ch. 14 Nonparametric Regression (ELMR)

14.1 Kernel Estimators

In its simplest form, this is just a moving average estimator. More generally, our estimate of f, called $\hat{f}_\lambda(x)$, is:

$$\hat{f}_\lambda(x) = \frac{1}{n\lambda}\sum_{j=1}^{n} K\left(\frac{x-x_j}{\lambda}\right) Y_j = \frac{1}{n}\sum_{j=1}^{n} w_j Y_j \qquad \text{where} \qquad w_j = K\left(\frac{x-x_j}{\lambda}\right)/\lambda$$

K is a kernel where $\int K = 1$. The moving average kernel is rectangular, but smoother kernels can give better results. λ is called the bandwidth, window width or smoothing parameter. It controls the smoothness of the fitted curve.

If the xs are spaced very unevenly, then this estimator can give poor results. This problem is somewhat ameliorated by the *Nadaraya–Watson* estimator:

$$f_\lambda(x) = \frac{\sum_{j=1}^{n} w_j Y_j}{\sum_{j=1}^{n} w_j} = \frac{1}{c}\sum_{j=1}^{n} w_j Y_j$$

We see that this estimator simply modifies the moving average estimator so that it is a true weighted average where the weights for each y will sum to one.

It is worth understanding the basic asymptotics of kernel estimators. The optimal choice of λ gives:

$$\text{MSE}(x) = E(f(x) - \hat{f}_\lambda(x))^2 = O(n^{-4/5})$$

MSE stands for mean squared error and we see that this decreases at a rate proportional to $n^{-4/5}$ with the sample size. Compare this to the typical parametric estimator where $\text{MSE}(x) = O(n^{-1})$, provided that the parametric model is correct. So the kernel estimator is less efficient. Indeed, the relative difference between the MSEs becomes substantial as the sample size increases. However, if the parametric model is incorrect, the MSE will be $O(1)$ and the fit will not improve past a certain point even with unlimited data. The advantage of the nonparametic approach is the protection against model specification error. Without assuming much stronger restrictions on f, nonparametric estimators cannot do better than $O(n^{-4/5})$.

The implementation of a kernel estimator requires two choices: the kernel and the smoothing parameter. For the choice of kernel, smoothness and compactness are desirable. We prefer smoothness to ensure that the resulting estimator is smooth, so for example, the uniform kernel will give stepped-looking fit that we may wish to avoid. We also prefer a compact kernel because this ensures that only data, local to the point at which f is estimated, is used in the fit. This means that the Gaussian kernel is less desirable, because although it is light in the tails, it is not zero, meaning that the contribution of every point to the fit must be computed. The optimal choice under some standard assumptions is the Epanechnikov kernel:

$$K(x) = \begin{cases} \frac{3}{4}(1-x^2) & |x| < 1 \\ 0 & \text{otherwise} \end{cases}$$

This kernel has the advantage of some smoothness, compactness and rapid computation. This latter feature is important for larger datasets, particularly when resampling

techniques like bootstrap are being used. Even so, any sensible choice of kernel will produce acceptable results, so the choice is not crucially important.

The choice of smoothing parameter λ is critical to the performance of the estimator and far more important than the choice of kernel. If the smoothing parameter is too small, the estimator will be too rough; but if it is too large, important features will be smoothed out.

We demonstrate the Nadaraya–Watson estimator next for a variety of choices of bandwidth on the Old Faithful data shown in Figure 14.2. We use the `ksmooth` function which is part of the R base package. This function lacks many useful features that can be found in some other packages, but it is adequate for simple use. The default uses a uniform kernel, which is somewhat rough. We have changed this to the normal kernel:

```
for(bw in c(0.1,0.5,2)){
with(faithful,{
     plot(waiting ~ eruptions, col=gray(0.75))
     lines(ksmooth(eruptions,waiting,"normal",bw))
     })}
```

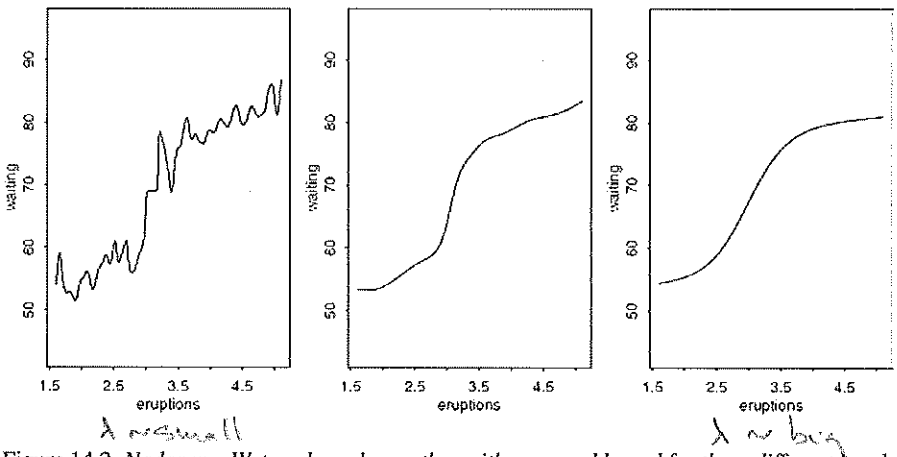


Figure 14.2 *Nadaraya–Watson kernel smoother with a normal kernel for three different bandwidths on the Old Faithful data.*

The central plot in Figure 14.2 is the best choice of the three. Since we do not know the true function relating waiting time and eruption duration, we can only speculate, but it does seem reasonable to expect that this function is quite smooth. The fit on the left does not seem plausible since we would not expect the mean waiting time to vary so much as a function of eruptions. On the other hand, the plot on the right is even smoother than the plot in the middle. It is not so easy to choose between these. Another consideration is that the eye can always visualize additional smoothing, but it is not so easy to imagine what a less smooth fit might look like. For this reason, we recommend picking the least smooth fit that does not show any implausible fluctuations. Of the three plots shown, the middle plot seems best. Smoothers are often used as a graphical aid in interpreting the relationship between variables. In such cases,

visual selection of the amount of smoothing is effective because the user can employ background knowledge to make an appropriate choice and avoid serious mistakes.

You can choose λ interactively using this subjective method. Plot $\hat{f}_\lambda(x)$ for a range of different λ and pick the one that looks best as we have done above. You may need to iterate the choice of λ to focus your decision. Knowledge about what the true relationship might look like can be readily employed.

In cases where the fitted curve will be used to make numerical predictions of future values, the choice of the amount of smoothing has an immediate effect on the outcome. Even here subjective methods may be used. If this method of selecting the amount of smoothing seems disturbingly subjective, we should also understand that the selection of a family of parametric models for the same data would also involve a great deal of subjective choice although this is often not explicitly recognized. Statistical modeling requires us to use our knowledge of what general forms of relationship might be reasonable. It is not possible to determine these forms from the data in an entirely objective manner. Whichever methodology you use, some subjective decisions will be necessary. It is best to accept this and be honest about what these decisions are.

parametric vs. nonparametric

Even so, automatic methods for selecting the amount of smoothing are also useful. Selecting the amount of smoothing using subjective methods requires time and effort. When a large number of smooths are necessary, some automation is desirable. In other cases, the statistician will want to avoid the explicit appearance of subjectivity in the choice. Cross-validation (CV) is a popular general-purpose method. The criterion is:

$$CV(\lambda) = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{f}_{\lambda(j)}(x_j))^2$$

leave-one-out CV

where (j) indicates that point j is left out of the fit. We pick the λ that minimizes this criterion. True cross-validation is computationally expensive, so an approximation to it, known as *generalized cross-validation* or GCV, is sometimes used. There are also many other methods of automatically selecting the λ.

Our practical experience has been that automatic methods, such as CV, often work well, but sometimes produce estimates that are clearly at odds with the amount of smoothing that contextual knowledge would suggest. For this reason, we are unwilling to trust automatic methods completely. We recommend using them as a starting point for a possible interactive exploration of the appropriate amount of smoothing if time permits. They are also useful when very large numbers of smooths are needed such as in the additive modeling approach described in Chapter 15.

When smoothing is used to determine whether f has certain features such as multiple maximums (called *bump hunting*) or monotonicity, special methods are necessary to choose the amount of smoothing since this choice will determine the outcome of the investigation.

The sm library, described in Bowman and Azzalini (1997), allows the computation of the cross-validated choice of smoothing parameter. For example, we find the CV choice of smoothing parameter for the Old Faithful and plot the result:

```
library(sm)
```

```
with(faithful,sm.regression(eruptions, waiting, h=h.select(eruptions,
    ↪ waiting)))
```

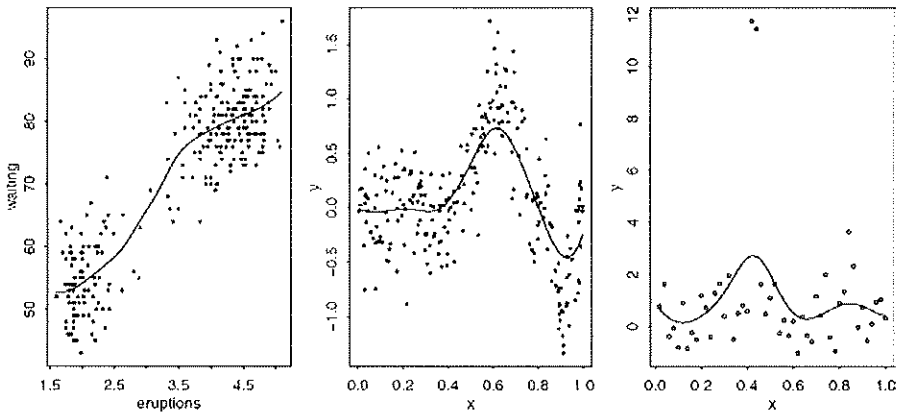


Figure 14.3 *The first panel shows the kernel estimated smooth of the Old Faithful data for a cross-validated choice of smoothing parameter. The second and third panels show the resulting fits for Examples A and B, respectively.*

We see the resulting fit plotted in the first panel of Figure 14.3. The `sm` package uses a Gaussian kernel where the smoothing parameter is the standard deviation of the kernel.

We repeat the exercise for Example A; the plots are shown in the second panel of Figure 14.3. The resulting fit is somewhat oversmoothed.

```
with(exa, sm.regression(x, y, h=h.select(x,y)))
```

Finally, we compute the fit for Example B as seen in the third panel of Figure 14.3.

```
with(exb, sm.regression(x, y, h=h.select(x,y)))
```

We see that the fitted curve notices the two outliers but does not reach out to them. A much smaller choice of smoothing parameter would allow the fitted curve to pass near these two points but only at the price of a much rougher fit elsewhere.

14.2 Splines

Smoothing Splines: The model is $y_i = f(x_i) + \varepsilon_i$, so in the spirit of least squares, we might choose $\hat{f}$ to minimize the MSE: $\frac{1}{n}\sum(y_i - f(x_i))^2$. The solution is $\hat{f}(x_i) = y_i$. This is a "join the dots" regression that is almost certainly too rough. Instead, suppose we choose $\hat{f}$ to minimize a modified least squares criterion:

$$\frac{1}{n}\sum(Y_i - f(x_i))^2 + \lambda \int [f''(x)]^2 dx$$

where $\lambda > 0$ is the smoothing parameter and $\int [f''(x)]^2 dx$ is a *roughness penalty*. When f is rough, the penalty is large, but when f is smooth, the penalty is small. Thus the two parts of the criterion balance fit against smoothness. This is the *smoothing spline* fit.

For this choice of roughness penalty, the solution is of a particular form: $\hat{f}$ is a cubic spline. This means that $\hat{f}$ is a piecewise cubic polynomial in each interval (x_i, x_{i+1}) (assuming that the x_is are unique and sorted). It has the property that $\hat{f}$, $\hat{f}'$ and $\hat{f}''$ are continuous. Given that we know the form of the solution, the estimation is reduced to the parametric problem of estimating the coefficients of the polynomials. This can be done in a numerically efficient way.

Several variations on the basic theme are possible. Other choices of roughness penalty can be considered, where penalties on higher-order derivatives lead to fits with more continuous derivatives. We can also use weights by inserting them in the sum of squares part of the criterion. This feature is useful when smoothing splines are means to an end for some larger procedure that requires weighting. A robust version can be developed by modifying the sum of squares criterion to:

$$\sum \rho(y_i - f(x_i)) + \lambda \int [f''(x)]^2 dx$$

where $\rho(x) = |x|$ is one possible choice.

In R, cross-validation is used to select the smoothing parameter by default. We show this default choice of smoothing for our three test cases:

```
with(faithful,{
    plot(waiting ~ eruptions, col=gray(0.75))
    lines(smooth.spline(eruptions,waiting),lty=2)
})
with(exa,{
    plot(y ~ x, col=gray(0.75))
    lines(x,m)
    lines(smooth.spline(x,y),lty=2)
})
with(exb,{
    plot(y ~ x, col=gray(0.75))
    lines(x,m)
    lines(smooth.spline(x,y),lty=2)
})
```

The fits may be seen in Figure 14.4. The fit for the Old Faithful data looks reasonable. The fit for Example A does a good job of tracking the hills and valleys but overfits in the smoother region. The default choice of smoothing parameter given by CV is a disaster for Example B as the data is just interpolated. This illustrates the danger of blindly relying on automatic smoothing parameter selection methods.

Regression Splines: Regression splines differ from smoothing splines in the following way: for regression splines, the number of knots of the B-splines used for the basis are typically much smaller than the sample size. The number of knots chosen controls the amount of smoothing. For smoothing splines, the observed unique x values are the knots and λ is used to control the smoothing. It is arguable whether the regression spline method is parametric or nonparametric, because once the knots are chosen, a parametric family has been specified with a finite number of parameters. It is the freedom to choose the number of knots that makes the method nonparametric. One of the desirable characteristics of a nonparametric regression estimator is that it should be consistent for smooth functions. This can be achieved for regression

$$\hat{\beta} = \arg\min \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$\hat{f}(x) = \sum_{k=1}^{K} \beta_k b_k(x)$$

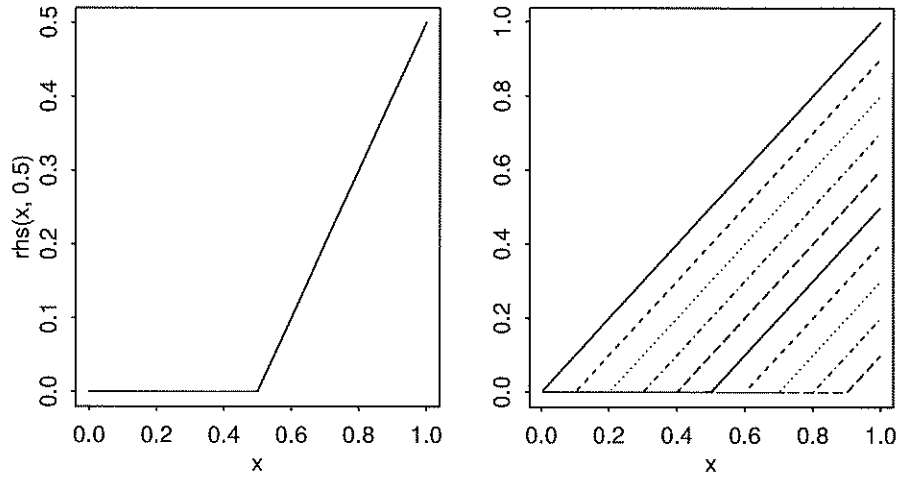


Figure 14.5 *One basis function for linear regression splines shown on the left and the complete set shown on the right.*

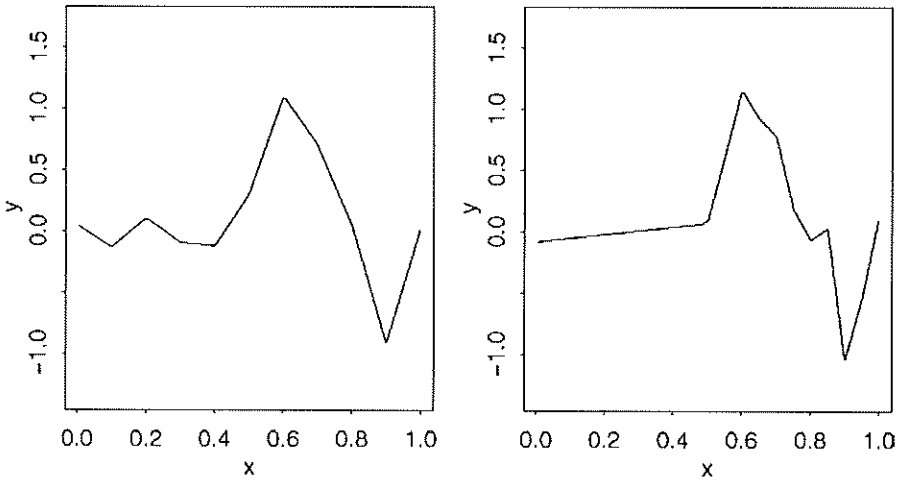


Figure 14.6 *Evenly spaced knots fit shown on the left and knots spread relative to the curvature on the right.*

Ex A Ex A.

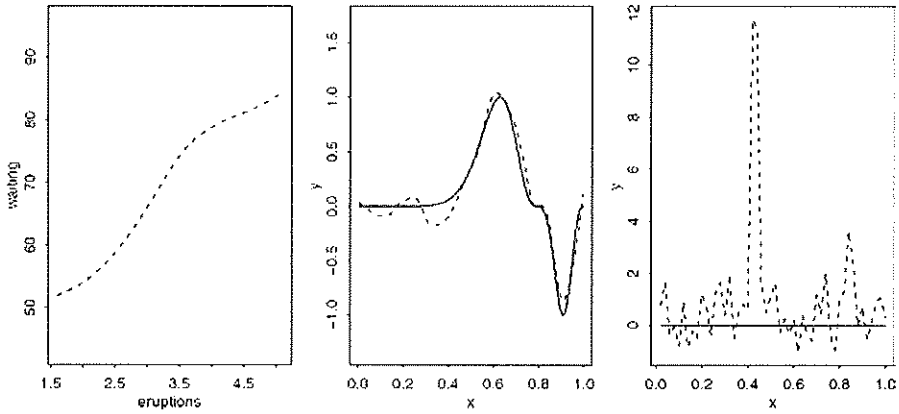


Figure 14.4 *Smoothing spline fits. For Examples A and B, the true function is shown as solid and the spline fit as dashed.*

splines if the number of knots is allowed to increase at an appropriate rate with the sample size.

We demonstrate some regression splines here. We use piecewise linear splines in this example, which are constructed and plotted as follows: knots

```
rhs <- function(x,c) ifelse(x>c,x-c,0)
curve(rhs(x,0.5),0,1)
```

where the spline is shown in the first panel of Figure 14.5. Now we define some knots for Example A:

```
(knots <- 0:9/10)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

and compute a design matrix of splines with knots at these points for each x:

```
dm  <- outer(exa$x,knots,rhs)
matplot(exa$x,dm,type="l",col=1, xlab="x", ylab="")
```

where the basis functions are shown in the second panel of Figure 14.5. Now we compute and display the regression fit:

```
lmod <- lm(exa$y ~ dm)
plot(y ~ x, exa, col=gray(0.75))
lines(exa$x,predict(lmod))
```

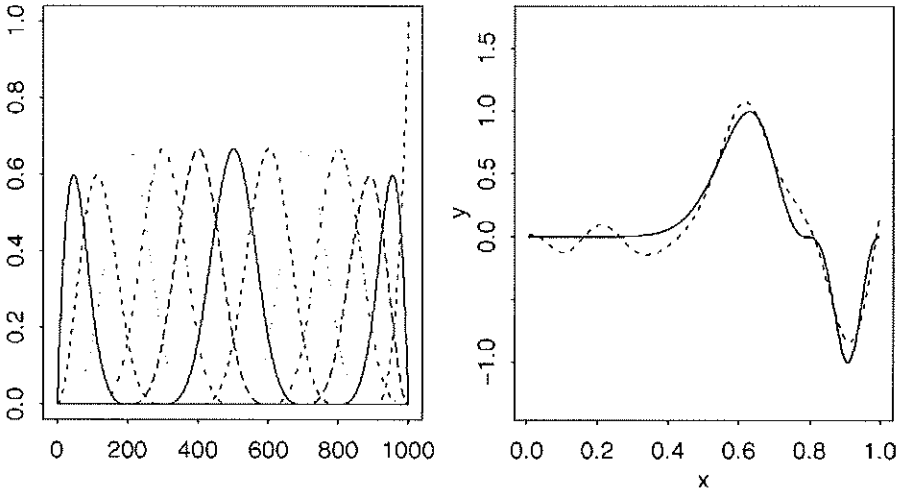
where the plot is shown in the first panel of Figure 14.6. Because the basis functions are piecewise linear, the fit is also piecewise linear. A better fit may be obtained by adjusting the knots so that they are denser in regions of greater curvature:

```
newknots <- c(0,0.5,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95)
dmn  <- outer(exa$x,newknots,rhs)
lmod <- lm(exa$y ~ dmn)
plot(y ~x, exa, col=gray(0.75))
lines(exa$x,predict(lmod))
```

where the plot is shown in the second panel of Figure 14.6. We obtain a better fit but only by using our knowledge of the true curvature. This knowledge would not be available for real data, so more practical methods place the knots adaptively according to the *estimated* curvature.

One can achieve a smoother fit by using higher-order splines. The `bs()` function can be used to generate the appropriate spline basis. The default is cubic B-splines. We display 12 cubic B-splines evenly spaced on the [0,1] interval. The splines close to the boundary take a different form as seen in the first panel of Figure 14.7:

```
library(splines)
matplot(bs(seq(0,1,length=1000),df=12),type="l",ylab="",col=1)
```



Ex A

Figure 14.7 *A cubic B-spline basis is shown in the left panel and the resulting fit to the Example A data is shown in the right panel.*

We can now use least squares to determine the coefficients. We then display the fit as seen in the second panel of Figure 14.7:

```
lmod <- lm(y ~ bs(x,12),exa)
plot(y ~ x, exa, col=gray(0.75))
lines(m ~ x, exa)
lines(predict(lmod) ~ x, exa, lty=2)
```

We see a smooth fit, but again we could do better by placing more knots at the points of high curvature and fewer in the flatter regions.

"semiautomatic" cubic regression splines (12 knots are evenly spaced)

14.3 Local Polynomials (loess or lowess)

Both the kernel and spline methods have been relatively vulnerable to outliers as seen by their performance on Example B. The fits can be improved with some manual intervention, either to remove the outliers or to increase the smoothing parameters. However, smoothing is frequently just a small part of an analysis and so we might wish to avoid giving each smooth individual attention. Furthermore, habitual removal of outliers is an ad hoc strategy that is better replaced with a method that deals with long-tailed errors gracefully. The local polynomial method combines robustness ideas from linear regression and local fitting ideas from kernel methods.

First we select a window. We then fit a polynomial to the data in that window using robust methods. The predicted response at the middle of the window is the

fitted value. We then slide the window over the range of the data, repeating the fitting process as the window moves. The most well-known implementation of this type of smoothing is called *lowess* or *loess* and is due to Cleveland (1979).

As with any smoothing method, there are choices to be made. We need to choose the order of the polynomial fit. A quadratic allows us to capture peaks and valleys in the function. However, a linear term also performs well and is the default choice in the `loess` function. As with most smoothers, it is important to pick the window width well. The default choice takes three quarters of the data and may not be a good choice as we shall see below.

For the Old Faithful data, the default choice is satisfactory, as seen in the first panel of Figure 14.8:

```
with(faithful,{
    plot(waiting ~ eruptions, col=gray(0.75))
    f <- loess(waiting ~ eruptions)
    i <- order(eruptions)
    lines(f$x[i],f$fitted[i])
})
```

For Example A, the default choice is too large. The choice that minimizes the integrated squared error between the estimated and true function requires a span (proportion of the range) of 0.22. Both fits are seen in the middle panel of Figure 14.8:

```
with(exa,{
    plot(y ~ x, col=gray(0.75))
    lines(m ~ x)
    f <- loess(y ~ x)
    lines(f$x,f$fitted,lty=2)
    f <- loess(y ~ x,span=0.22)
    lines(f$x,f$fitted,lty=5)
})
```

In practice, the true function is, of course, unknown and we would need to select the span ourselves, but this optimal choice does at least show how well loess can do in the best of circumstances. The fit is similar to that for smoothing splines.

For Example B, the optimal choice of span is one (that is all the data). This is not surprising since the true function is a constant and so maximal smoothing is desired. We can see that the robust qualities of loess prevent the fit from becoming too distorted by the two outliers even with the default choice of smoothing span:

```
with(exb,{
    plot(y ~ x, col=gray(0.75))
    lines(m ~ x)
    f <- loess(y ~ x)
    lines(f$x,f$fitted,lty=2)
    f <- loess(y ~ x,span=1)
    lines(f$x,f$fitted,lty=5)
})
```

14.4 Confidence Bands

It is helpful to have some expression of uncertainty in the curve estimates. Both the regression spline and loess methods use (local) linear fitting using parametric methods. These same methods naturally generate a standard error which can be used to construct a confidence interval at any point in x. We may connect these intervals to-

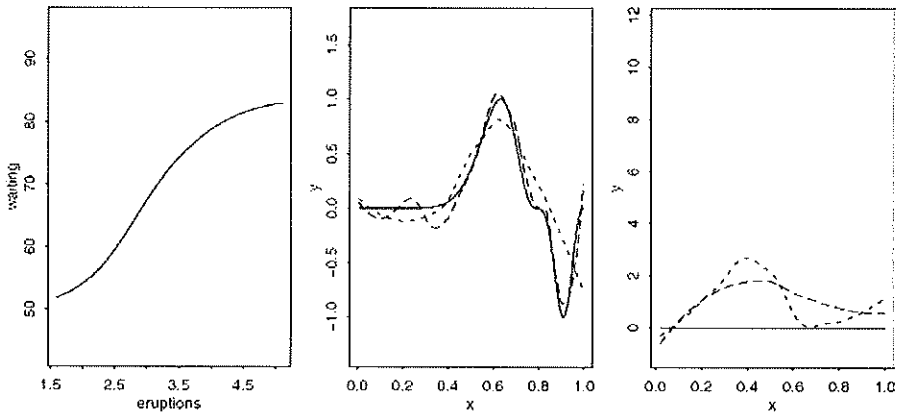


Figure 14.8 *Loess smoothing: Old Faithful data is shown in the left panel with the default amount of smoothing. Example A data is shown in the middle and B in the right panel. The true function is shown as a solid line along with the default choice (dotted) and respective optimal amounts of smoothing (dashed) are also shown.*

gether to form a confidence band. These are conveniently constructed and displayed using the `ggplot2` package.

We construct the 95% confidence band for Example A data using loess:

```
ggplot(exa, aes(x=x,y=y)) + geom_point(alpha=0.25) + geom_smooth(
    ↪ method="loess", span=0.22) + geom_line(aes(x=x,y=m),linetype=2)
```

The plot is seen in the first panel of Figure 14.9. We have added the true function as a dashed line. We observe that the true function falls just outside the band in a few areas. However, the band we have constructed is a *pointwise* confidence band. The 95% confidence applies at each point but since we have a wide range of points, the 95% probability of the interval containing the true value cannot hold across the range. For this we would need a *simultaneous* confidence band.

We can also construct a band using splines. We need the `mgcv` package which includes a spline smoother.

```
library(mgcv)
ggplot(exa, aes(x=x,y=y)) + geom_point(alpha=0.25) + geom_smooth(
    ↪ method="gam", formula=y ~ s(x, k=20)) + geom_line(aes(x=x,y=m),
    ↪ linetype=2)
```

We see the resulting plot in the second panel of Figure 14.9. In this case, we have manually chosen the smoothing parameter, `k=20`, representing the degrees of freedom in the fit. It is larger to accommodate the variation in this function, producing a better although not perfect fit to that seen in Figure 14.6.

14.5 Wavelets

Regression splines are an example of a basis function approach to fitting. We approximate the curve by a family of basis functions, $\phi_i(x)$, so that $\hat{f}(x) = \sum_i c_i \phi_i(x)$. Thus the fit requires estimating the coefficients, c_i. The choice of basis functions will

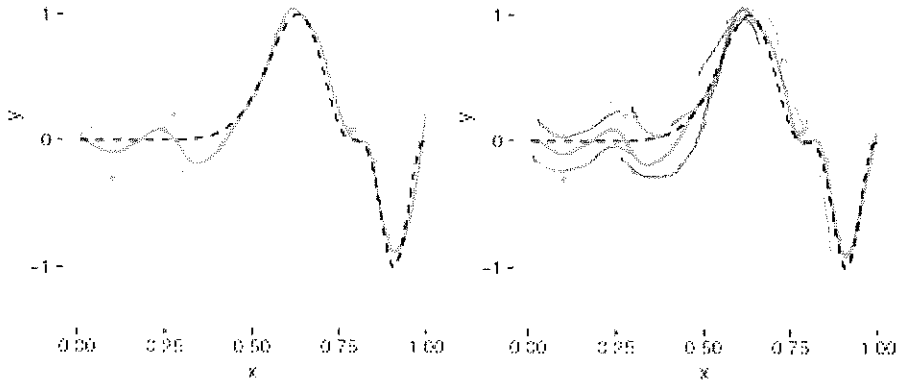


Figure 14.9 *95% confidence bands for loess (left) and spline (right) fits to Example A.*

determine the properties of the fitted curve. The estimation of c_i is particularly easy if the basis functions are orthogonal.

Examples of orthogonal bases are orthogonal polynomials and the Fourier basis. The disadvantage of both these families is that the basis functions are not compactly supported so that the fit of each basis function depends on the whole data. This means that these fits lack the desirable local fit properties that we have seen in previously discussed smoothing methods. Although Fourier methods are popular for some applications, particularly those involving periodic data, they are not typically used for general-purpose smoothing.

Cubic B-splines are compactly supported, but they are not orthogonal. *Wavelets* have the advantage that they are compactly supported and can be defined so as to possess the orthogonality property. They also possess the *multiresolution* property which allows them to fit the grosser features of the curve while focusing on the finer detail where necessary.

We begin with the simplest type of wavelet: the *Haar* basis. The *mother wavelet* for the Haar family is defined on the interval $[0,1)$ as:

$$w(x) = \begin{cases} 1 & x \leq 1/2 \\ -1 & x > 1/2 \end{cases}$$

We generate the members of the family by dilating and translating this function. The next two members of the family are defined on $[0,1/2)$ and $[1/2,1)$ by rescaling the mother wavelet to these two intervals. The next four members are defined on the quarter intervals in the same way. We can index the family members by level j and within the level by k so that each function will be defined on the interval $[k/2^j,(k+1)/2^j)$ and takes the form:

$$h_n(x) = 2^{j/2} w(2^j x - k)$$

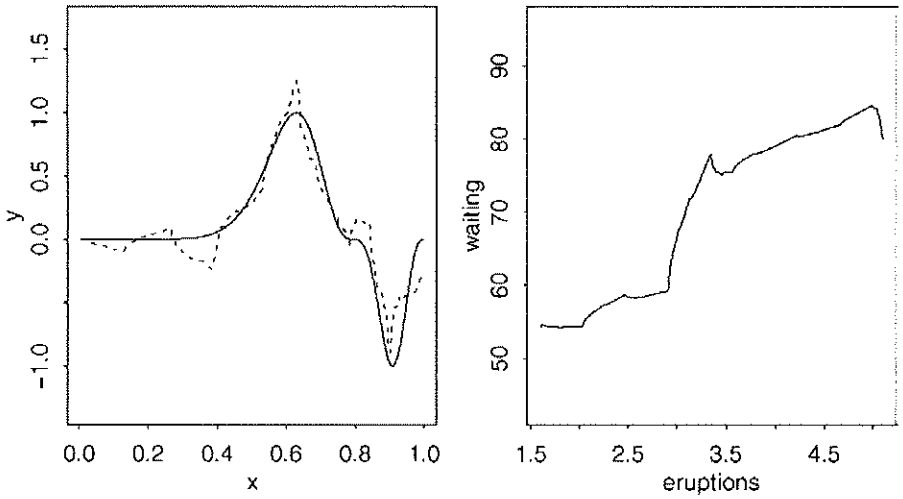


Figure 14.13 *Daubechies wavelet N=2 thresholded fit to the Example A data shown on the left. Irregular wavelet fit to the Old Faithful data is shown on the right.*

```
x <- with(faithful, (eruptions-min(eruptions))/(max(eruptions)-min(
    ↪ eruptions)))
gridof <- makegrid(x, faithful$waiting)
wdof <- irregwd(gridof, bc="symmetric")
wtof <- threshold(wdof)
wrof <- wr(wtof)
plot(waiting ~ eruptions, faithful, col=grey(0.75))
with(faithful,lines(seq(min(eruptions),max(eruptions),len=512), wrof))
```

The resulting plot, shown in the second panel of Figure 14.13, reveals a discontinuity in the fit not seen in previous plots. This demonstrates one of the main advantages of the wavelet method in handling and revealing discontinuities. Wavelet methods are particularly useful in processing very large datasets such as those found in image and sound files because the filtering method of thresholding coefficients can drastically reduce file size without losing much information.

14.6 Discussion of Methods

We have presented only a selection of the wide variety of methods available. For example, *nearest neighbor* methods adjust for varying density in the predictor space by adjusting window widths to be wider in sparser regions and narrower in denser regions. Window widths are also nonconstant in *variable bandwidth* methods. Such methods are particularly appropriate for functions like Example A where the smoothness of the function varies. We would like to use a smaller bandwidth in regions where the function changes rapidly but a wider one where it is more constant.

Bayesian methods of smoothing are evident in the *Gaussian Process* method as described in Rasmussen and Williams (2006). This method is particularly appropriate if you have prior knowledge and also works well on quite small datasets.

why Use ?

The construction of alternate smoothing methods has long been a popular topic of interest for statisticians and researchers in other fields. Because no definitive solution is possible, this has encouraged the development of a wide range of methods. But it is important not to allow the enthusiasm to solve an interesting technical problem to overshadow the purpose of a data analysis. We propose four possible objectives:

Description Sometimes we simply want to draw a line on a scatterplot to aid in the visual interpretation of the relationship displayed. We do not want to spend a lot of time or effort in constructing this smooth. We want a method that is simple and reliable.

Auxiliary In some applications, the smoothed fit is needed as part of some larger analysis. The smooth is not the principal aim. For example, the smooth might be used to impute some missing values. In such examples, we need to choose the smoothing method to optimize the wider objective. Sometimes, this means choosing rougher or smoother results than we would pick in a standalone problem.

Prediction Interpolating values in noisy data is one example where these methods could be useful. Extrapolation is more problematic as this requires some assumptions about how the function will behave outside the range of the data. Parametric methods do better here as, although extrapolation is inherently risky, it is more transparent how they will behave. We may also want to construct confidence statements which is easier to do with the basis function methods, such as splines, because we can mirror the parametric methods.

Explanation In linear modeling, the most common regression question concerns whether there is a relationship between x and y. No relationship corresponds to an assertion that the function is constant. We can construct confidence bands for some of the methods. We can then observe whether a constant function fits between the bands to decide the question. Even so, it would be better to directly construct a hypothesis test for which parametric methods are more amenable. Nonparametric methods do at least give us more information about situations where the assertion of no relationship only holds for part of the range of the predictor.

So when should we use nonparametric regression and which particular method should we choose? In the univariate case, we can describe three situations. When there is very little noise, interpolation (or at most, very mild smoothing) is the best way to recover the relation between x and y. Splines are good for this purpose. When there is a moderate amount of noise, nonparametric methods are most effective. There is enough noise to make smoothing worthwhile but also enough signal to justify a flexible fit. When the amount of noise becomes larger, parametric methods become relatively more attractive. There is insufficient signal to justify anything more than a simple model.

It is not reasonable to claim that any one smoother is better than the rest. The best choice of smoother will depend on the characteristics of the data and knowledge about the true underlying relationship. The choice will also depend on whether the fit is to be made automatically or with human intervention. When only a single dataset is being considered, it's simple enough to craft the fit and intervene if a particular method produces unreasonable results. If a large number of datasets are to be fit

Note: In regression (parametric, nonparametric) we seek to replace the scatterplot of x vs. y with a curve (this is smoothing)!