m){matrix(1,nrow=n,ncol=m)}



h.cv.gcv <- function(x,y,h.v = exp(seq(log(diff(range(x))/20),
                                       log(diff(range(x))/4),l=10)),
                     p=1,type.kernel="normal"){
  n <- length(x)
  cv <- h.v*0
  gcv <- h.v*0
  for (i in (1:length(h.v))){
```

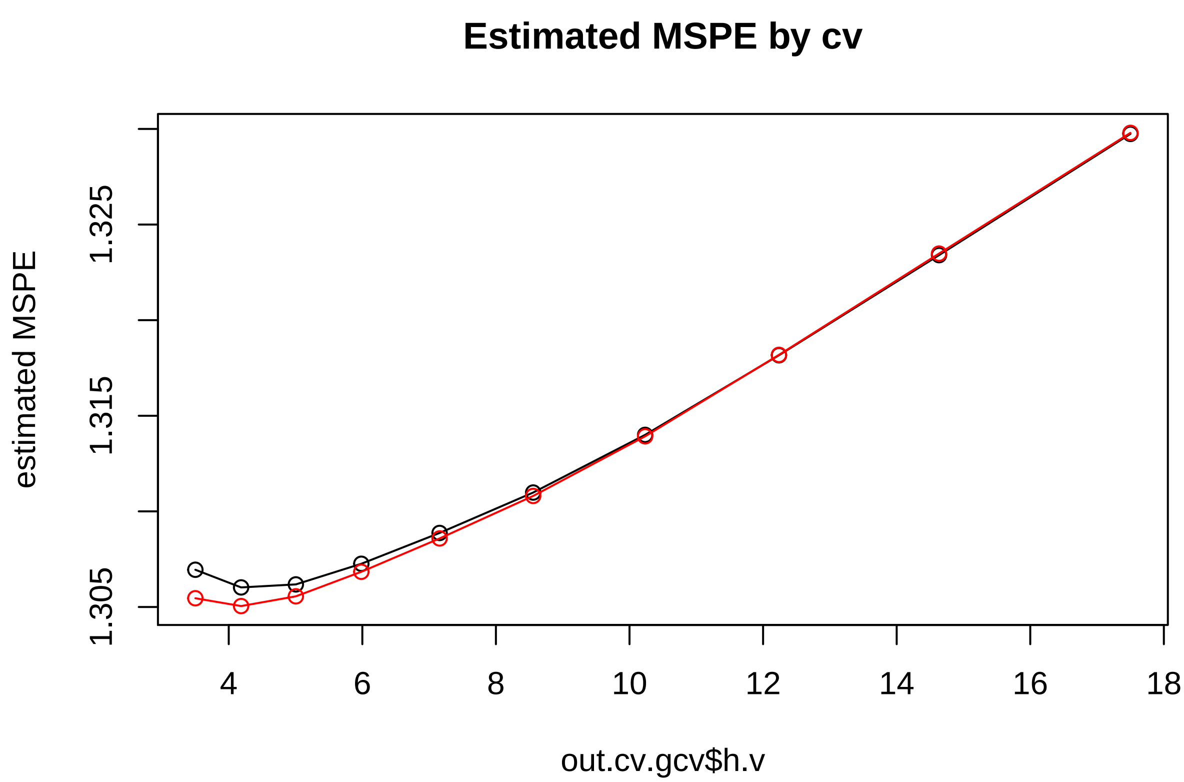
```r
    h <- h.v[i]
    aux <- locpolreg(x=x,y=y,h=h,p=p,tg=x,
                     type.kernel=type.kernel, doing.plot=FALSE)
    S <- aux$S
    h.y <- aux$mtgr
    hii <- diag(S)
    av.hii <- mean(hii)
    cv[i] <- sum(((y-h.y)/(1-hii))^2)/n
    gcv[i] <- sum(((y-h.y)/(1-av.hii))^2)/n
  }
  return(list(h.v=h.v,cv=cv,gcv=gcv))
}
```

```r
out.cv.gcv<-h.cv.gcv(x=Yr,
         y=lgWeight)

y.max <- max(c(out.cv.gcv$cv,out.cv.gcv$gcv))
y.min <- min(c(out.cv.gcv$cv,out.cv.gcv$gcv))

plot(out.cv.gcv$h.v,out.cv.gcv$cv,ylim=c(y.min,y.max),ylab="estimated MSPE",
     main="Estimated MSPE by cv")
lines(out.cv.gcv$h.v,out.cv.gcv$cv)

points(out.cv.gcv$h.v,out.cv.gcv$gcv,col=2)
lines(out.cv.gcv$h.v,out.cv.gcv$gcv,col=2)
```



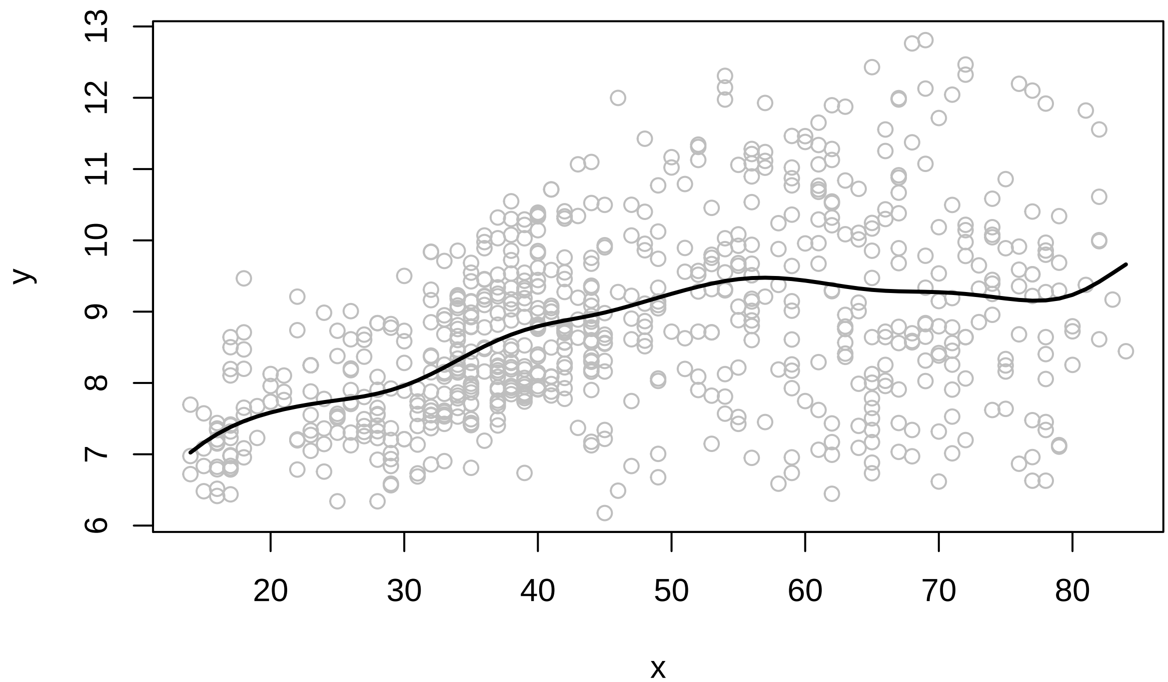
```
opt.h.cv <- out.cv.gcv$h.v[which.min(out.cv.gcv$gcv)]

print(opt.h.cv)
```

```
## [1] 4.185346
```

1.Fit a nonparametric regression to data (xi,yi) and save the estimated values m(xi). -FIRST TIME-

```
 lpr1<-locpolreg(x=Yr,
                 y=lgWeight,
                 opt.h.cv,
                 doing.plot = TRUE)
```

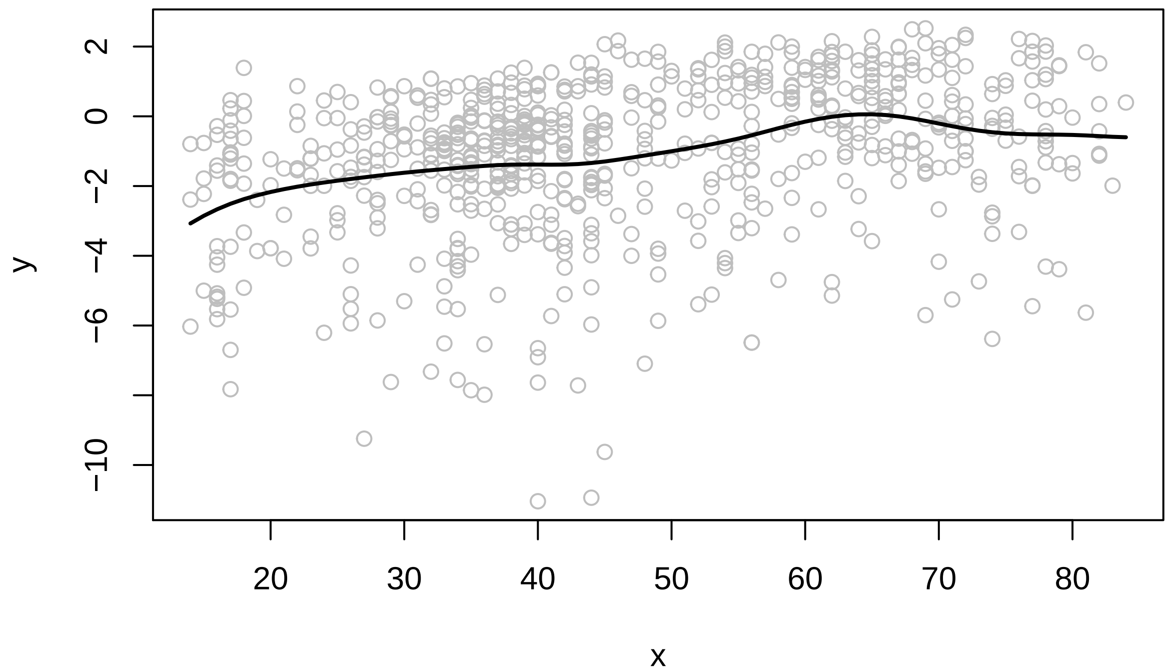


2.Transform the estimated residuals -FIRST TIME-

```
mtgr<-lpr1$mtgr #a mtgr guardem l'estimaci? de la variable y
resmtgr<-lgWeight-mtgr #residus
z= log(resmtgr^2) #transformaci? dels residus
```

3.Fit a nonparametric regression to data (xi, zi) and call the estimated function q(x). Observe that q(x) is an estimate of log var(x) -FIRST TIME-

```
 lpr3<-locpolreg(x=Yr,
                 y=z,
                 opt.h.cv)
```

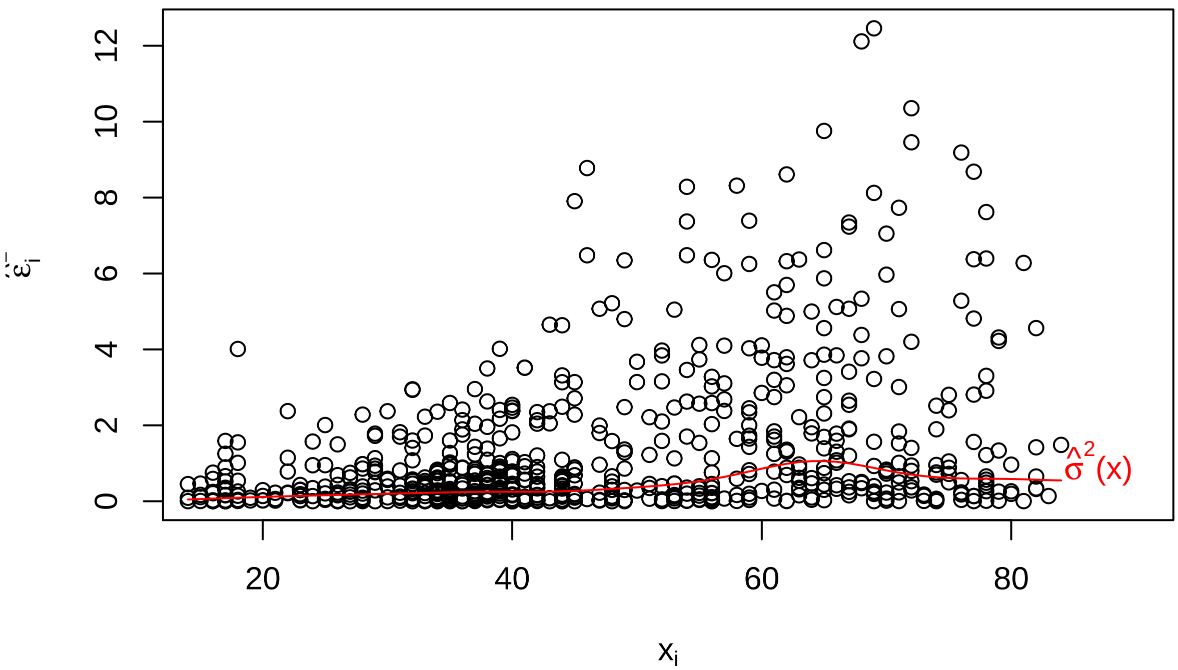


4.Estimate var(x) -FIRST TIME-

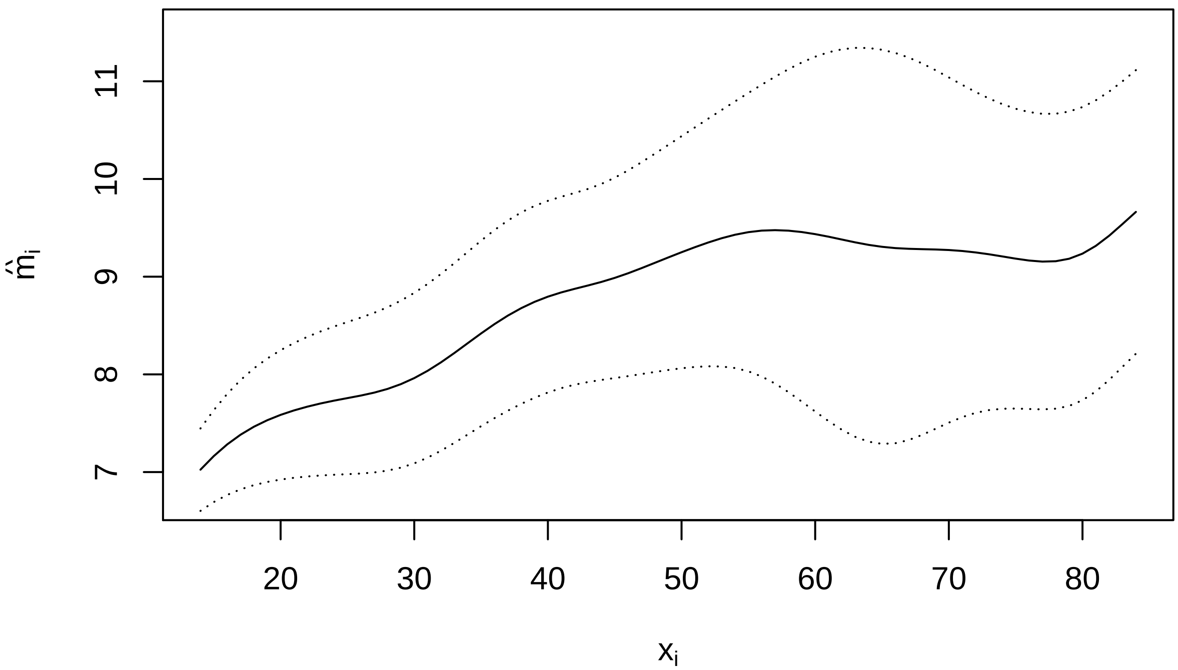
```
Vx<-exp(lpr3$mtgr)
```

Draw a graphic of resmtgr^2 against xi and superimpose the estimated function var(x). Lastly draw the function m(x) and superimpose the bands m(x)+-1.96var(x) -FIRST TIME-

```
plot(Yr, (resmtgr*resmtgr),
     ylab=expression(hat(epsilon)[i]^2),
     xlab=expression(x[i]),
     xlim=c(15,90))
lines(Yr,Vx, col='red')
text(87,1,
     expression(paste(hat(sigma)^2, "(x)")),
     col="red")
```




```
plot(Yr, lpr1$mtgr,type="l",
     ylim = c(min(lpr1$mtgr+1.96*sqrt(Vx))*0.9,
              max(lpr1$mtgr+1.96*sqrt(Vx)*1.1)),
     xlab=expression(x[i]),
     ylab=expression(hat(m)[i]))
lines(Yr,lpr1$mtgr+1.96*sqrt(Vx),lty=3)
lines(Yr,lpr1$mtgr-1.96*sqrt(Vx),lty=3)
```



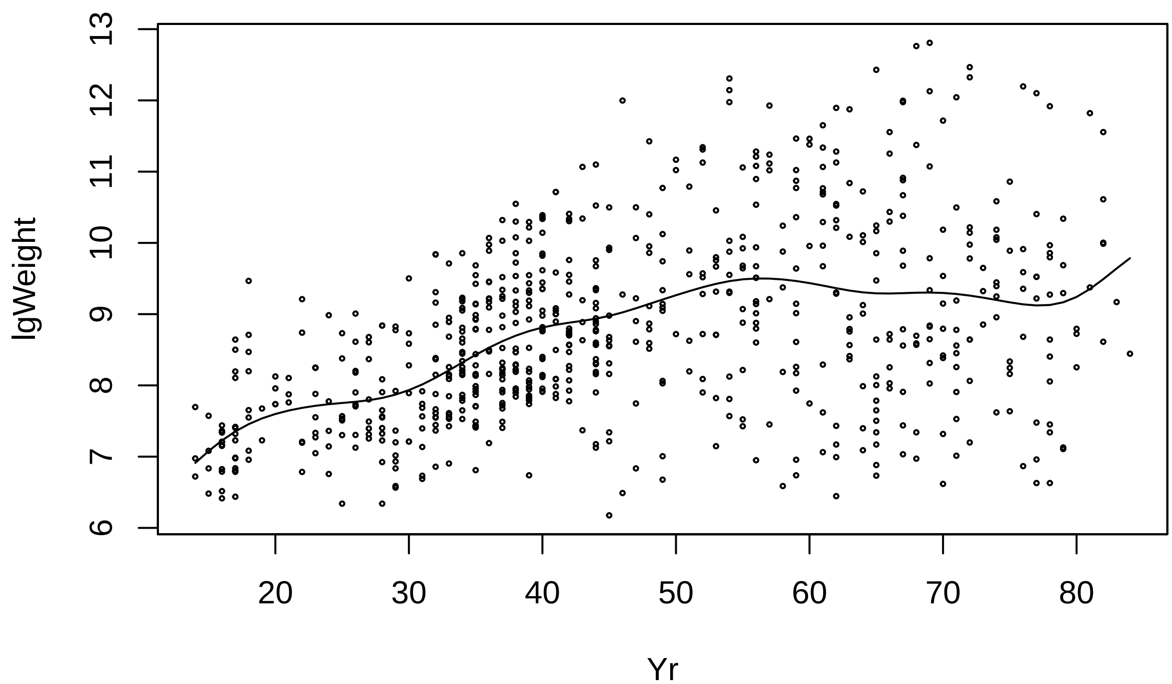
1.Fit a nonparametric regression to data (xi,yi) and save the estimated values m(xi). -SECOND TIME-

```
opt.h.cv<-(h.select(x=Yr, y=lgWeight, method="cv"))

print(dpill(x=Yr,
            y=lgWeight,
            gridsize=101,
            range.x=range(Yr)))
```

```
## [1] 5.433481
```

```
sm.options(eval.points=Yr)
res<-sm.regression(x=Yr,y=lgWeight,h=opt.h.cv)
```

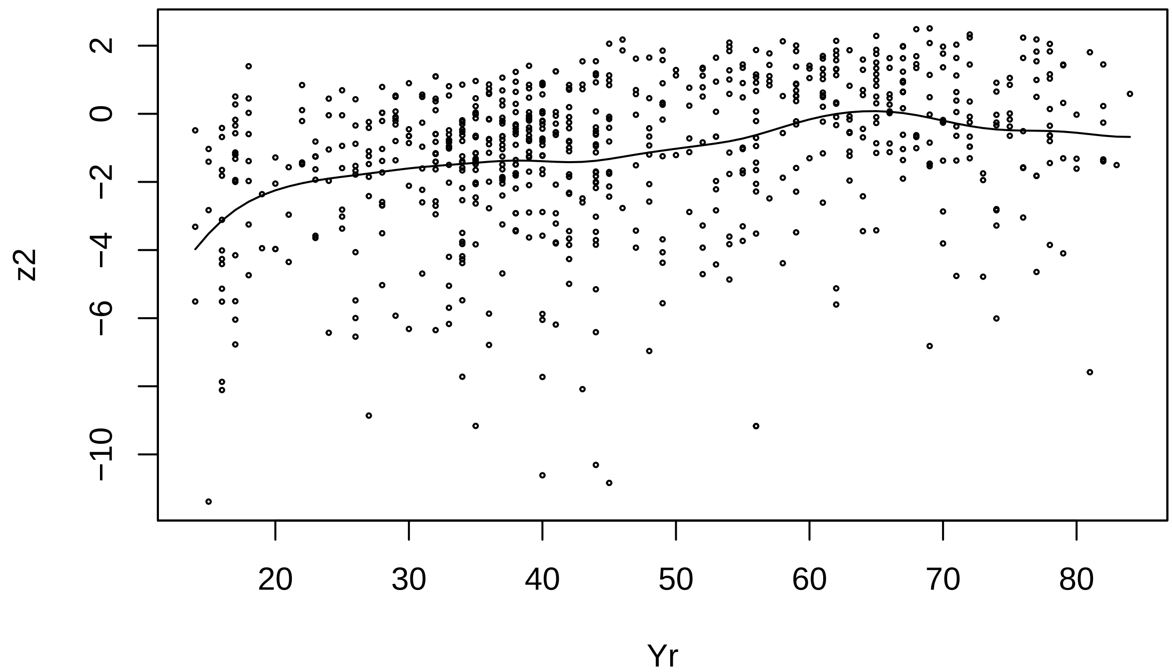


2.Transform the estimated residuals -SECOND TIME-

```
res2<-lgWeight-res$estimate #residus
z2= log(res2^2) #transformació dels residus
```

3.Fit a nonparametric regression to data (xi, zi) and call the estimated function q(x). Observe that q(x) is an estimate of log var(x) -SECOND TIME-

```
sm.options(eval.points=Yr)
res3<-sm.regression(x=Yr,y=z2,h=opt.h.cv)
```

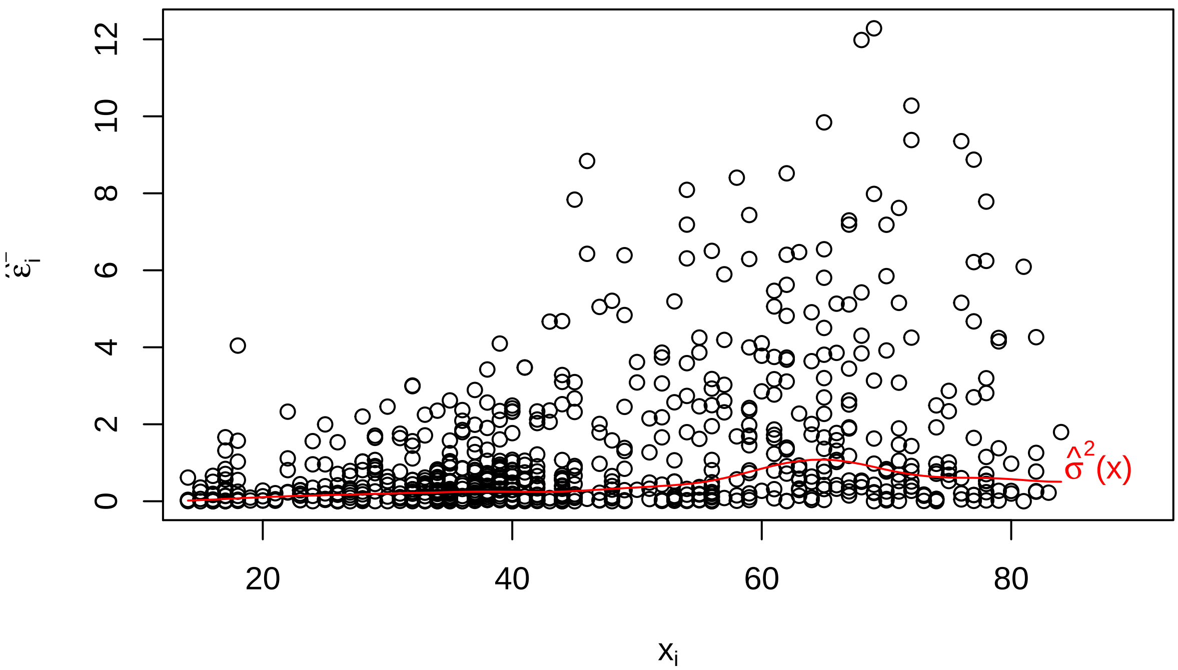


4.Estimate var(x) -SECOND TIME-

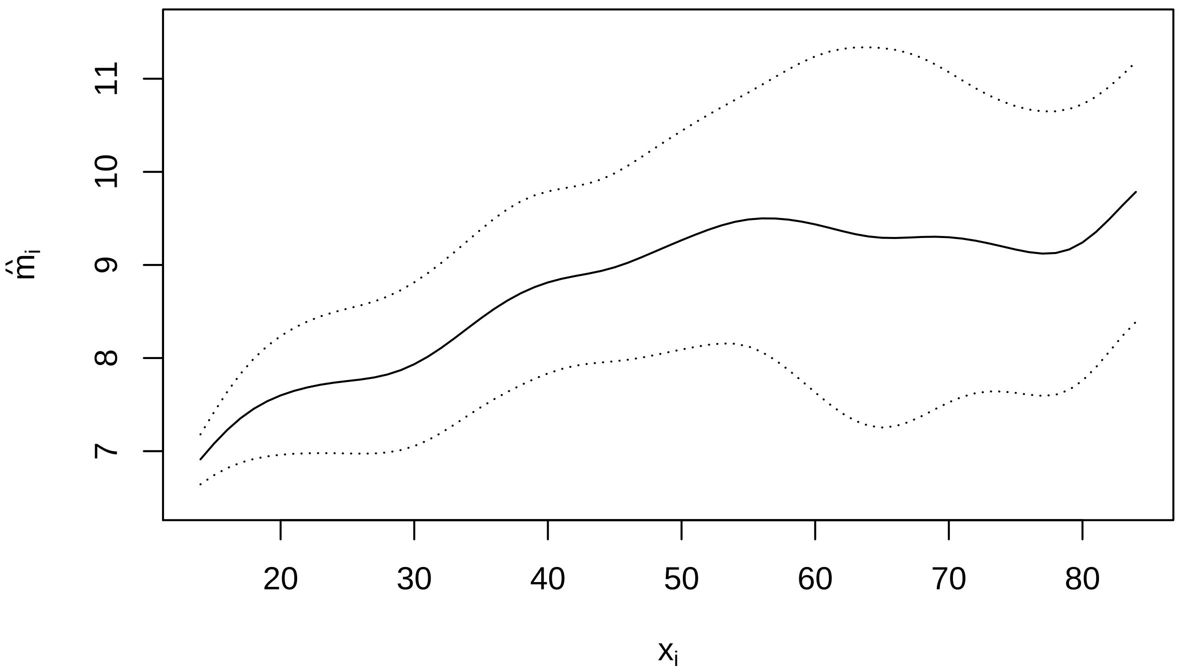
```
Vx2<-exp(res3$estimate)
```

Draw a graphic of resmtgr^2 against xi and superimpose the estimated function var(x). Lastly draw the function m(x) and superimpose the bands m(x)+-1.96var(x) -SECOND TIME-

```
plot(Yr, (res2*res2),
     ylab=expression(hat(epsilon)[i]^2),
     xlab=expression(x[i]),
     xlim=c(15,90))
lines(Yr,Vx2, col='red')
text(87,1,
     expression(paste(hat(sigma)^2, "(x)")),
     col="red")
```




```
plot(Yr, res$estimate,type="l",
     ylim = c(min(res$estimate+1.96*sqrt(Vx2))*0.9,
              max(res$estimate+1.96*sqrt(Vx2)*1.1)),
     xlab=expression(x[i]),
     ylab=expression(hat(m)[i]))
lines(Yr,res$estimate+1.96*sqrt(Vx2),lty=3)
lines(Yr,res$estimate-1.96*sqrt(Vx2),lty=3)
```



7
8
9
10
11
$\hat{m}_i$
20
30
40
50
60
70
80
x_i