5.1 Local smoothing

Local smoothing

In this section we discuss a class of regression techniques that achieve flexibility in estimating the regression function $f(X)$ over the domain $\boldsymbol{R}^p$ by fitting a different but simple model separately at each point x_0.

This is done by using only those observations close to the target point x_0 to fit the simple model, and in such a way that the resulting estimated function $f(X)$ is smooth in $\boldsymbol{R}^p$. This localisation is achieved via a weighting function or kernel, which assigns a weight to x_i based on its distance from x_0.

So far, we have only discussed linear models, for which the mean response is a linear function of the predictors.

Our next aim is to model data as in Figure 1. A standard model assumption in nonlinear regression is

$$y_i = f(x_i) + \epsilon_i$$

where y_i is the response, x_i is a vector of predictors and the ϵ_i are zero mean uncorrelated errors with a common variance σ^2.

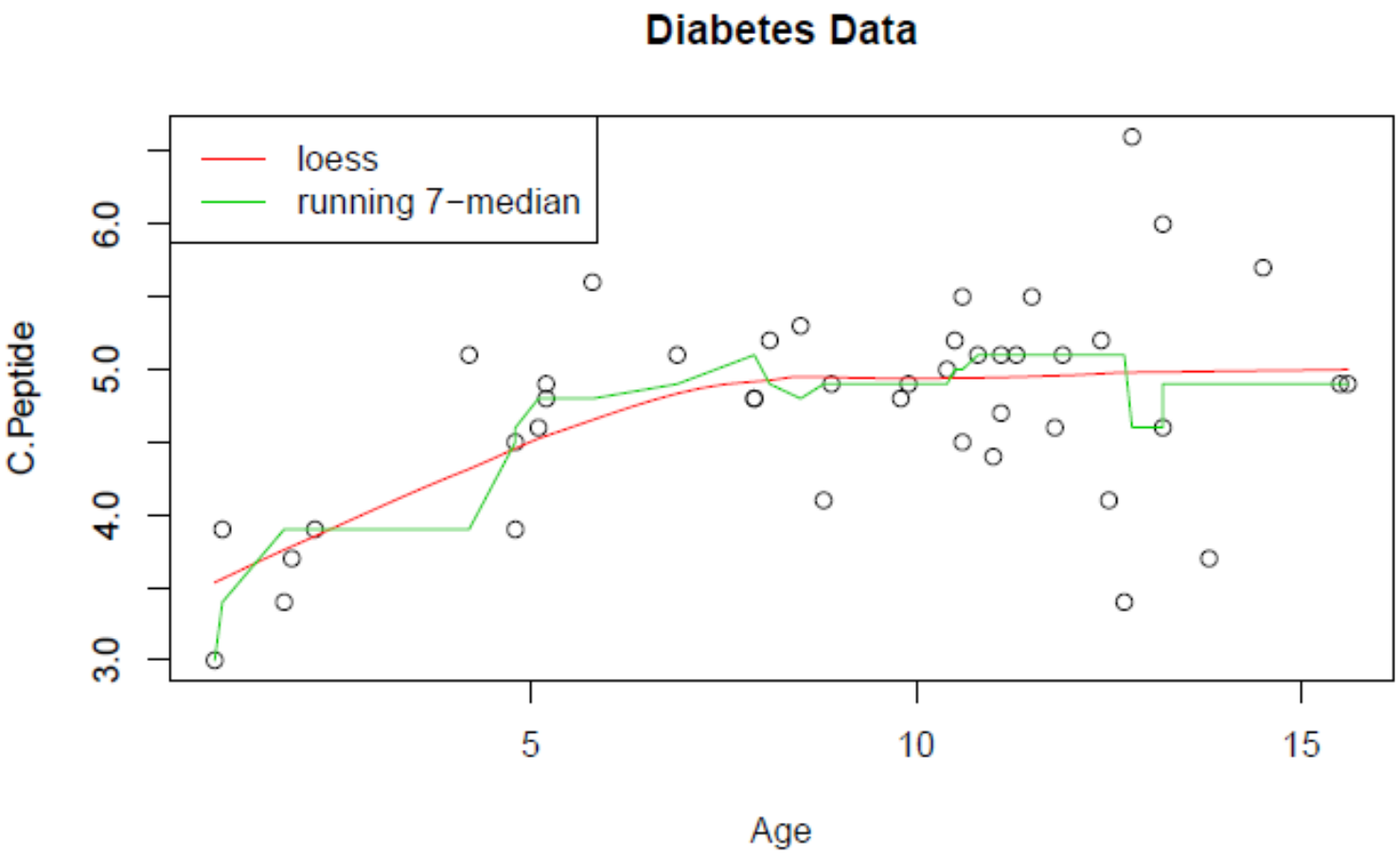


Figure 5.1.1: Data from a study (Sockett et al. 1987) of the factors affecting patterns of insulin-dependent diabetes mellitus in children.

The interest is in the "underlying trend" in a scatterplot, where scatterplot points are simply treated as a collection of points on a plane, without much regard to an underlying probabilistic model.

A scatterplot smoother is a function of the data which defines an estimator $\hat{f}(x)$ of $f(x)$ over the range of the predictor values.

Smoothers are often defined by specifying an estimate of f for the observed x_i and then using interpolation.

In what follows, assume the x_i's are distinct, and that the predictors are ordered, $x_1 < \ldots < x_n$.

Nearest neighbour smoothers

The symmetric k nearest neighbourhood of x_i consists of x_i and the k nearest predictor values on either side (take as many values as you can if there aren't k on either side):

$$\{x_j : j \in N_k^S(x_i)\} \text{ where } N_k^S(x_i) = \{\max(1, i-k), ..., \min(n, i+k)\}.$$

Note that the superscript S indicates that it is *symmetric.*

The running median smoother

The running median smoother at x_i computes the median $\hat{f}(x_i)$ of the responses of the k nearest predictors $x \in N_k^S(x_k)$, and interpolates these values.

Note that k is a *smoothing parameter*. **If k is too large, we won't capture any sharp changes in f ($\hat{f}$ will have an appreciable bias). If k is too small, $\hat{f}$ will be highly variable**.

The running median smoother is easily computed, but has several disadvantages:

- "Boundary bias": there may be **substantial bias** in $\hat{f}$ near the endpoints x_1 or x_n since we may average over an asymmetric neighbourhood there, see Figure 5.1.2.

- The estimated f is **not differentiable** for running medians/means. However, signals of measurements of continuous quantities are typically required to be smooth.

The running line smoother

The running line smoother achieves more smoothness and better boundary behaviour. It fits a linear regression model locally to the data in the symmetric nearest neighbourhood about x_i, and interpolates the fitted values $\hat{f}(x_i)$.

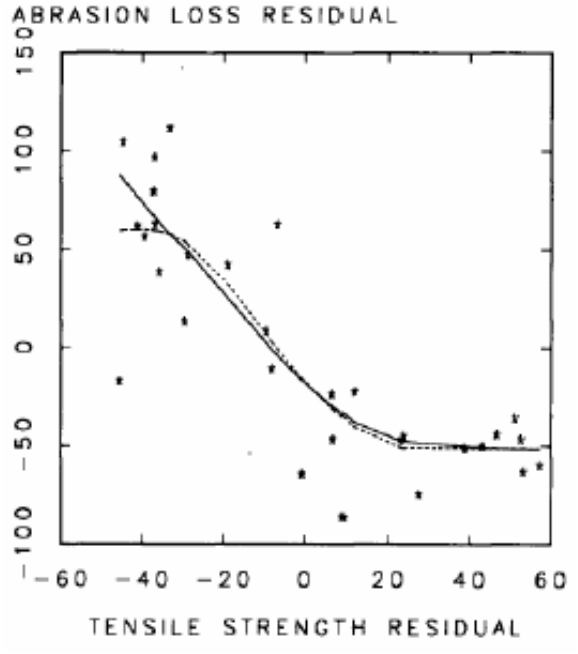


Figure 5.1.2: Boundary bias, illustrated by signal (full line) and nonparametric estimate (dashed line) for a running line fit. Source: Cleveland (1979).

The slope in the local regression captures the trend near the boundary, and thus bias is reduced. However, the fitted $\hat{f}(x)$ is not differentiable, since it is a linear interpolation of the fitted values $\hat{f}(x_i)$.

Loess smoothers

Suppose that the fitted value $\hat{f}(x_0)$ is to be computed at x_0, where data are given at (x_i, y_i), $i = 1, \ldots, n$. Loess (**lo**cally **we**ighted **s**catterplot **s**moothing) smoothing considers the weighted least squares criterion

$$\sum_i w_i(x_0)(y_i - \beta_0(x_0) - \beta_1(x_0)x_i)^2$$

at the point x_0. The weights w_i are smaller for x_i which are further away from x_0.

Minimising the above criterion fits a line where points with x_i close to x_0 get more weight in the fit. The line is discarded, however, and only the fitted value

$$\hat{f}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0$$

is kept.

Below the Loess algorithm of Cleveland (1979) is outlined:

1. Pick a point x_0 (not necessarily one of the x_i). Find the k nearest x_i values to x_0, set of indices $N_k(x_0)$.

2. Calculate

$$\Delta(x_0) = \max_{i \in N_k(x_0)} |x_0 - x_i|.$$

3. Assign weights to each point as

$$K\left(\frac{|x_0 - x_i|}{\Delta(x_0)}\right)$$

   where

$$K(u) = \left\{ \begin{array}{ll} (1-u^3)^3 & \text{for } 0 \leq u \leq 1 \\ 0 & \text{otherwise.} \end{array} \right\}.$$

4. Calculate the weighted least squares line within the neighbourhood defined by $N_k(x_0)$ and take the fitted value at x_0 as $\hat{f}(x_0)$.

5. Repeat for every desired value of x_0.

Remarks:

1. As $K(u) = 0$ for $|u| \geq 1$,

$$K\left(\frac{|x_0 - x_i|}{\Delta(x_0)}\right) = 0$$

   if $|x_0 - x_i| \geq \Delta(x_0)$. So only x_i closer than $\Delta(x_0)$ contribute to the fit.

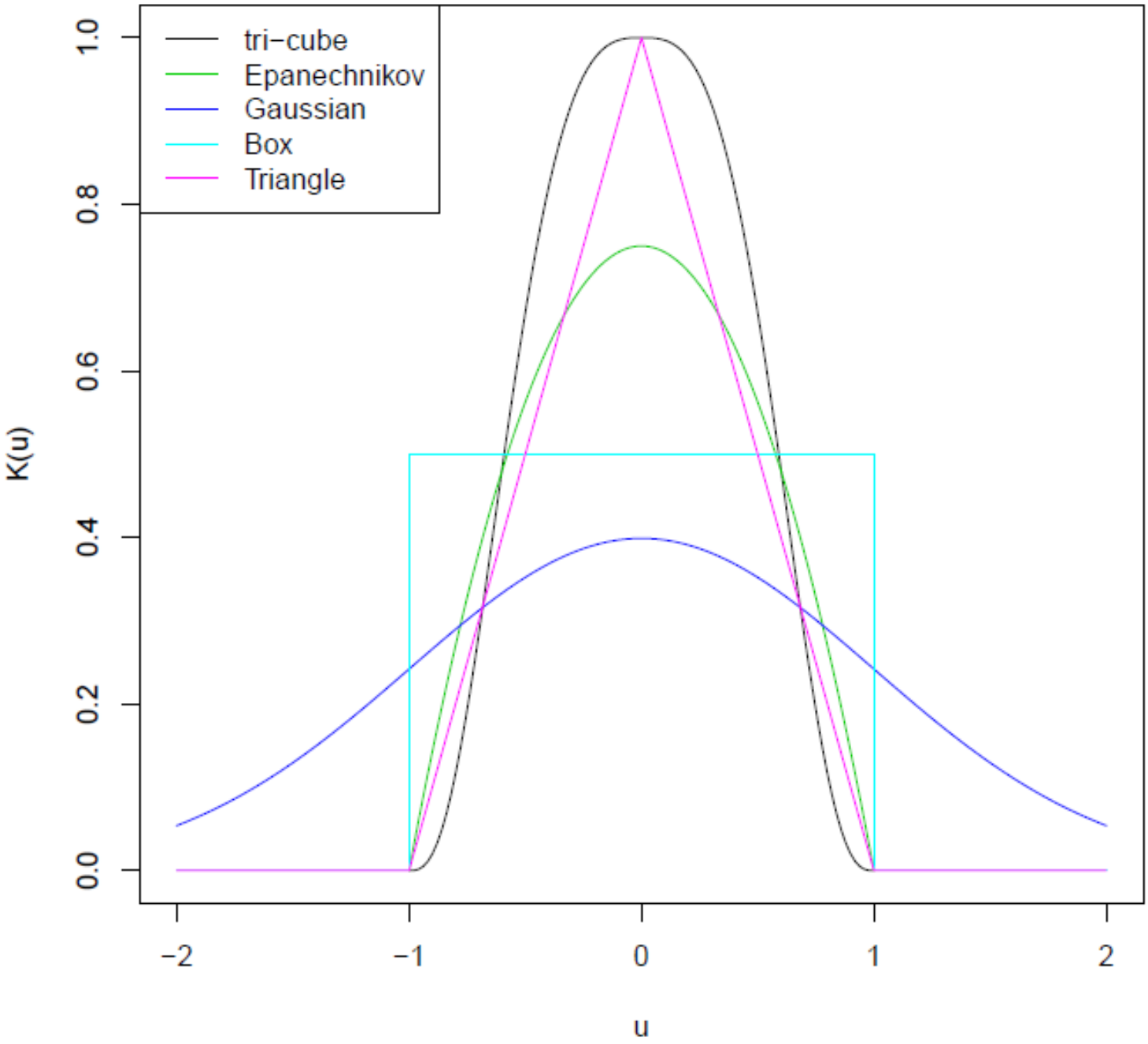
2. $K(u)$ has its maximum at $u = 0$ and decreases as $|u|$ increases. So the weight is given to x_i decreases as x_i gets further from x_0.

3. There are variations on the basic loess algorithm which attempts to down weight the influence of outliers (points which don't fit the pattern of the rest of the data).

Different kinds of weighting functions

Below, we visualise some weighting functions that can be used in the above algorithm. Note that they all integrate to 1. They mathematical definition is provided in the next slide.

```
xx1=seq(from = -1,to = 1,length.out = 100)
yy=(1-abs(xx1)^3)^3
plot(xx1,yy,xlim = c(-2,2),type='l',xlab='u',ylab='K(u)')
lines(c(-2,-1),c(0,0)); lines(c(1,2),c(0,0))
yy=3/4*(1-xx1^2)
lines(xx1,yy,col=3)
xx2=seq(from = -2,to = 2,length.out = 200)
yy=dnorm(xx2)
lines(xx2,yy,col=4)
lines(c(-2,-1,-1,1,1,2),c(0,0,1/2,1/2,0,0),col=5)
lines(c(-2,-1,0,1,2),c(0,0,1,0,0),col=6)
legend("topleft", legend=c("tri-cube", "Epanechnikov", "Gaussian", "Box",
                           "Triangle"),lty=1, col=c(1,3,4,5,6))
```



Example: Scatterplot smoothers

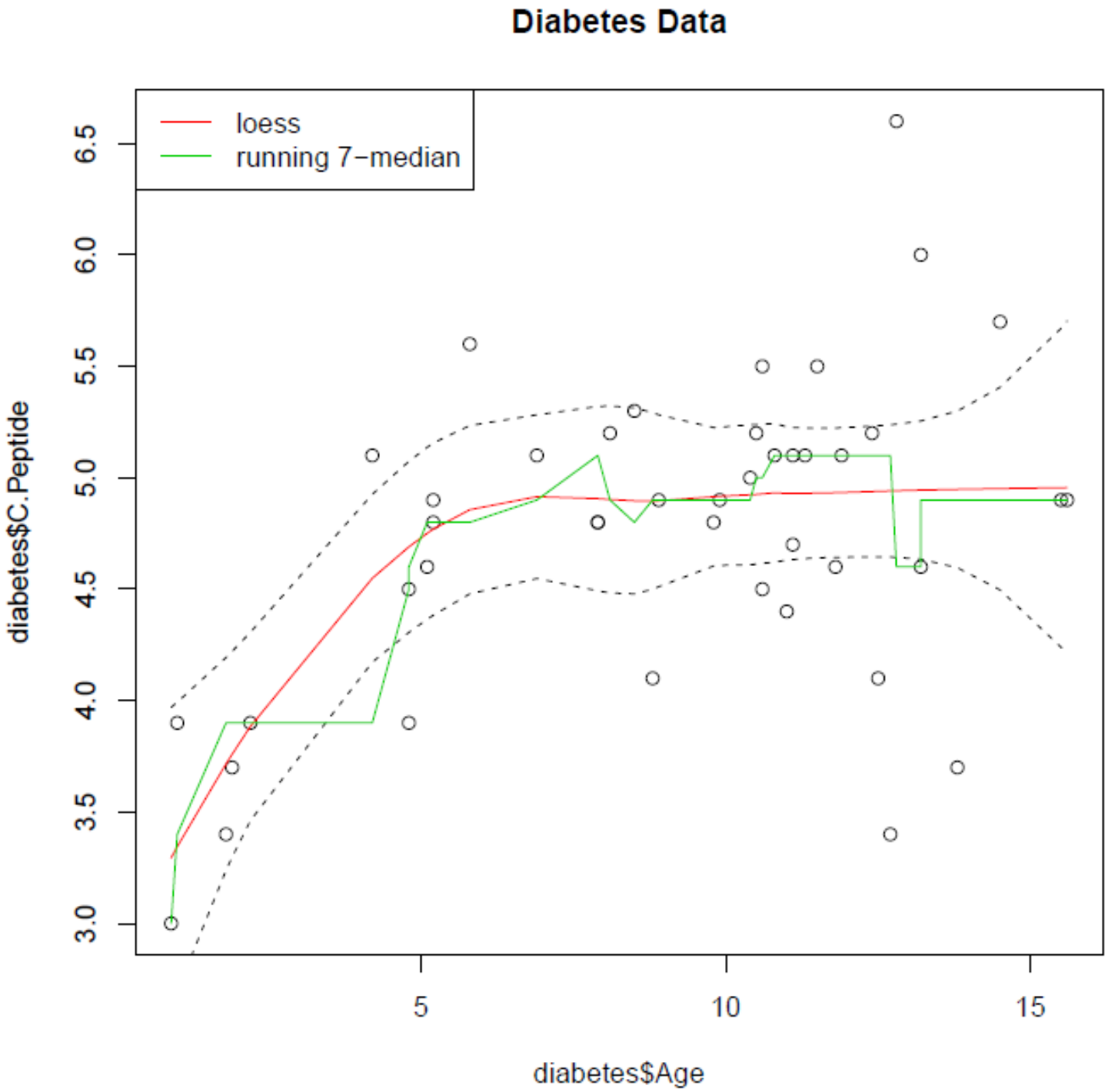
Recall the diabetes data set from the start of this section. This is data from a study of the factors affecting patterns of insulin-dependent diabetes mellitus in children, Sockett et al. 1987.

```
diabetes <- read.table("/course/data/diabetes.dat", header=TRUE, quote="\"")
diabetes <- diabetes[order(diabetes$Age),]
head(diabetes)
##     Age  Base.Deficit C.Peptide
## 15  0.9         -11.6       3.0
## 24  1.0          -8.2       3.9
## 6   1.8         -19.2       3.4
## 10  1.9         -25.0       3.7
## 11  2.2          -3.1       3.9
## 36  4.2         -17.0       5.1
```

The scatterplot smoothers for the diabetes data set were obtained as follows:

```
diabetes <- read.table("/course/data/diabetes.dat", header=TRUE, quote="\"")
diabetes <- diabetes[order(diabetes$Age),]
plot(diabetes$Age,diabetes$C.Peptide,main="Diabetes Data")
fit=loess(diabetes$C.Peptide~diabetes$Age,diabetes)
pred=predict(fit,diabetes$Age,se=T)

# lines(loess.smooth(diabetes$Age,diabetes$C.Peptide),col=2)
lines(diabetes$Age,predict(fit),col=2)
lines(diabetes$Age,pred$fit+2*pred$se.fit,lty=2)
lines(diabetes$Age,pred$fit-2*pred$se.fit,lty=2)
lines(diabetes$Age,runmed(x = diabetes$C.Peptide,k = 7),col=3)
legend("topleft", legend = c("loess", "running 7-median"),lty=c(1,1), col = 2:3)
##     Age  Base.Deficit C.Peptide
## 15  0.9         -11.6       3.0
## 24  1.0          -8.2       3.9
## 6   1.8         -19.2       3.4
## 10  1.9         -25.0       3.7
## 11  2.2          -3.1       3.9
## 36  4.2         -17.0       5.1
```



The function $loess.smooth()$ returns a list with two components, $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{x}$ contains a set of predictor values and $\mathbf{y}$ is the vector of smoothed values given by $loess.smooth()$. The default specification of k in $loess.smooth()$ is a "span" of $2/3$ of all data values.

Bias/variance tradeoff

There is a natural bias/variance tradeoff in loess smooth as we change the width of the averaging window, which is most explicit for local averages:

- If the **window is narrow**, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0, and its **variance will be relatively large** -- close to that of an individual y_i. The **bias will tend to be**

small, again because each of the $\boldsymbol{E}[y_i] = f(x_i)$ should be close to $f(x_0)$.

- If the **window is wide**, the **variance of $\hat{f}(x_0)$ will be small** relative to the variance of any y_i, because of the effects of averaging. The **bias will be higher**, because we are now using observations x_i further from x_0 , and there is no guarantee that $f(x_i)$ will be close to $f(x_0)$.

Higher Order Loess smoothers

Why only consider loess with local linear fits? We can fit local polynomials of any degree d, by minimizing

$$\sum_{i=1}^{n} K_h(x_0, x_i) \left[y_i - \beta_0(x_0) - \sum_{j=1}^{d} \beta_j(x_0) x_i^j \right]^2 .$$

The solution

$$\hat{f}(x0) = \hat{\beta}_0(x_0) + \sum_{j=1}^{d} \hat{\beta}_j(x_0) x_0^j$$

is **less prone to "trimming the hills and filling the valleys,"** (local linear fits tend to be biased in regions of curvature of the true function) and **boundary bias will also be reduced**. The price to be paid for this bias reduction is, of course, an **increased variance**.

Kernel smoothers

Kernel smoothing calculates $\hat{f}(x_0)$ by a weighted average of the responses y_i:

$$\hat{f}(x_0) = \sum_i w_i(x_0) y_i$$

The weight $w_i(x_0)$ given to y_i is

$$w_i(x_0) = \frac{K((x_i - x_0)/h)}{\sum_j K((x_j - x_0)/h)}$$

where $K(u)$ is a symmetric function decreasing in $|u|$ called the kernel function, and $h > 0$ is a smoothing parameter (called the bandwidth). Note that $\sum_i w_i(x_0) = 1$. The estimate $\hat{f}(x_0)$ is known as the *Nadaraya-Watson kernel estimator*.

The kernel functions can be selected from the following choices:

Epanechikov kernel:

$$K(t) = \left\{ \begin{array}{ll} \frac{3}{4}(1-t^2) & \text{for } |t| \le 1 \\ 0 & \text{otherwise} \end{array} \right\}.$$

Box kernel:

$$K(t) = \left\{ \begin{array}{ll} 1 & |t| \le 1 \\ 0 & \text{otherwise} \end{array} \right\}.$$

Triangular kernel:

$$K(t) = \left\{ \begin{array}{ll} 1-|t| & |t| \le 1 \\ 0 & \text{otherwise} \end{array} \right\}.$$

Tri-cube kernel:

$$K(t) = \left\{ \begin{array}{ll} (1-|t|^3)^3 & |t| \le 1 \\ 0 & \text{otherwise} \end{array} \right\}.$$

Gaussian kernel: $K(t)$ is the standard normal density function.

The fitted functions are continuous / smooth whenever $K(u)$ is continuous/smooth. As the window is moving through the interval, data points enter the neighbourhood initially with weight zero, and then their contributions slowly increases.

The choice of kernel is less crucial to the predictive performance than the choice of smoothing

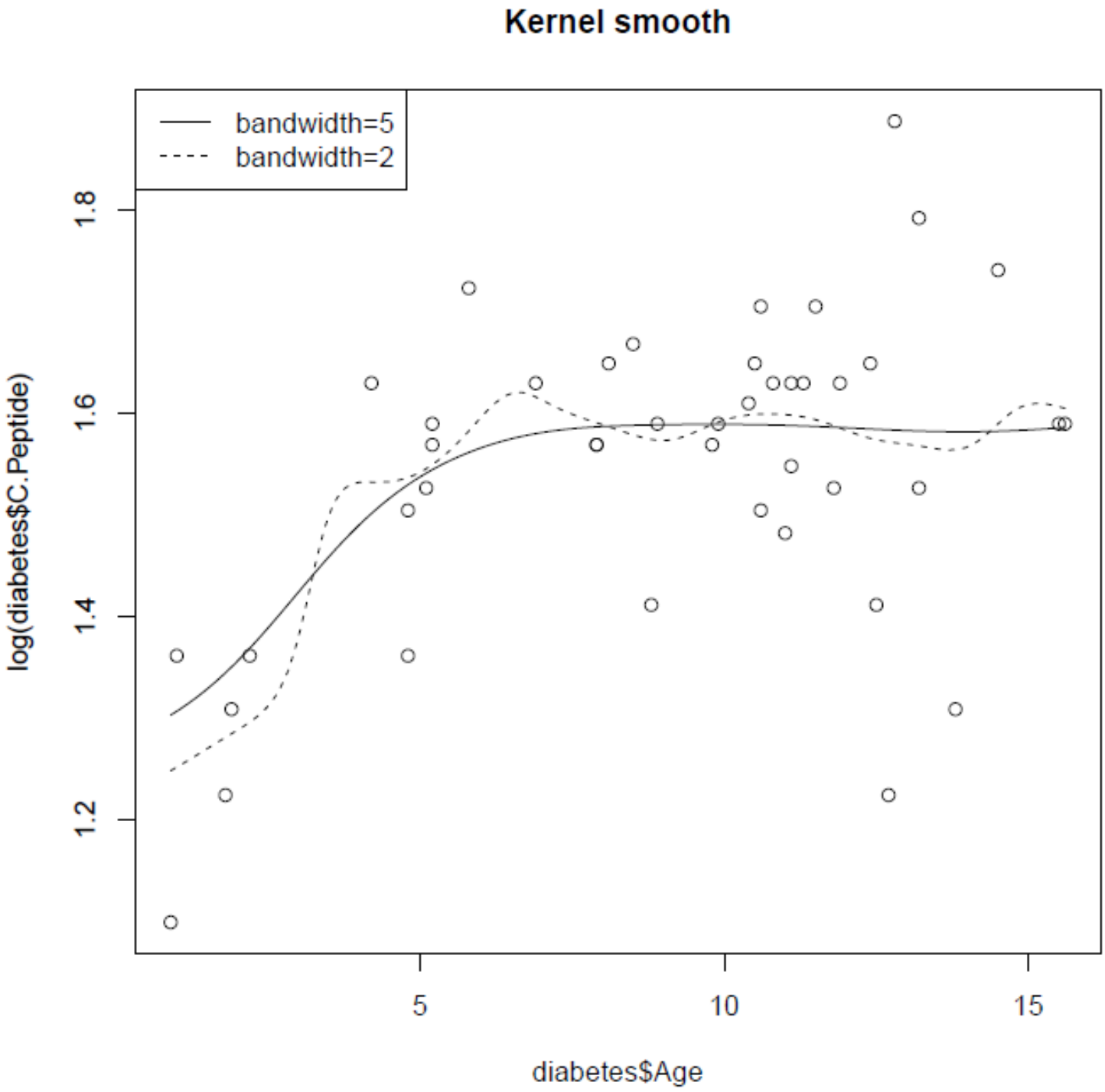
parameter h.

As above, large values of h imply lower variance but higher bias.

Example: Kernel smoothing

In this example we illustrate kernel smoothing for the diabetes data (with `log(C.peptide)` as the response and `age` as the predictor):

```
diabetes <- read.table("/course/data/diabetes.dat", header=TRUE, quote="\"")
diabetes <- diabetes[order(diabetes$Age),]
plot(diabetes$Age,log(diabetes$C.Peptide))
lines(ksmooth(diabetes$Age,log(diabetes$C.Peptide), bandwidth=5.0, kernel="normal"))
lines(ksmooth(diabetes$Age,log(diabetes$C.Peptide), bandwidth=2.0, kernel="normal"),
      lty=2)
title("Kernel smooth")
legend("topleft",c("bandwidth=5", "bandwidth=2"),lty=1:2)
```



As for *loess.smooth()*, the output of the function *ksmooth()* is a list with components **x** and **y** (numeric vectors, by default **x** is the set of observed predictors and **y** gives corresponding values of the kernel smooth).

The argument **bandwidth** controls the degree of smoothing (the default value is usually not sensible) and **kernel** allows specification of the kernel (options **box** and **normal**, see help for details).

The equivalent kernel

Now consider local regression where instead of fitting a line we do a weighted fit with just a constant term: minimize

$$\sum_i w_i(x_0)(y_i - \beta_0(x_0))^2$$

with respect to $\beta_0(x_0)$. Differentiating with respect to $\beta_0(x_0)$ and writing $\hat{\beta}_0(x_0)$ for the minimizer, we have

$$-2\sum_i w_i(x_0)(y_i - \hat{\beta}_0(x_0)) = 0$$

from which we have

$$\hat{\beta}_0(x_0) = \sum_i w_i(x_0) y_i,$$

since $\sum_i w_i(x_0) = 1$. So kernel smoothing can be thought of as local weighted fitting of a model containing just a constant term. The locally weighted averages suffer from boundary bias, due to the asymmetry of the kernel in that region. Fitting straight lines rather than constants can remove this bias exactly to first order.

Local vs.nearest neighbour

Note that for nearest neighbour methods (such as loess) the smoothing window size is adjusted so that it always contains the same number of data points. For kernel smoothers, on the other hand, the smoothing window size (bandwidth) is constant.

This means that loess smooths have constant variance across input space, whereas the variance of kernel smoothers increases with data sparsity. Where data are sparse, nearest neighbour methods are biased, since far away points with a different mean enter the regression.

Local likelihood

The loess approach readily generalises to maximum likelihood fits. Suppose that at x_0, the response follows a distribution from an exponential family with mean $\mu(x_0) = \beta_0 + \beta x_0$ and log-likelihood $\ell(y_0; \mu(x_0))$. The local likelihood at x_0 can then be defined as

$$\ell(\beta_0, \beta; x_0) = \sum_{i=1}^{n} K_\lambda(x_0, x_i)\ell(y_i, \mu(x_0)).$$

Its optimisation then yields the estimate of the mean at x_0: $\hat{\mu}(x_0) = \hat{\beta}_0 + \hat{\beta} x_0$, which may be plotted in all x_0 of interest.

Example: Local likelihood

Recall the trade union data set, which we have used as an example for logistic regression. The package `locfit` fits a nonparametric logistic regression model as follows:

```
trade.union <- read.table("/course/data/tradeunion.dat", header = T)
attach(trade.union)
require(locfit)

# locfit 1.5-9.4      2020-03-24

fit=locfit(union.member~lp(wage,nn=0.7,deg=2), data=trade.union,family="binomial")
plot(fit,ylim = c(0,1),type='n',main="Trade Union Data")
newxx=seq(from = min(wage),to = max(wage),length.out = 100)
newdat=data.frame(wage=newxx)
ypred=predict(fit,newdat,se.fit = T)
lines(newxx,ypred$fit)
lines(newxx,ypred$fit-ypred$se.fit,lty=2)
lines(newxx,ypred$fit+ypred$se.fit,lty=2)
points(jitter(wage), union.member, main = "trade union data set", pch = 3)
```

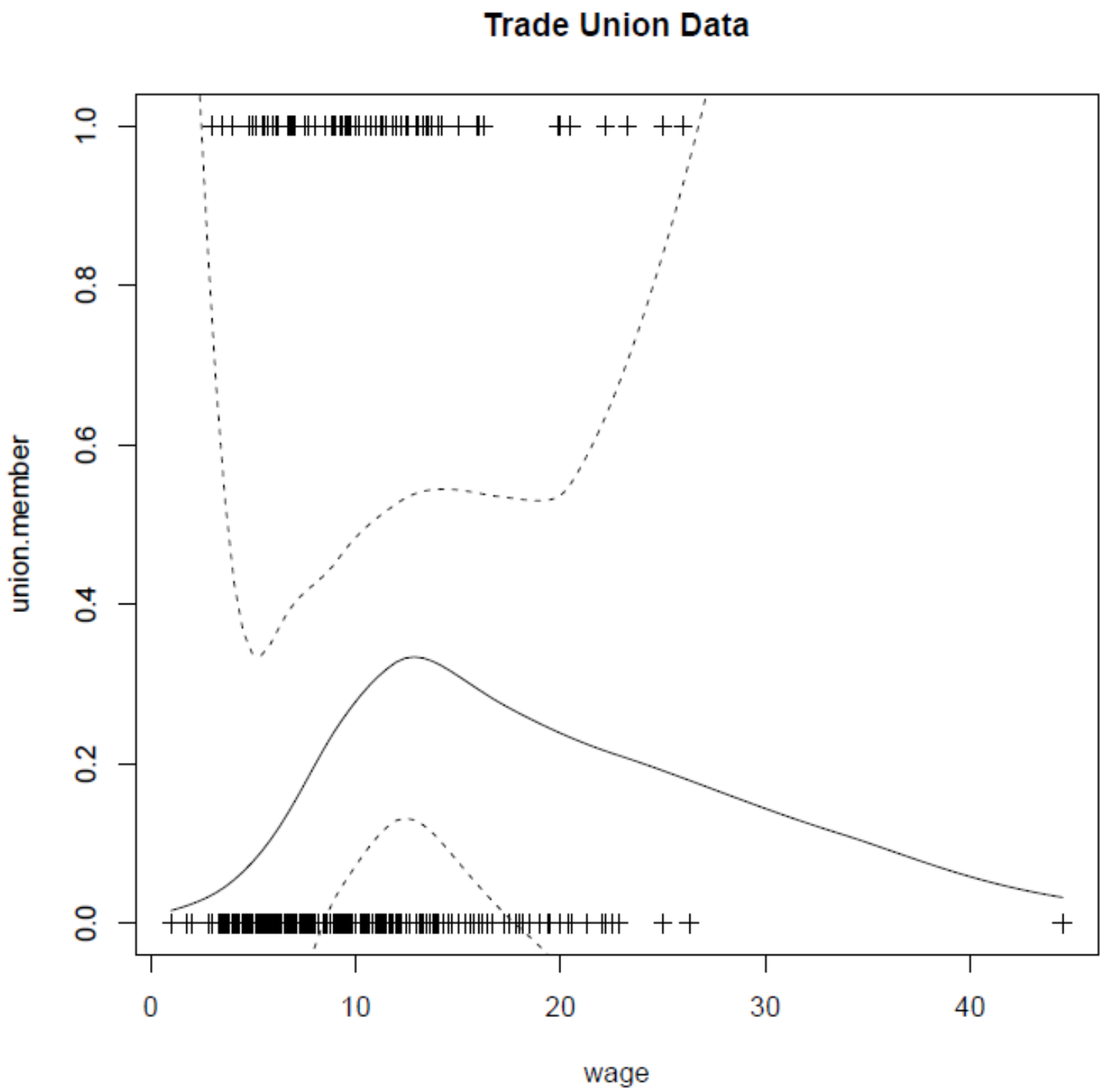


Figure 5.1.3: The local binomial likelihood for a logistic regression model. A (non-simultaneous) confidence band is given for plus/minus one standard deviation.

Activity in R: Loess Smoother

Download the data set **`ethanol`**. The **`ethanol`** data frame contains 88 measurements generated in an experiment in which ethanol was burned in a single cylinder automobile test engine. The variables are **`NOx`** (a measure of the nitrogen oxides in the exhaust), **`C`** (compression ratio of an engine) and **`E`** (a measure of the richness of the air/ethanol mix used).

(a) Do a scatterplot of **`NOx`** against **`E`**.
After looking at the scatterplot you might suggest a polynomial regression model to capture the relationship between the response **`NOx`** and the predictor **`E`**. Fit a quadratic regression model and superimpose the fitted curve on the scatterplot.

(b) Superimpose on the scatterplot in a) a suitable loess smooth. Do you think the smooth captures structure in the data that is missed by the quadratic regression model?

Activity: Loess smoother

Question *Submitted Feb 18th 2024 at 8:10:31 pm*

Consider a locally weighted regression

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2, \tag{5.1.1}$$

which solves a separate weighted least squares problem at each point x_0.

(a) Show that there exists matrix $\boldsymbol{L}$ such that

$$\boldsymbol{L}y = \hat{f},$$

where $\hat{f} = (\hat{f}(z_1), \ldots, \hat{f}(z_m))^\top$ for m points in the interval in which loess curve is to be plotted.

(b) Show that for local linear regression

$$\boldsymbol{L}\mathbf{1}_n = \mathbf{1}_m$$

and

$$\boldsymbol{L}x = z,$$

where $\mathbf{1}_n = (1, \ldots, 1)^\top \in \boldsymbol{R}^n$, and where $x = (x_1, x_2, \ldots, x_n)^\top$ are the given data points and $z = (z_1, \ldots, z_m)^\top$ the points in the regression interval in which the loess curve is to be plotted.

(c) Using the results from b) explain in what sense "$\hat{f}(x)$ is unbiased up to first order".

Solution

(a) We need to show that there exists matrix $\boldsymbol{L}$ such that

$$\boldsymbol{L}y = \hat{f},$$

where where $\hat{f} = (\hat{f}(z_1), \ldots, \hat{f}(z_m))^\top$ for m points in the interval in which loess curve is to be plotted.

Note that $(5.1.1)$ is a weighted least square problem solved at each $z_1, z_2, \ldots, z_m$. The solution here is

$$\hat{f}(z_i) = \hat{\alpha}(z_i) + \hat{\beta}(z_i) z_i.$$

This can be rewritten as

$$\hat{f}(z_i) = (1, z_i)(\boldsymbol{B}^\top \boldsymbol{W}(z_i) \boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{W}(z_i) y.$$

Here

$$\boldsymbol{W}(z_i) = \operatorname{diag}(K_\lambda(z_i, x_i))_{N \times N}$$

and

$$\boldsymbol{B}^\top = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \end{pmatrix}.$$

Moreover,

$$\begin{aligned} \hat{f}(z_i) &= (1, z_i)(\boldsymbol{B}^\top \boldsymbol{W}(z_i) \boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{W}(z_i) y \\ &= \sum_{j=1}^{N} l_j(z_i) y_j \\ &= \tilde{l}(z_i)^\top y. \end{aligned}$$

Now we take ith row of the matrix $\boldsymbol{L}$ to be $l(z_i)^\top$. Thus the ith element in the vector $\boldsymbol{L}y$ is $\tilde{l}(z_i)^\top y = \hat{f}(z_i)$ as required.

(b) Now we need to show that for local linear regression

$$\boldsymbol{L}\mathbf{1}_n = \mathbf{1}_m \tag{5.1.2}$$

and

$$\boldsymbol{L}x = z, \tag{5.1.3}$$

where $\mathbf{1}_n = (1, \ldots, 1)^\top \in \boldsymbol{R}^n$, and where $x = (x_1, x_2, \ldots, x_n)^\top$ are the given data points and $z = (z_1, \ldots, z_m)^\top$ the points in the regression interval in which the loess curve is to be plotted.

We consider here every row of $\boldsymbol{L}\mathbf{1}_n$ separately and prove that they are all 1.

Since $\mathbf{1}_n$ is a leading column of $\boldsymbol{B}$ we have $\mathbf{1}_n = \boldsymbol{B}e_1$, where e_1 is the first $d = 2$ dimensional standard basis vector.

The value of the ith row of $\boldsymbol{L}$ is

$$\begin{aligned}
&(1, z_i)(\boldsymbol{B}^\top \boldsymbol{W}(z_i)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \boldsymbol{W}(z_i)\mathbf{1}_n \\
&= (1, z_i)(\boldsymbol{B}^\top \boldsymbol{W}(z_i)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \boldsymbol{W}(z_i)\boldsymbol{B}e_1 \\
&= (1, z_i)e_1 = 1.
\end{aligned}$$

Thus every element in the vector $\boldsymbol{L}\mathbf{1}_n$ is 1, and thus $\boldsymbol{L}\mathbf{1}_n = \mathbf{1}_m$.

Similarly, we consider every row of $\boldsymbol{L}x$ separately and show that its ith row is z_i.

Similarly, we consider every row of $\boldsymbol{L}x$ separately and show that its ith row is z_i.

$$\begin{aligned}
&(1, z_i)(\boldsymbol{B}^\top \boldsymbol{W}(z_i)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \boldsymbol{W}(z_i)x \\
&= (1, z_i)(\boldsymbol{B}^\top \boldsymbol{W}(z_i)\boldsymbol{B})^{-1}\boldsymbol{B}^\top \boldsymbol{W}(z_i)\boldsymbol{B}e_2 \\
&= (1, z_i)e_2 = z_i,
\end{aligned}$$

since $x = \boldsymbol{B}e_2$ since x is the second column of $\boldsymbol{B}$.

(c) Local linear regression automatically modifies the kernel to correct the bias exactly to first order, a phenomenon known as automatic kernel carpentry.

Consider the following expansion for $\boldsymbol{E}(\hat{f}(x_0))$, using the linearity of local regression and a series expansion of the true funciton f around x_0 we obtain:

$$\begin{aligned}
\boldsymbol{E}\hat{f}(x_0) &= \sum_{i=1}^{N} l_i(x_0) f(x_i) \\
&= f(x_0)\sum_{i=1}^{N} l_i(x_0) + f'(x_0)\sum_{i=1}^{N}(x_i - x_0) l_i(x_0) \\
&+ \frac{1}{2} f^{"}(x_0)\sum_{i=1}^{N}(x_i - x_0)^2 l_i(x_0) + R,
\end{aligned}$$

where the reminder R is typically small under suitable smoothness assumptions.

From $(5.1.2)$ we have $\sum_{i=1}^{N} l_i(x_0) = 1$. Additionally,

$$\begin{aligned}\sum_{i=1}^{N}(x_i - x_0)l_i(x_0) &= \sum_{i=1}^{N} x_i l_i(x_0) - \sum_{i=1}^{N} x_0 l_i(x_0) \\ &= \sum_{i=1}^{N} x_i l_i(x_0) - x_0 \sum_{i=1}^{N} l_i(x_0) \\ &= x_0 - x_0 = 0,\end{aligned}$$

from (2) and (3).

Therefore,

$$\boldsymbol{E}\hat{f}(x_0) = f(x_0) + \frac{1}{2}f^{"}(x_0)\sum_{i=1}^{N}(x_i - x_0)^2 l_i(x_0) + R$$

and

$$\text{bias}(\hat{f}) = \boldsymbol{E}(\hat{f}(x_0)) - f(x_0) = \frac{1}{2}f^{"}(x_0)\sum_{i=1}^{N}(x_i - x_0)^2 l_i(x_0) + R.$$

This means that $\hat{f}(x)$ is unbiased up to first order.

Note also from $\hat{f}(z_i) = \sum_{j=1}^{M} l_j(z_i)y_j$ and the fact that $\sum_{j=1}^{M} l_j(z_i) = 1$ the weights $l_j(x_0)$ are somethimes referred to as the equivalent kernel.

Activity in R: Kernel Smoother

Download the data set **`air`** (there are four columns in this data frame, ozone, radiation, temperature and wind). The data set records New York Air Quality Measurements, May to September 1973.

Plot a scatterplot of **`ozone`** against **`radiation`** and superimpose loess and kernel smooths on the scatterplot. You may need to change the default level of smoothing for the kernel and loess smooths.

Additional Activity

Question 1 *Submitted Feb 18th 2024 at 8:14:31 pm*

Which are true of the running median smoother:

- [ ] if k (the smoothing parameter) is too small, we won't capture any sharp changes in f;
- [x] there may be a substantial boundary bias present in $\hat{f}$;
- [ ] the estimated f is a smooth, differentiable function.

Question 2 *Submitted Feb 18th 2024 at 8:14:39 pm*

Which achieves better boundary behaviour:

- [ ] the running median smoother;
- [x] the running line smoother.

Question 3 *Submitted Feb 18th 2024 at 8:14:51 pm*

Which are true of the loess smoother:

- [x] the loess smoothing considers the weighted least squares criterion

$$\sum_i w_i(x_0)(y_i - \beta_0(x_0) - \beta_1(x_0)x_i)^2$$

  at the point x_0;

- [ ] the weights w_i are larger for x_i which are further away from x_0;
- [x] $K(u)$ - the weight in the loess algorithm has its maximum at $u = 0$.

Question 4 *Submitted Feb 18th 2024 at 8:15:00 pm*

Which are true about kernel smoothing:

- [ ] the choice of the smoothing parameter h is less crucial to the predictive performance than the choice of the kernel;
- [x] large values of h (smoothing parameter) imply lower variance and higher bias;
- [x] kernel smoothing can be thought of as local weithed fitting of a model containing just a constant term.