Nonparametric Regression

1 Introduction

So far we have assumed that

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon.$$

In other words, $m(x) = \mathbb{E}[Y|X = x] = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$. Now we want to drop the assumption of linearity. We will assume only that $m(x)$ is a smooth function.

Given a sample (X_1, Y_1), ..., (X_n, Y_n), where $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}$, we estimate the regression function

$$m(x) = \mathbb{E}(Y|X = x) \tag{1}$$

without making parametric assumptions (such as linearity) about the regression function $m(x)$. Estimating m is called *nonparametric regression* or *smoothing*. We can write

$$Y = m(X) + \epsilon$$

where $\mathbb{E}(\epsilon) = 0$. This follows since, $\epsilon = Y - m(X)$ and $\mathbb{E}(\epsilon) = \mathbb{E}(\mathbb{E}(\epsilon|X)) = \mathbb{E}(m(X) - m(X)) = 0$

Example 1 *Figure 1 shows data from on bone mineral density. The plots show the relative change in bone density over two consecutive visits, for men and women. The smooth estimates of the regression functions suggest that a growth spurt occurs two years earlier for females. In this example, Y is change in bone mineral density and X is age.*

Example 2 (Multiple nonparametric regression) *Figure 2 shows an analysis of some diabetes data from Efron, Hastie, Johnstone and Tibshirani (2004). The outcome Y is a measure of disease progression after one year. We consider four covariates (ignoring for now, six other variables): age, bmi (body mass index), and two variables representing blood serum measurements. A nonparametric regression model in this case takes the form*

$$Y = m(x_1, x_2, x_3, x_4) + \epsilon. \tag{2}$$

A simpler, but less general model, is the additive model

$$Y = m_1(x_1) + m_2(x_2) + m_3(x_3) + m_4(x_4) + \epsilon. \tag{3}$$

Figure 2 shows the four estimated functions $\widehat{m}_1, \widehat{m}_2, \widehat{m}_3$ and $\widehat{m}_4$.

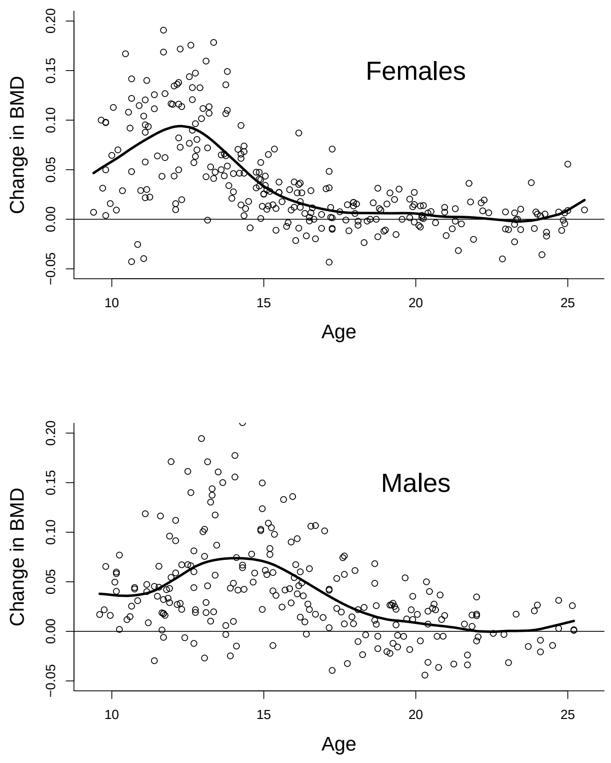

Figure 1: Bone Mineral Density Data

2 The Bias–Variance Tradeoff

The *prediction risk* or *prediction error* is

$$R(m, \widehat{m}) = \mathbb{E}((Y - \widehat{m}(X))^2) \tag{4}$$

where (X, Y) denotes a new observation. It follows that

$$\begin{aligned} R(m, \widehat{m}) &= \tau^2 + \mathbb{E} \int (m(x) - \widehat{m}(x))^2 dP(x) & (5) \\ &= \tau^2 + \int b_n^2(x) dP(x) + \int v_n(x) dP(x) & (6) \end{aligned}$$

where $b_n(x) = \mathbb{E}(\widehat{m}(x)) - m(x)$ is the bias and $v_n(x) = \mathrm{Var}(\widehat{m}(x))$ is the variance. Recall that τ^2 is the *unavoidable error*, due to the fact that there would be some prediction error even if we knew the true regression funcion $m(x)$.

The estimator $\widehat{m}$ typically involves smoothing the data in some way. The main challenge is to determine how much smoothing to do. When the data are oversmoothed, the bias term is large and the variance is small. When the data are undersmoothed the opposite is true.

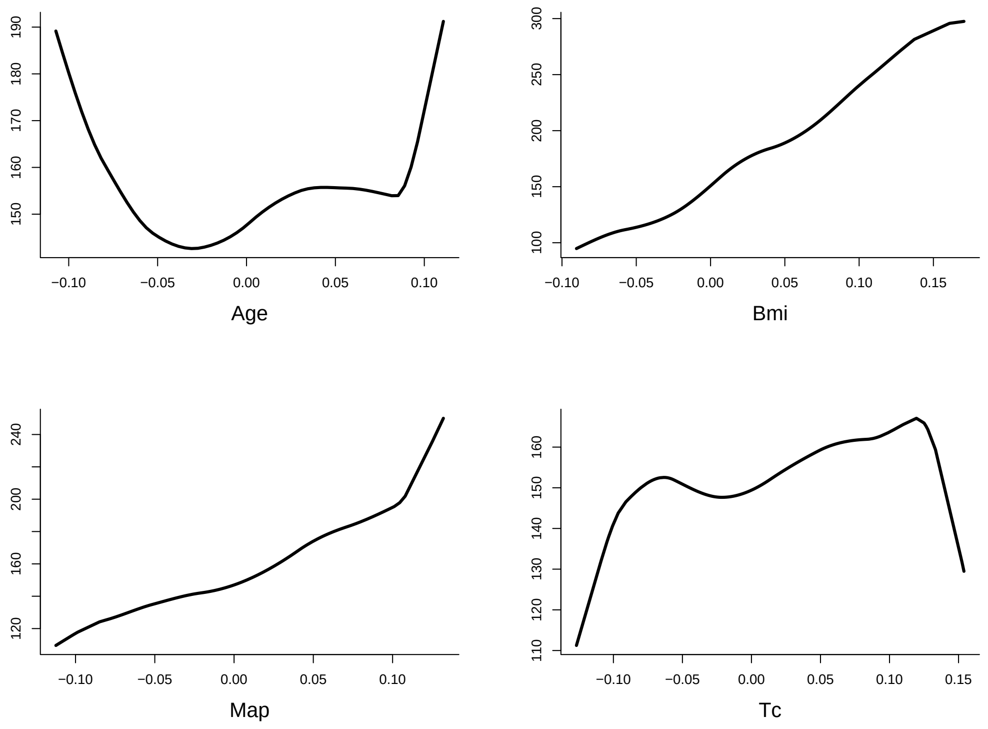

Figure 2: Diabetes Data

This is called the *bias–variance tradeoff*. Minimizing risk corresponds to balancing bias and variance.

3 The Regressogram

We will begin by assuming there is only one covariate X. For simplicity, assume that $0 \leq X \leq 1$. The simplest nonparametric estimator of m is the regressogram. Let k be an integer. Divide $[0, 1]$ into k bins:

$$B_1 = [0, h],\ B_2 = [h, 2h],\ B_3 = [2h, 3h], \ldots.$$

Let n_j denote the number of observations in bin B_j. In other words $n_i = \sum_i I(X_i \in B_j)$ where $I(X_i \in B_j) = 1$ if $X_i \in B_j$ and $I(X_i \in B_j) = 0$ if $X_i \notin B_j$.

We then define $\overline{Y}_j$ to be the average of Y_i's in B_j:

$$\overline{Y}_j = \frac{1}{n_j} \sum_{X_i \in B_j} Y_i.$$

Finally, we define $\widehat{m}(x) = \overline{Y}_j$ for all $x \in B_j$. We can write this as

$$\widehat{m}(x) = \sum_{j=1}^{k} \overline{Y}_j \; I(x \in B_j).$$

Here are some examples.

```
regressogram = function(x,y,left,right,k,plotit,xlab="",ylab="",sub=""){
     ### assumes the data are on the interval [left,right]
     n = length(x)
     B = seq(left,right,length=k+1)
     WhichBin = findInterval(x,B)
     N = tabulate(WhichBin)
     m.hat = rep(0,k)
     for(j in 1:k){
          if(N[j]>0)m.hat[j] = mean(y[WhichBin == j])
          }
     if(plotit==TRUE){
          a = min(c(y,m.hat))
          b = max(c(y,m.hat))
          plot(B,c(m.hat,m.hat[k]),lwd=3,type="s",
            xlab=xlab,ylab=ylab,ylim=c(a,b),col="blue",sub=sub)
          points(x,y)
          }
     return(list(bins=B,m.hat=m.hat))
     }
```

```
pdf("regressogram.pdf")
par(mfrow=c(2,2))
### simulated example
n = 100
x = runif(n)
y = 3*sin(8*x) + rnorm(n,0,.3)
plot(x,y,pch=20)
out = regressogram(x,y,left=0,right=1,k=5,plotit=TRUE)
out = regressogram(x,y,left=0,right=1,k=10,plotit=TRUE)
out = regressogram(x,y,left=0,right=1,k=20,plotit=TRUE)
dev.off()
```

```
### Bone mineral density versus age for men and women
pdf("bmd.pdf")
par(mfrow=c(2,2))
library(ElemStatLearn)
attach(bone)

age.male = age[gender == "male"]
density.male = spnbmd[gender == "male"]
out = regressogram(age.male,density.male,left=9,right=26,k=10,plotit=TRUE,
        xlab="Age",ylab="Density",sub="Male")
out = regressogram(age.male,density.male,left=9,right=26,k=20,plotit=TRUE,
        xlab="Age",ylab="Density",sub="Male")


age.female = age[gender == "female"]
density.female = spnbmd[gender == "female"]
out = regressogram(age.female,density.female,left=9,right=26,k=10,plotit=TRUE,xlab="Age"
out = regressogram(age.female,density.female,left=9,right=26,k=20,plotit=TRUE,xlab="Age"
dev.off()
```

From Figures 3 and 4 you can see two things. First, we need a way to choose k. (The answer will be cross-validation). But also, $\widehat{m}(x)$ is very unsmooth. To get a smoother estimator, we will use *kernel smoothing*.

4 The Kernel Estimator

The idea of *kernel smoothing* is very simple. To estimate $\widehat{m}(x)$ we will take a local average of the $Y_i's$. In other words, we average all Y_i such that $|X_i - x| \le h$ where h is some small number called the bandwidth. We can write this estimator as

$$\widehat{m}(x) = \frac{\sum_{i=1}^n K\left(\frac{x-X_i}{h}\right) Y_i}{\sum_{i=1}^n K\left(\frac{x-X_i}{h}\right)}$$

where

$$K(z) = \begin{cases} 1 & \text{if } |\text{z}| \le 1 \\ 0 & \text{if } |\text{z}| > 1. \end{cases}$$

The function K is called *the boxcar kernel*.

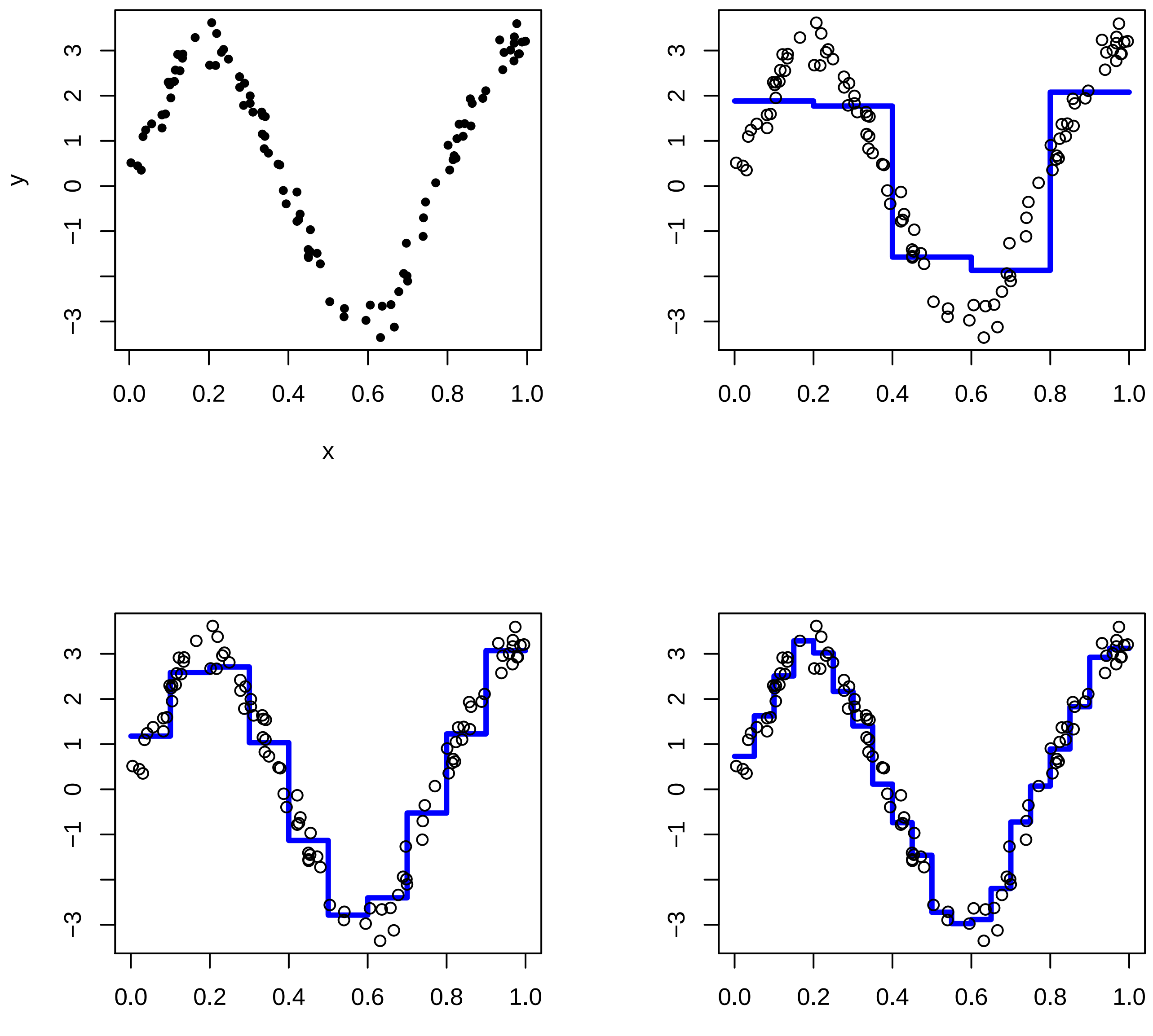

Figure 3: Simulated data. Various choices of k.

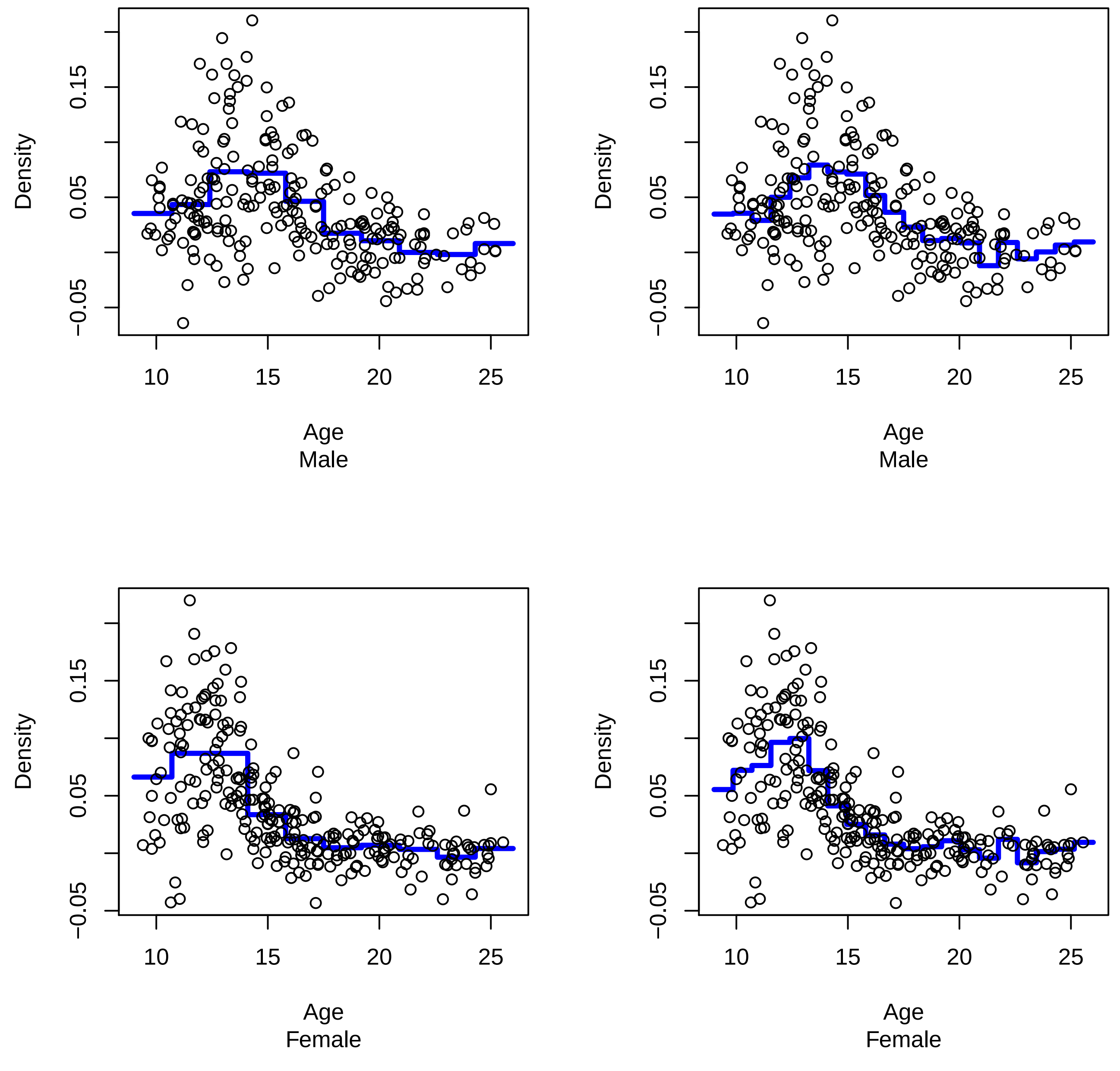

Figure 4: Bone density example. Men and women. Left plots use $k = 10$. Right plots use $k = 20$.

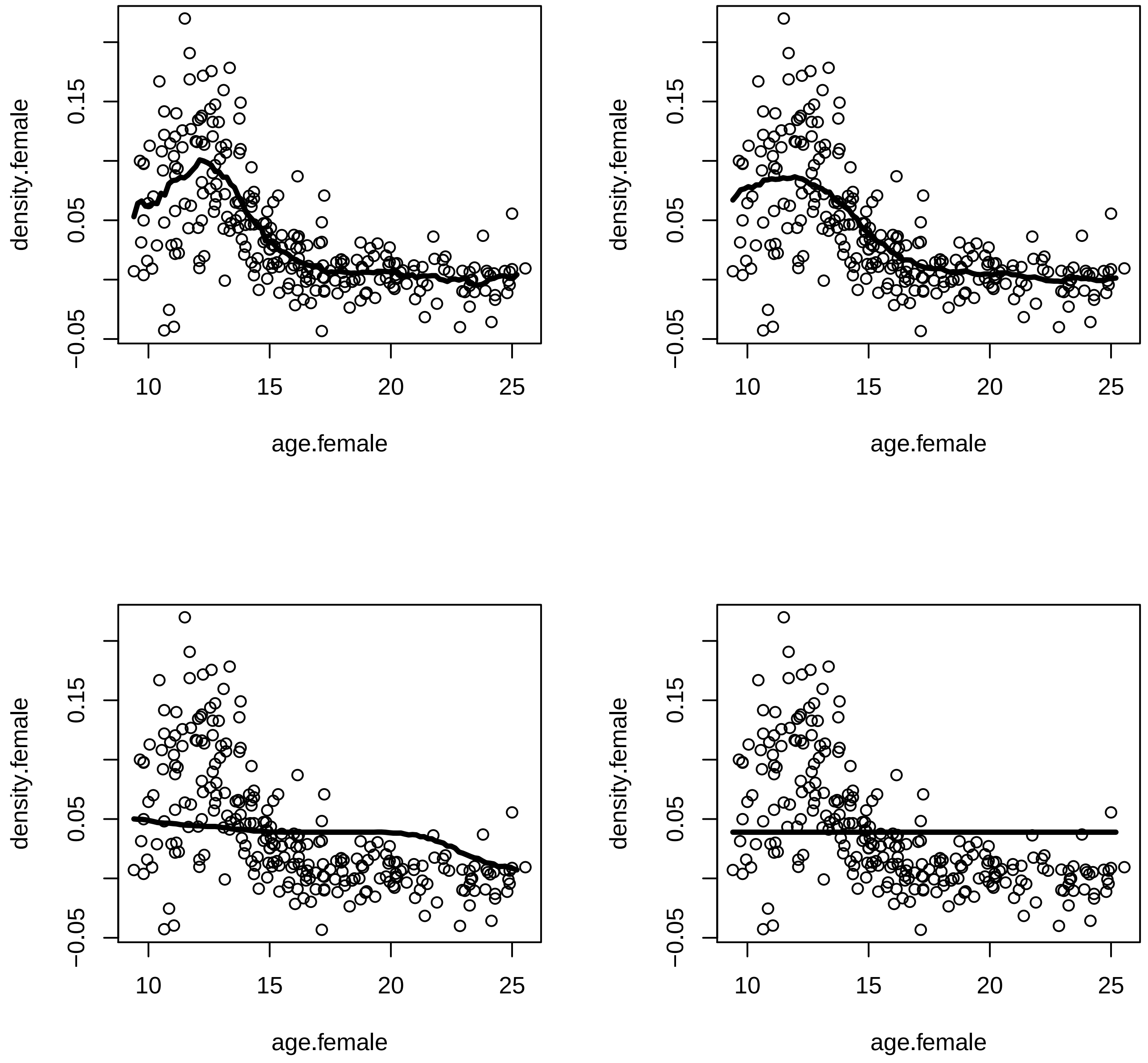

Figure 5: Boxcar kernel estimators for various choices of h.

As you can see in Figure 5, this gives much smoother estimates than the regressogram. But we can improve this even further by replacing K with a smoother function.

This leads us to the following definition. A one-dimensional *smoothing kernel* is any smooth function K such that $K(x) \geq 0$ and

$$\int K(x)\,dx = 1, \quad \int xK(x)dx = 0 \quad \text{and} \quad \sigma_K^2 \equiv \int x^2 K(x)dx > 0. \tag{7}$$

Let $h > 0$ be a positive number, called the *bandwidth*. The *Nadaraya–Watson kernel estimator* is defined by

$$\widehat{m}(x) \equiv \widehat{m}_h(x) = \frac{\sum_{i=1}^n Y_i\, K\left(\frac{x-X_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{x-X_i}{h}\right)} = \sum_{i=1}^n Y_i \ell_i(x) \tag{8}$$

where $\ell_i(x) = K((x-X_i)/h)/\sum_j K((x-X_j)/h)$.

Thus $\widehat{m}(x)$ is a local average of the Y_i's. It can be shown that there is an optimal kernel called the Epanechnikov kernel. But the choice of kernel K is not too important. Estimates obtained by using different kernels are usually numerically very similar. This observation is confirmed by theoretical calculations which show that the risk is very insensitive to the choice of kernel. What does matter much more is the choice of bandwidth h which controls the amount of smoothing. Small bandwidths give very rough estimates while larger bandwidths give smoother estimates. A common choice for the kernel is the normal density:

$$K(z) \propto e^{-z^2/2}.$$

This does **not** mean that we are assuming that the data are Normal. We are just using this as a convenient way to define the smoothing weights.

Figure 6 shows the kernel estimator $\widehat{m}(x)$ for 4 different choices of h using a Gaussian kernel. As you can see, we get much nicer estimates. But we still need a way to choose the smoothing parameter h. Before we discuss that, let's first discuss linear smoothers.

5 The Kernel Estimator is a Linear Smoother

The kernel estimator $\widehat{m}(x)$ is defined at each x. As we did for linear regression, we can also compute the estimator at the original data points X_i. Again, we call $\widehat{Y}_i = \widehat{m}(X_i)$ the fitted values. Note that

$$\widehat{m}_h(X_i) = \frac{\sum_{j=1}^n Y_i\, K\left(\frac{X_i-X_j}{h}\right)}{\sum_{j=1}^n K\left(\frac{X_i-X_j}{h}\right)} = \sum_{j=1}^n Y_i \ell_j(X_i) \tag{9}$$

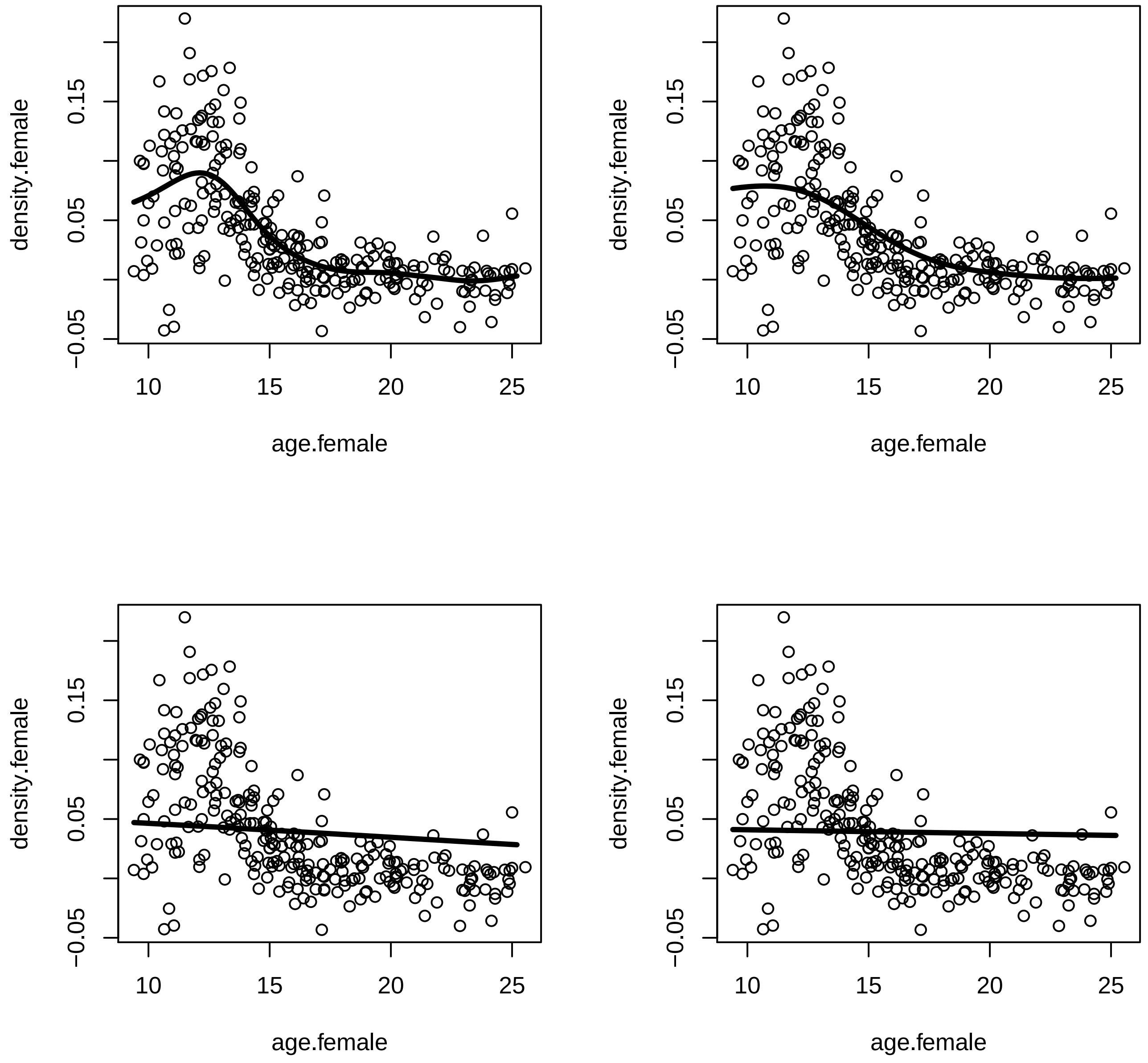

Figure 6: Kernel estimators for various choices of h.

where $\ell_j(x) = K((x - X_j)/h)/\sum_t K((x - X_t)/h)$. Let $\widehat{\mathbf{Y}} = (\widehat{Y}_1, \ldots, \widehat{Y}_n)$, $\mathbf{Y} = (Y_1, \ldots, Y_n)$ and let $\mathbf{L}$ be the $n \times n$ matrix with entries $\mathbf{L}_{ij} = \ell_j(X_i)$. We then see that

$$\widehat{\mathbf{Y}} = \mathbf{L}\mathbf{Y}. \tag{10}$$

This looks a lot like the equation $\widehat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}$ from linear regression.

The matrix $\mathbf{L}$ is called the smoothing matrix. It is like the hat matrix but it is not a projection matrix. We call

$$\nu = \mathrm{tr}(\mathbf{L}) = \sum_i \mathbf{L}_{ii}$$

the *effective degrees of freedom*. As the bandwidth h gets smaller, ν gets larger. In other words, small bandwidths correspond to more complex models.

The equation $\widehat{\mathbf{Y}} = \mathbf{L}\mathbf{Y}$ means that we can write each $\widehat{Y}_i$ as a linear combination of the Y_i's. Because of this, we say that kernel regression is a *linear smoother*. This does **not** mean that $\widehat{m}(x)$ is linear. It just means that each $\widehat{Y}_i$ as a linear combination of the Y_i's.

The are many other linear smoothers besides kernel regression estimators but we will focus on the kernel estimator for now. Remember: a linear smoother does not mean linear regression.

6 Choosing h by Cross-Validation

We will choose h by cross-validation. For simplicity, we focus on leave-one-out cross-validation. Let $\widehat{m}_h^{(-i)}$ be the kernel estimator (with bandwidth h) obtained after leaving out (X_i, Y_i). The cross-validation score is

$$CV(h) = \frac{1}{n}\sum_i (Y_i - \widehat{m}_h^{(-i)}(X_i))^2.$$

It turns out that there is a handy short cut formual for CV which is

$$CV(h) = \frac{1}{n}\sum_i \left(\frac{(Y_i - \widehat{m}_h(X_i))}{1 - \mathbf{L}_{ii}}\right)^2. \tag{11}$$

Our strategy is to fit $\widehat{m}_h$ for many values of h. We then compute $CV(h)$ using (11). Then we find $\widehat{h}$ to minimize $CV(h)$. The bandwidth $\widehat{h}$ is the one we use.

Some people approximate the formula above replacing each $\mathbf{L}_{ii}$ by the average value $(1/n)\sum_i \mathbf{L}_{ii} = (1/n)\mathrm{tr}(\mathbf{L}) = \nu/n$. If we replace $\mathbf{L}_{ii}$ with ν/n we get this formula:

$$GCV(h) = \frac{1}{\left(1 - \frac{\nu}{n}\right)^2}\frac{1}{n}\sum_i (Y_i - \widehat{m}_h(X_i))^2 \tag{12}$$

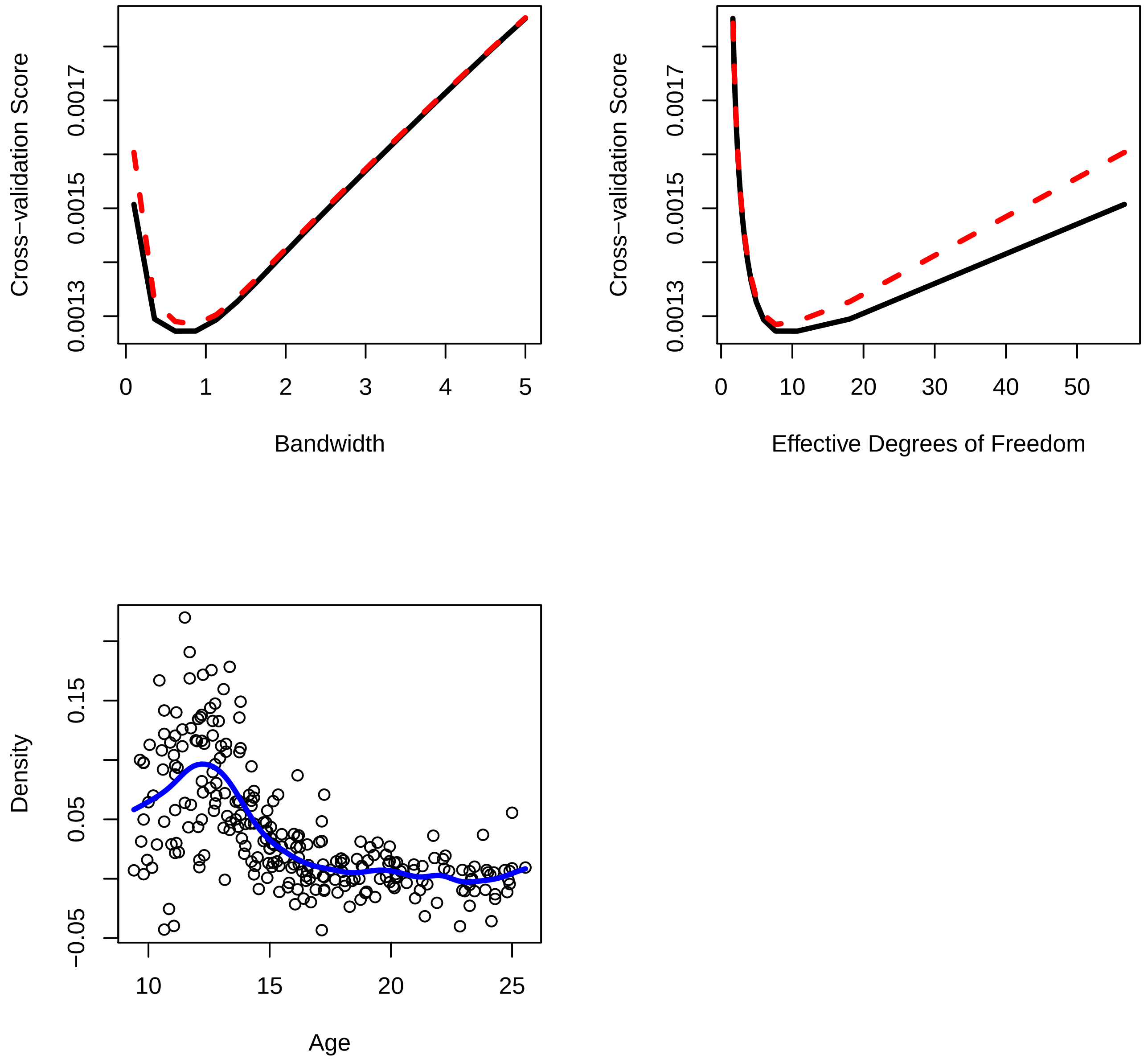

Figure 7: Cross validation (black) and generalized cross validation (red dotted line) versus h and versus effective degrees of freedom. The bottom left plot is the kernel regression estimator using the bandwidth that minimizes the cross-validation score.

which is called *generalized cross validation*.

Figure 7 shows the cross-validation score and the generalized cross validation score. I plotted it versus h and then I plotted it versus the effective degrees of freedom ν. The bottom left plot is the kernel regression estimator using the bandwidth that minimizes the cross-validation score.

Here is the code:

```
kernel = function(x,y,grid,h){
     ### kernel regression estimator at a grid of values
     n = length(x)
     k = length(grid)
     m.hat = rep(0,k)
     for(i in 1:k){
          w = dnorm(grid[i],x,h)
          m.hat[i] = sum(y*w)/sum(w)
          }
     return(m.hat)
     }

kernel.fitted = function(x,y,h){
     ### fitted values and diaginal of smoothing matrix
     n = length(x)
     m.hat = rep(0,n)
     S = rep(0,n)
     for(i in 1:n){
          w = dnorm(x[i],x,h)
          w = w/sum(w)
          m.hat[i] = sum(y*w)
          S[i] = w[i]
          }
     return(list(fitted=m.hat,S=S))
     }


CV = function(x,y,H){
     ### H is a vector of bandwidths
     n = length(x)
     k = length(H)
     cv = rep(0,k)
     nu = rep(0,k)
     gcv = rep(0,k)
     for(i in 1:k){
          tmp = kernel.fitted(x,y,H[i])
          cv[i] = mean(((y - tmp$fitted)/(1-tmp$S))^2)
          nu[i] = sum(tmp$S)
          gcv[i] = mean((y - tmp$fitted)^2)/(1-nu[i]/n)^2
          }
     return(list(cv=cv,gcv=gcv,nu=nu))
     }




pdf("crossval.pdf")
par(mfrow=c(2,2))


bone = read.table("BoneDensity.txt",header=TRUE)
attach(bone)


age.female = age[gender == "female"]
density.female = spnbmd[gender == "female"]


H = seq(.1,5,length=20)
out = CV(age.female,density.female,H)
plot(H,out$cv,type="l",lwd=3,xlab="Bandwidth",ylab="Cross-validation Score")
lines(H,out$gcv,lty=2,col="red",lwd=3)
plot(out$nu,out$cv,type="l",lwd=3,xlab="Effective Degrees of Freedom",ylab="Cross-valida
lines(out$nu,out$gcv,lty=2,col="red",lwd=3)

j = which.min(out$cv)
h = H[j]
grid = seq(min(age.female),max(age.female),length=100)
m.hat = kernel(age.female,density.female,grid,h)
plot(age.female,density.female,xlab="Age",ylab="Density")
lines(grid,m.hat,lwd=3,col="blue")


dev.off()
```

7 Analysis of the Kernel Estimator

Recall that the prediction error of $\widehat{m}$ is

$$\mathbb{E}(Y - \widehat{m}_h(X))^2 = \tau^2 + \int b_n^2(x)p(x)dx + \int v_n(x)p(x)dx$$

where τ^2 is the *unavoidable error*, $b_n(x) = \mathbb{E}(\widehat{m}_h(x)) - m(x)$ is the bias and $v_n(x) = \text{Var}(\widehat{m}_h(x))$ is the variance. The first term is unavoidable. It is the second two terms that we can try to make small. That is, we would like to minimize the integrated mean squared error

$$IMSE = \int b_n^2(x)p(x)dx + \int v_n(x)p(x)dx.$$

It can be shown (under certain conditions) that

$$\int b_n^2(x)p(x)dx \approx C_1 h^4$$

for some constant C_1 and

$$\int v_n(x)p(x)dx \approx \frac{C_2}{nh}.$$

Hence,

$$IMSE = C_1 h^4 + \frac{C_2}{nh}.$$

If we minimize this expression over h we find that the best h is

$$h_n = \left(\frac{C_3}{n}\right)^{1/5}$$

where $C_3 = C_2/(4C_1)$. If we insert this h_n into the IMSE we find that

$$IMSE = \left(\frac{C_4}{n}\right)^{4/5}.$$

What do we learn from this? First, the optimal bandwidth gets smaller as the sample size increases. Second, the IMSE goes to 0 as n gets larger. But it goes to 0 slower than other estimators your are used to. For example, if you estimate the mean μ with the sample mean $\overline{Y}$ then $\mathbb{E}(\overline{Y} - \mu)^2 = \sigma^2/n$. Roughly speaking, the mean squared error of parametric estimators is something like $1/n$ but kernel estimators (and other nonparamettic estimators) behave like $(1/n)^{4/5}$ which goes to 0 more slowly. Slower convergence is the price of being nonparametric.

The formula we derived for h_n is interesting for helping our understanding of the theoretical behavior of $\widehat{m}$. But we can't use that formual in practice because those constants involve quantities that depend on the unknown function m. So we use cross-validation to choose h in practice.

8 Variability Bands

We would like to get some idea of how accurate our estimator is. To estimate the standard error of $\widehat{m}_h(x)$ we are going to use a tool called *the bootstrap*. In 402, you will see that the bootstrap is a very general tool. Here we will only use it to get standard errors for our kernel estimator.

Our data can be written as

$$Z_1, \ldots, Z_n$$

where $Z_i = (X_i, Y_i)$. Here is how the bootstrap works:

1. Compute the estimator $\widehat{m}(x)$.
2. Choose a large number B. Usually, we take $B = 1,000$ or $B = 10,000$.
3. For $j = 1, \ldots, B$ do the following:
    (a) Draw n observations, with replacement, from the original data $\{Z_1, \ldots, Z_n\}$. Call these observations $Z_1^*, \ldots, Z_n^*$. This is called a bootstrap sample.
    (b) Compute a kernel estimator $\widehat{m}_j^*(x)$ from the bootstrap sample. (Note that we compute $\widehat{m}_j^*(x)$ at every x.)
4. At each x, let $\text{se}(x)$ be the standard deviation of the numbers $\widehat{m}_1^*(x), \ldots, \widehat{m}_B^*(x)$.

The resulting function $\text{se}(x)$ is an estimate of the standard deviation of $\widehat{m}(x)$. (I am actually sweeping some technical details under the rug.) We can now plot $\widehat{m}(x) + 2\,\text{se}(x)$ and $\widehat{m}(x) - 2\,\text{se}(x)$. These are called *variability bands*. The are not actually valid confidence bands for some technical reasons that we will not go into here. But they do give us an idea of the variability of our estimator. An example is shown in Figure 8. Here is the code.

```
kernel = function(x,y,grid,h){
     ### kernel regression estimator at a grid of values
     n = length(x)
     k = length(grid)
     m.hat = rep(0,k)
     for(i in 1:k){
          w = dnorm(grid[i],x,h)
          m.hat[i] = sum(y*w)/sum(w)
          }
     return(m.hat)
     }
```

```
boot = function(x,y,grid,h,B){
    ### pointwise standard error for kernel regression using the bootstrap
    k = length(grid)
    n = length(x)
    M = matrix(0,k,B)
    for(j in 1:B){
          I = sample(1:n,size=n,replace=TRUE)
          xx = x[I]
          yy = y[I]
          M[,j] = kernel(xx,yy,grid,h)
          }

     s = sqrt(apply(M,1,var))
     return(s)
     }

bone = read.table("BoneDensity.txt",header=TRUE)
attach(bone)
age.female = age[gender == "female"]
density.female = spnbmd[gender == "female"]
h = .7
grid = seq(min(age.female),max(age.female),length=100)
plot(age.female,density.female)
mhat = kernel(age.female,density.female,grid,h)
lines(grid,mhat,lwd=3)

B = 1000
se = boot(age.female,density.female,grid,h,B)
lines(grid,mhat+2*se,lwd=3,lty=2,col="red")
lines(grid,mhat-2*se,lwd=3,lty=2,col="red")
```

9 Multiple Nonparametric Regression

Suppose now that the covariate is d-dimensional, $X_i = (x_{i1}, \ldots, x_{id})^T$. The regression equation is

$$Y = m(X_1, \ldots, X_d) + \epsilon. \tag{13}$$

A problem that occurs with smoothing methods is the *curse of dimensionality*. Estimation gets harder very quickly as the dimension of the observations increases.

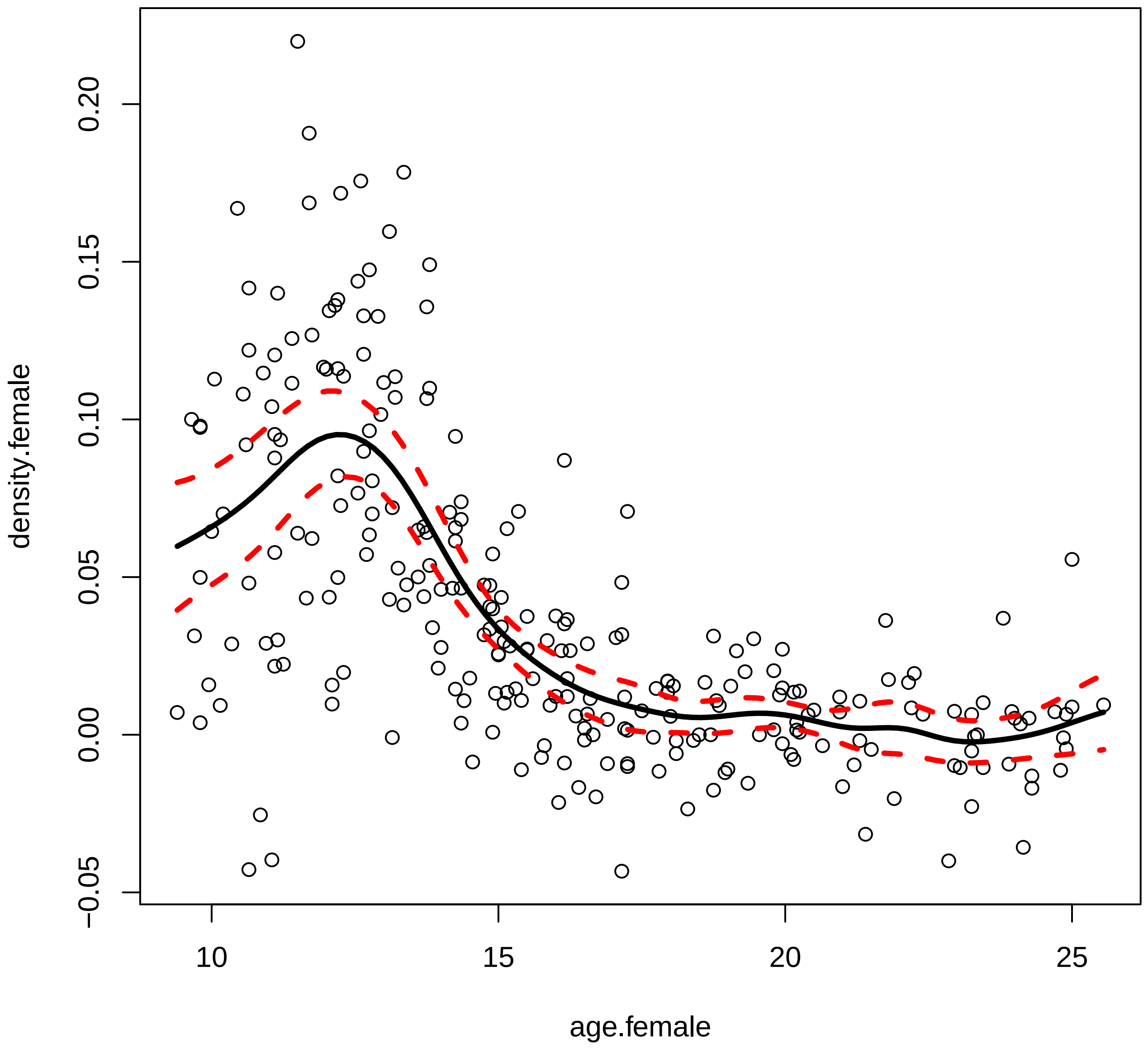

Figure 8: Variability bands for the kernel estimator.

The IMSE of of most nonparametric regression estimators has the form $n^{-4/(4+d)}$. This implies that the sample size needed for a given level of accuracy increases exponentially with d. The reason for this phenomenon is that smoothing involves estimating a function $m(x)$ using data points in a local neighborhood of x. But in a high-dimensional problem, the data are very sparse, so local neighborhoods contain very few points.

9.1 Kernels

The kernel estimator in the multivariate case is defined by

$$\widehat{m}(x) = \frac{\sum_{i=1}^n Y_i \; K(\|x - X_i\|/h)}{\sum_{i=1}^n K(\|x - X_i\|/h)}. \tag{14}$$

We now proceed as in the univariate case. For example, we use cross-validation to estimate h.

9.2 Additive Models

Interpreting and visualizing a high-dimensional fit is difficult. As the number of covariates increases, the computational burden becomes prohibitive. A practical approach is to use an *additive model*. An additive model is a model of the form

$$m(x_1, \ldots, x_d) = \alpha + \sum_{j=1}^d m_j(x_j) \tag{15}$$

where $m_1, \ldots, m_d$ are smooth functions.

The additive model is not well defined as we have stated it. We can add any constant to α and subtract the same constant from one of the m_j's without changing the regression function. This problem can be fixed in a number of ways; the simplest is to set $\widehat{\alpha} = \overline{Y}$ and then regard the m_j's as deviations from $\overline{Y}$. In this case we require that $\sum_{i=1}^n \widehat{m}_j(X_i) = 0$ for each j.

There is a simple algorithm called *backfitting* for turning any one-dimensional regression smoother into a method for fitting additive models. This is essentially a coordinate descent, Gauss-Seidel algorithm.

The Backfitting Algorithm

Initialization: set $\widehat{\alpha} = \overline{Y}$ and set initial guesses for $\widehat{m}_1, \ldots, \widehat{m}_d$. Now iterate the following steps until convergence. For $j = 1, \ldots, d$ do:

- Compute $\widetilde{Y}_i = Y_i - \widehat{\alpha} - \sum_{k \neq j} \widehat{m}_k(X_i)$, $i = 1, \ldots, n$.
- Apply a smoother to $\widetilde{Y}$ on X_j to obtain $\widehat{m}_j$.
- Set $\widehat{m}_j(x) \longleftarrow \widehat{m}_j(x) - n^{-1} \sum_{i=1}^{n} \widehat{m}_j(X_i)$.
- end do.

In principle, we could use a different bandwidth h_j for each $\widehat{m}_j$. This is a lot of work. Instead, it is common to use the same bandwidth h for each function.

9.3 Example

Here we consider predicting the price of cars based on some covariates.

```
library(mgcv)
D = read.table("CarData.txt",header=TRUE)
##data from Consumer Reports (1990)
attach(D)
names(D)
[1] "Price"   "Mileage" "Weight"  "Disp"    "HP"
pairs(D)
out = gam(Price ~ s(Mileage) + s(Weight) + s(Disp) + s(HP))
summary(out)

#Family: gaussian
#Link function: identity
#
#Formula:
#Price ~ s(Mileage) + s(Weight) + s(Disp) + s(HP)
#
#Parametric coefficients:
#            Estimate Std. Error t value Pr(>|t|)
#(Intercept)  12615.7      307.3   41.06   <2e-16 ***
#---
#Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
#
#Approximate significance of smooth terms:
#              edf Ref.df      F p-value
#s(Mileage) 4.328  5.314  1.897 0.12416
#s(Weight)  1.000  1.000  7.857 0.00723 **
#s(Disp)    1.000  1.000 11.354 0.00146 **
#s(HP)      4.698  5.685  3.374 0.00943 **
#---
#Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
#
#R-sq.(adj) =   0.66   Deviance explained = 72.4%
#GCV = 7.0853e+06  Scale est. = 5.6652e+06  n = 60
#

plot(out,lwd=3)
r = resid(out)
plot(Mileage,r);abline(h=0)
plot(Weight,r);abline(h=0)
plot(Disp,r);abline(h=0)
plot(HP,r);abline(h=0)
```

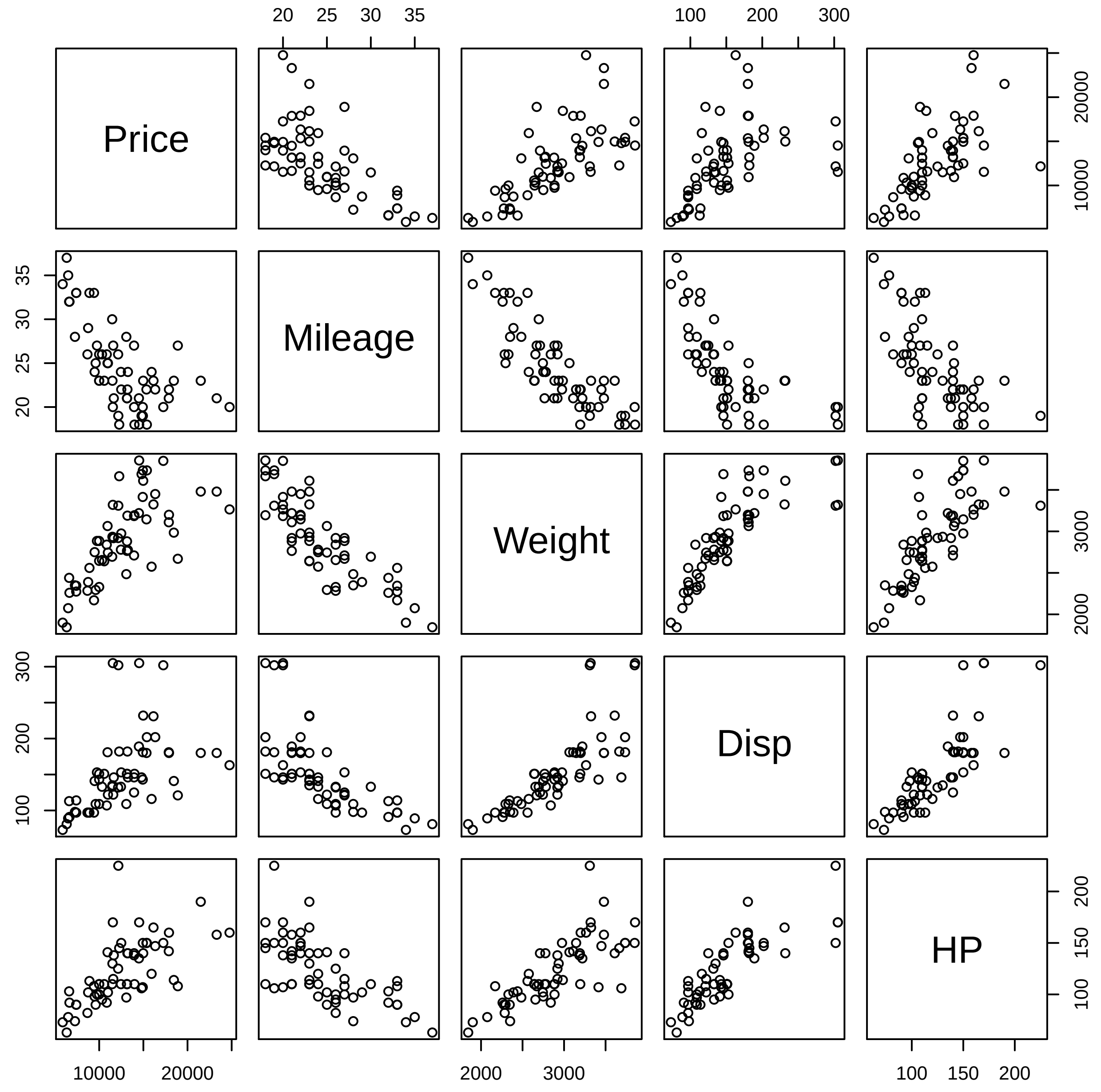

Figure 9: The car price data.

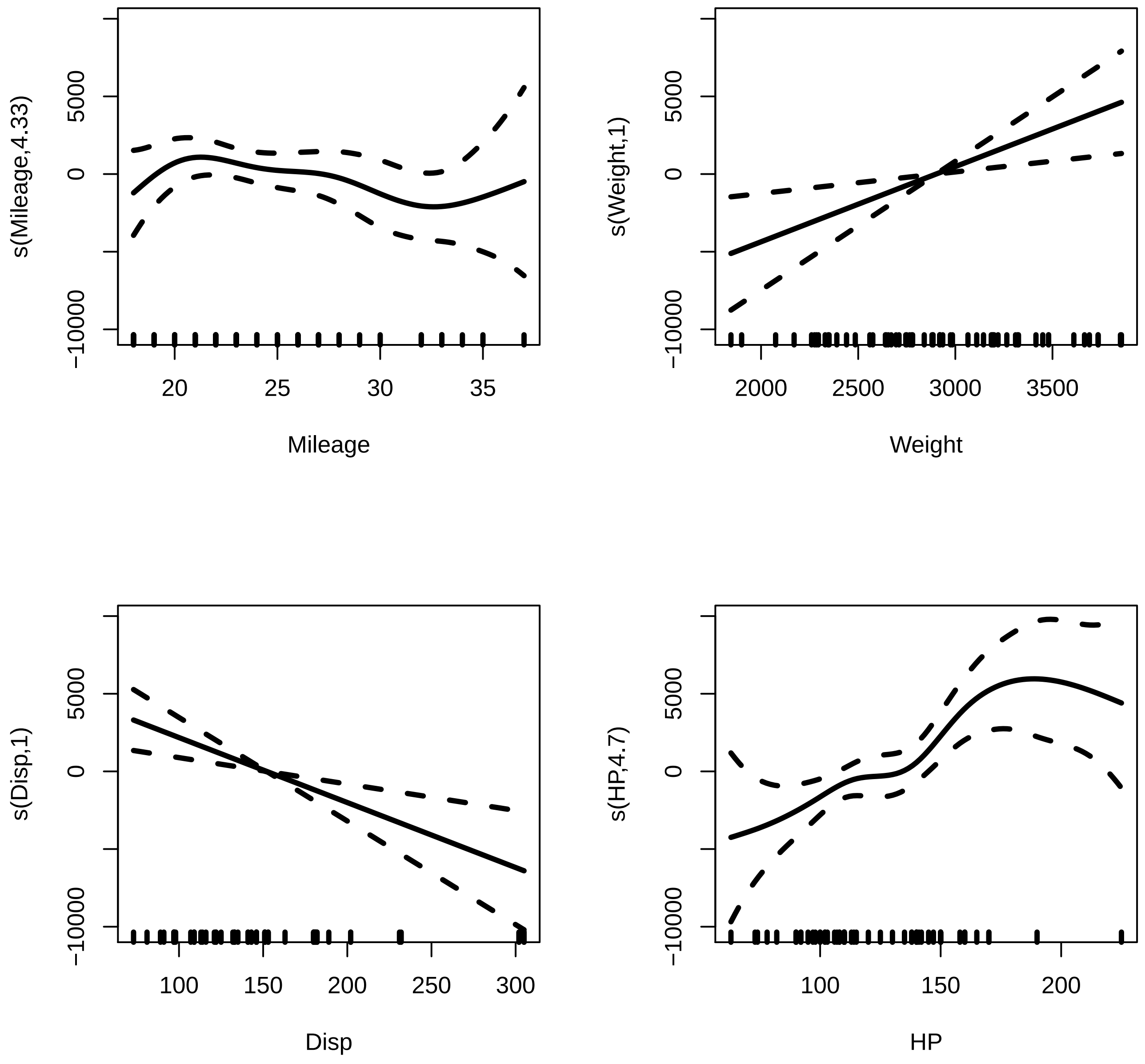

Figure 10: The car price data: estimated functions.

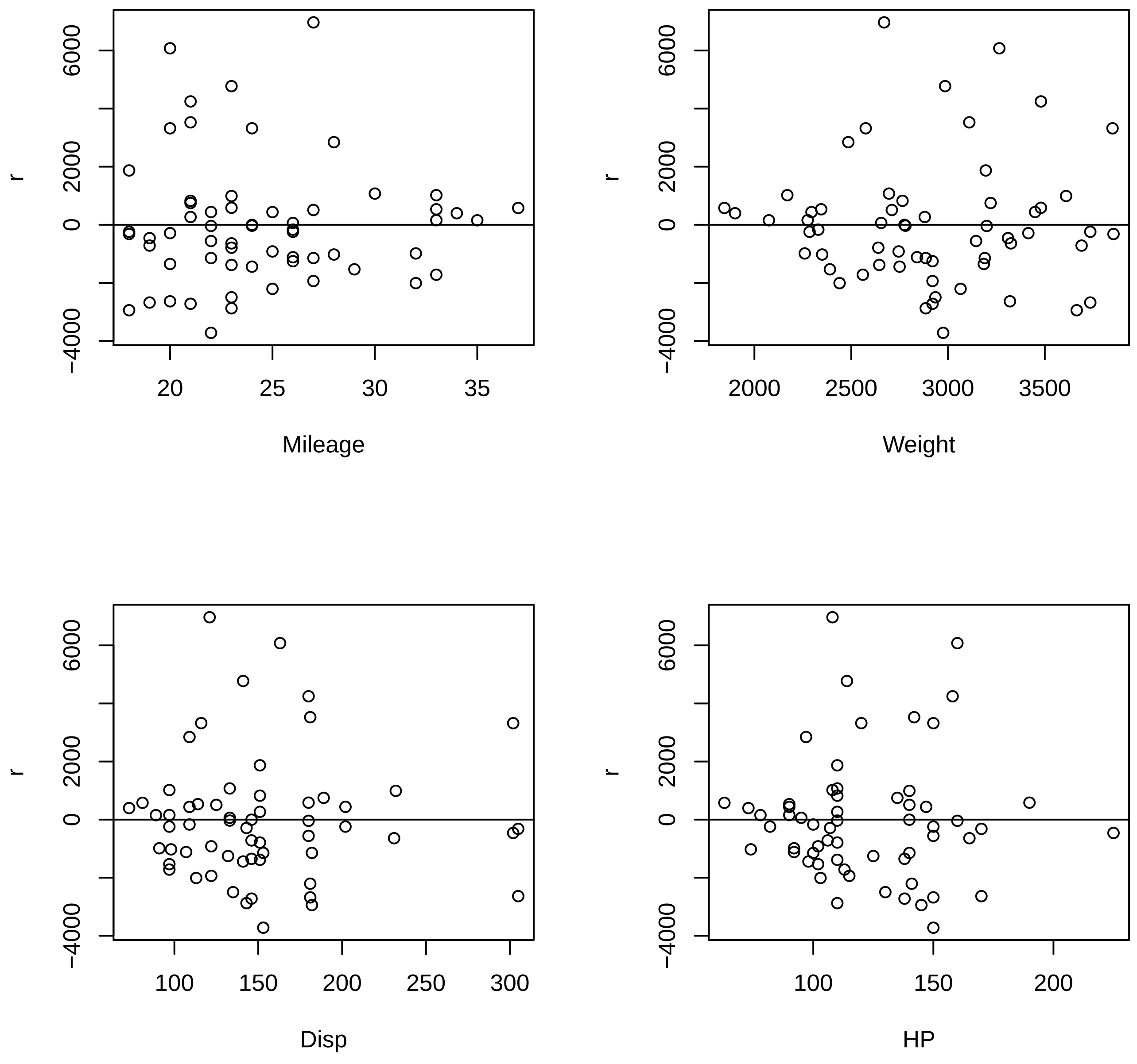

Figure 11: The car price data: residuals