# 7
# Moving Beyond Linearity


So far in this book, we have mostly focused on linear models. Linear models are relatively simple to describe and implement, and have advantages over other approaches in terms of interpretation and inference. However, standard linear regression can have significant limitations in terms of predictive power. This is because the linearity assumption is almost always an approximation, and sometimes a poor one. In Chapter 6 we see that we can improve upon least squares using ridge regression, the lasso, principal components regression, and other techniques. In that setting, the improvement is obtained by reducing the complexity of the linear model, and hence the variance of the estimates. But we are still using a linear model, which can only be improved so far! In this chapter we relax the linearity assumption while still attempting to maintain as much interpretability as possible. We do this by examining very simple extensions of linear models like polynomial regression and step functions, as well as more sophisticated approaches such as splines, local regression, and generalized additive models.

- *Polynomial regression* extends the linear model by adding extra predictors, obtained by raising each of the original predictors to a power. For example, a *cubic* regression uses three variables, $X$, $X^2$, and $X^3$, as predictors. This approach provides a simple way to provide a nonlinear fit to data.

- *Step functions* cut the range of a variable into $K$ distinct regions in order to produce a qualitative variable. This has the effect of fitting a piecewise constant function.

- *Regression splines* are more flexible than polynomials and step functions, and in fact are an extension of the two. They involve dividing the range of $X$ into $K$ distinct regions. Within each region, a polynomial function is fit to the data. However, these polynomials are



G. James et al., *An Introduction to Statistical Learning*, Springer Texts in Statistics,
https://doi.org/10.1007/978-3-031-38747-0_7

constrained so that they join smoothly at the region boundaries, or *knots*. Provided that the interval is divided into enough regions, this can produce an extremely flexible fit.

- *Smoothing splines* are similar to regression splines, but arise in a slightly different situation. Smoothing splines result from minimizing a residual sum of squares criterion subject to a smoothness penalty.
- *Local regression* is similar to splines, but differs in an important way. The regions are allowed to overlap, and indeed they do so in a very smooth way.
- *Generalized additive models* allow us to extend the methods above to deal with multiple predictors.

In Sections 7.1–7.6, we present a number of approaches for modeling the relationship between a response $Y$ and a single predictor $X$ in a flexible way. In Section 7.7, we show that these approaches can be seamlessly integrated in order to model a response $Y$ as a function of several predictors $X_1, \ldots, X_p$.

## 7.1 Polynomial Regression

Historically, the standard way to extend linear regression to settings in which the relationship between the predictors and the response is non-linear has been to replace the standard linear model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

with a polynomial function

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \epsilon_i, \tag{7.1}$$

where $\epsilon_i$ is the error term. This approach is known as *polynomial regression*, and in fact we saw an example of this method in Section 3.3.2. For large enough degree $d$, a polynomial regression allows us to produce an extremely non-linear curve. Notice that the coefficients in (7.1) can be easily estimated using least squares linear regression because this is just a standard linear model with predictors $x_i, x_i^2, x_i^3, \ldots, x_i^d$. Generally speaking, it is unusual to use $d$ greater than 3 or 4 because for large values of $d$, the polynomial curve can become overly flexible and can take on some very strange shapes. This is especially true near the boundary of the $X$ variable.

polynomial regression

The left-hand panel in Figure 7.1 is a plot of `wage` against `age` for the `Wage` data set, which contains income and demographic information for males who reside in the central Atlantic region of the United States. We see the results of fitting a degree-4 polynomial using least squares (solid blue curve). Even though this is a linear regression model like any other, the individual coefficients are not of particular interest. Instead, we look at the entire fitted function across a grid of 63 values for `age` from 18 to 80 in order to understand the relationship between `age` and `wage`.

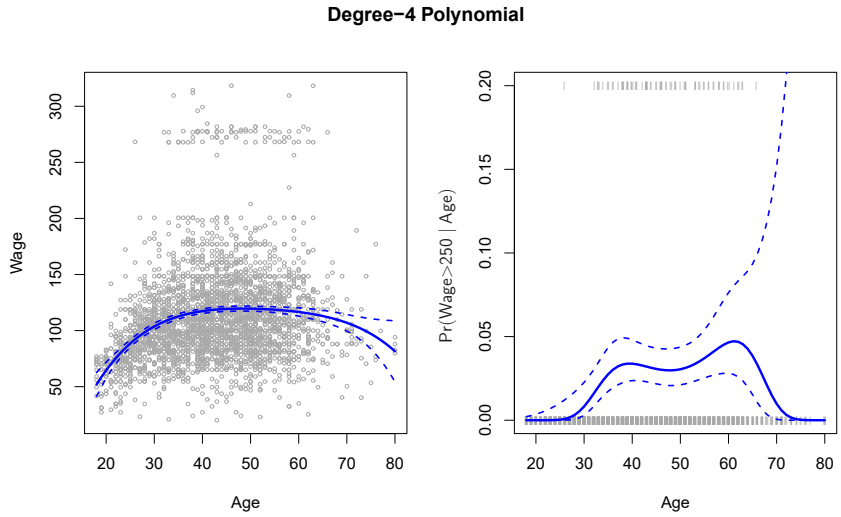


**FIGURE 7.1.** *The* `Wage` *data.* Left: *The solid blue curve is a degree-4 polynomial of* `wage` *(in thousands of dollars) as a function of* `age`*, fit by least squares. The dashed curves indicate an estimated 95 % confidence interval.* Right: *We model the binary event* `wage>250` *using logistic regression, again with a degree-4 polynomial. The fitted posterior probability of* `wage` *exceeding* $250,000 *is shown in blue, along with an estimated 95 % confidence interval.*

In Figure 7.1, a pair of dashed curves accompanies the fit; these are (2×) standard error curves. Let's see how these arise. Suppose we have computed the fit at a particular value of `age`, $x_0$:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4. \tag{7.2}$$

What is the variance of the fit, i.e. $\mathrm{Var}\hat{f}(x_0)$? Least squares returns variance estimates for each of the fitted coefficients $\hat{\beta}_j$, as well as the covariances between pairs of coefficient estimates. We can use these to compute the estimated variance of $\hat{f}(x_0)$.[1] The estimated *pointwise* standard error of $\hat{f}(x_0)$ is the square-root of this variance. This computation is repeated at each reference point $x_0$, and we plot the fitted curve, as well as twice the standard error on either side of the fitted curve. We plot twice the standard error because, for normally distributed error terms, this quantity corresponds to an approximate 95 % confidence interval.

It seems like the wages in Figure 7.1 are from two distinct populations: there appears to be a *high earners* group earning more than $250,000 per annum, as well as a *low earners* group. We can treat `wage` as a binary variable by splitting it into these two groups. Logistic regression can then be used to predict this binary response, using polynomial functions of `age`

[1] If $\hat{\mathbf{C}}$ is the $5 \times 5$ covariance matrix of the $\hat{\beta}_j$, and if $\ell_0^T = (1, x_0, x_0^2, x_0^3, x_0^4)$, then $\mathrm{Var}[\hat{f}(x_0)] = \ell_0^T \hat{\mathbf{C}} \ell_0$.

as predictors. In other words, we fit the model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d)}. \quad (7.3)$$

The result is shown in the right-hand panel of Figure 7.1. The gray marks on the top and bottom of the panel indicate the ages of the high earners and the low earners. The solid blue curve indicates the fitted probabilities of being a high earner, as a function of `age`. The estimated 95 % confidence interval is shown as well. We see that here the confidence intervals are fairly wide, especially on the right-hand side. Although the sample size for this data set is substantial ($n = 3{,}000$), there are only 79 high earners, which results in a high variance in the estimated coefficients and consequently wide confidence intervals.

## 7.2 Step Functions

Using polynomial functions of the features as predictors in a linear model imposes a *global* structure on the non-linear function of $X$. We can instead use *step functions* in order to avoid imposing such a global structure. Here we break the range of $X$ into *bins*, and fit a different constant in each bin. This amounts to converting a continuous variable into an *ordered categorical variable*.

step function

ordered categorical variable

In greater detail, we create cutpoints $c_1$, $c_2, \ldots, c_K$ in the range of $X$, and then construct $K + 1$ new variables

$$\begin{aligned} C_0(X) &= I(X < c_1), \\ C_1(X) &= I(c_1 \le X < c_2), \\ C_2(X) &= I(c_2 \le X < c_3), \\ &\vdots \\ C_{K-1}(X) &= I(c_{K-1} \le X < c_K), \\ C_K(X) &= I(c_K \le X), \end{aligned} \quad (7.4)$$

where $I(\cdot)$ is an *indicator function* that returns a 1 if the condition is true, and returns a 0 otherwise. For example, $I(c_K \le X)$ equals 1 if $c_K \le X$, and equals 0 otherwise. These are sometimes called *dummy* variables. Notice that for any value of $X$, $C_0(X) + C_1(X) + \cdots + C_K(X) = 1$, since $X$ must be in exactly one of the $K + 1$ intervals. We then use least squares to fit a linear model using $C_1(X), C_2(X), \ldots, C_K(X)$ as predictors[2]:

indicator function

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon_i. \quad (7.5)$$

For a given value of $X$, at most one of $C_1, C_2, \ldots, C_K$ can be non-zero. Note that when $X < c_1$, all of the predictors in (7.5) are zero, so $\beta_0$ can

[2]We exclude $C_0(X)$ as a predictor in (7.5) because it is redundant with the intercept. This is similar to the fact that we need only two dummy variables to code a qualitative variable with three levels, provided that the model will contain an intercept. The decision to exclude $C_0(X)$ instead of some other $C_k(X)$ in (7.5) is arbitrary. Alternatively, we could include $C_0(X), C_1(X), \ldots, C_K(X)$, and exclude the intercept.

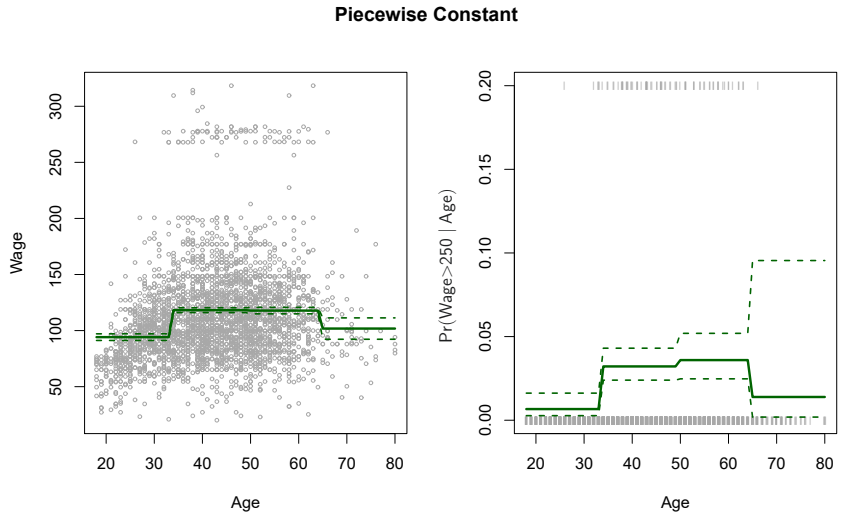


**FIGURE 7.2.** *The* `Wage` *data.* Left: *The solid curve displays the fitted value from a least squares regression of* `wage` *(in thousands of dollars) using step functions of* `age`*. The dashed curves indicate an estimated 95 % confidence interval.* Right: *We model the binary event* `wage>250` *using logistic regression, again using step functions of* `age`*. The fitted posterior probability of* `wage` *exceeding* \$250,000 *is shown, along with an estimated 95 % confidence interval.*

be interpreted as the mean value of $Y$ for $X < c_1$. By comparison, (7.5) predicts a response of $\beta_0+\beta_j$ for $c_j \leq X < c_{j+1}$, so $\beta_j$ represents the average increase in the response for $X$ in $c_j \leq X < c_{j+1}$ relative to $X < c_1$.

An example of fitting step functions to the `Wage` data from Figure 7.1 is shown in the left-hand panel of Figure 7.2. We also fit the logistic regression model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 C_1(x_i) + \cdots + \beta_K C_K(x_i))}{1 + \exp(\beta_0 + \beta_1 C_1(x_i) + \cdots + \beta_K C_K(x_i))} \tag{7.6}$$

in order to predict the probability that an individual is a high earner on the basis of `age`. The right-hand panel of Figure 7.2 displays the fitted posterior probabilities obtained using this approach.

Unfortunately, unless there are natural breakpoints in the predictors, piecewise-constant functions can miss the action. For example, in the left-hand panel of Figure 7.2, the first bin clearly misses the increasing trend of `wage` with `age`. Nevertheless, step function approaches are very popular in biostatistics and epidemiology, among other disciplines. For example, 5-year age groups are often used to define the bins.

## 7.3 Basis Functions

Polynomial and piecewise-constant regression models are in fact special cases of a *basis function* approach. The idea is to have at hand a fam-

basis function

ily of functions or transformations that can be applied to a variable $X$: $b_1(X), b_2(X), \ldots, b_K(X)$. Instead of fitting a linear model in $X$, we fit the model

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \cdots + \beta_K b_K(x_i) + \epsilon_i. \quad (7.7)$$

Note that the basis functions $b_1(\cdot), b_2(\cdot), \ldots, b_K(\cdot)$ are fixed and known. (In other words, we choose the functions ahead of time.) For polynomial regression, the basis functions are $b_j(x_i) = x_i^j$, and for piecewise constant functions they are $b_j(x_i) = I(c_j \leq x_i < c_{j+1})$. We can think of (7.7) as a standard linear model with predictors $b_1(x_i), b_2(x_i), \ldots, b_K(x_i)$. Hence, we can use least squares to estimate the unknown regression coefficients in (7.7). Importantly, this means that all of the inference tools for linear models that are discussed in Chapter 3, such as standard errors for the coefficient estimates and F-statistics for the model's overall significance, are available in this setting.

Thus far we have considered the use of polynomial functions and piecewise constant functions for our basis functions; however, many alternatives are possible. For instance, we can use wavelets or Fourier series to construct basis functions. In the next section, we investigate a very common choice for a basis function: *regression splines*.

regression spline

## 7.4 Regression Splines

Now we discuss a flexible class of basis functions that extends upon the polynomial regression and piecewise constant regression approaches that we have just seen.

### *7.4.1 Piecewise Polynomials*

Instead of fitting a high-degree polynomial over the entire range of $X$, *piecewise polynomial regression* involves fitting separate low-degree polynomials over different regions of $X$. For example, a piecewise cubic polynomial works by fitting a cubic regression model of the form

piecewise polynomial regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i, \quad (7.8)$$

where the coefficients $\beta_0$, $\beta_1$, $\beta_2$, and $\beta_3$ differ in different parts of the range of $X$. The points where the coefficients change are called *knots*.

knot

For example, a piecewise cubic with no knots is just a standard cubic polynomial, as in (7.1) with $d = 3$. A piecewise cubic polynomial with a single knot at a point $c$ takes the form

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{if } x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

In other words, we fit two different polynomial functions to the data, one on the subset of the observations with $x_i < c$, and one on the subset of the observations with $x_i \geq c$. The first polynomial function has coefficients

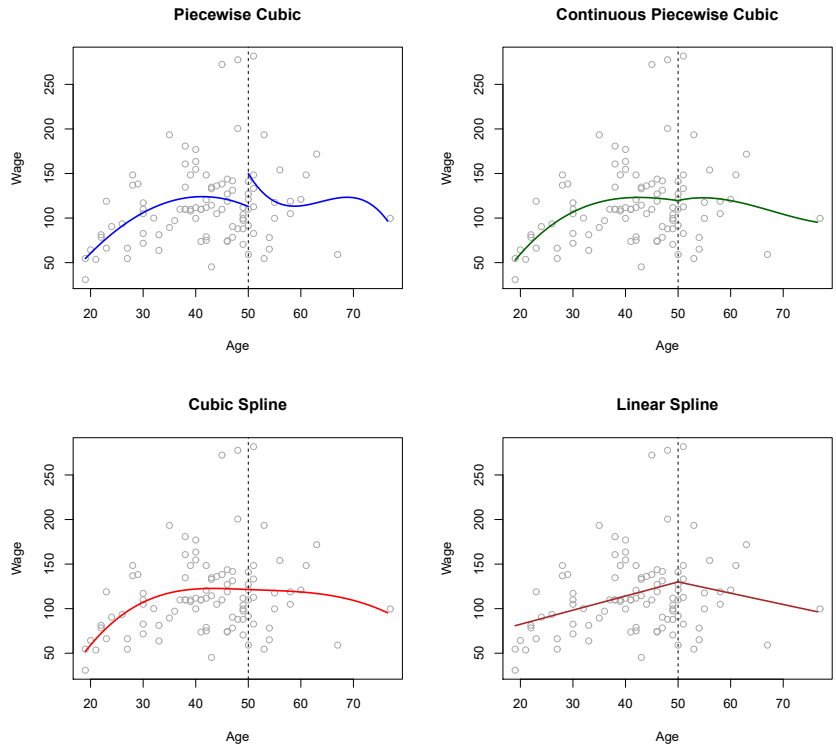


**FIGURE 7.3.** *Various piecewise polynomials are fit to a subset of the* `Wage` *data, with a knot at* `age=50`. Top Left: *The cubic polynomials are unconstrained.* Top Right: *The cubic polynomials are constrained to be continuous at* `age=50`. Bottom Left: *The cubic polynomials are constrained to be continuous, and to have continuous first and second derivatives.* Bottom Right: *A linear spline is shown, which is constrained to be continuous.*

$\beta_{01}, \beta_{11}, \beta_{21}$, and $\beta_{31}$, and the second has coefficients $\beta_{02}, \beta_{12}, \beta_{22}$, and $\beta_{32}$. Each of these polynomial functions can be fit using least squares applied to simple functions of the original predictor.

Using more knots leads to a more flexible piecewise polynomial. In general, if we place $K$ different knots throughout the range of $X$, then we will end up fitting $K + 1$ different cubic polynomials. Note that we do not need to use a cubic polynomial. For example, we can instead fit piecewise linear functions. In fact, our piecewise constant functions of Section 7.2 are piecewise polynomials of degree 0!

The top left panel of Figure 7.3 shows a piecewise cubic polynomial fit to a subset of the `Wage` data, with a single knot at `age=50`. We immediately see a problem: the function is discontinuous and looks ridiculous! Since each polynomial has four parameters, we are using a total of eight *degrees of freedom* in fitting this piecewise polynomial model.

degrees of freedom

### 7.4.2 Constraints and Splines

The top left panel of Figure 7.3 looks wrong because the fitted curve is just too flexible. To remedy this problem, we can fit a piecewise polynomial under the *constraint* that the fitted curve must be continuous. In other words, there cannot be a jump when age=50. The top right plot in Figure 7.3 shows the resulting fit. This looks better than the top left plot, but the V-shaped join looks unnatural.

In the lower left plot, we have added two additional constraints: now both the first and second *derivatives* of the piecewise polynomials are continuous at age=50. In other words, we are requiring that the piecewise polynomial be not only continuous when age=50, but also very *smooth*. Each constraint that we impose on the piecewise cubic polynomials effectively frees up one degree of freedom, by reducing the complexity of the resulting piecewise polynomial fit. So in the top left plot, we are using eight degrees of freedom, but in the bottom left plot we imposed three constraints (continuity, continuity of the first derivative, and continuity of the second derivative) and so are left with five degrees of freedom. The curve in the bottom left plot is called a *cubic spline*.[3] In general, a cubic spline with $K$ knots uses a total of $4 + K$ degrees of freedom.

derivative

cubic spline

In Figure 7.3, the lower right plot is a *linear spline*, which is continuous at age=50. The general definition of a degree-$d$ spline is that it is a piecewise degree-$d$ polynomial, with continuity in derivatives up to degree $d - 1$ at each knot. Therefore, a linear spline is obtained by fitting a line in each region of the predictor space defined by the knots, requiring continuity at each knot.

linear spline

In Figure 7.3, there is a single knot at age=50. Of course, we could add more knots, and impose continuity at each.

### 7.4.3 The Spline Basis Representation

The regression splines that we just saw in the previous section may have seemed somewhat complex: how can we fit a piecewise degree-$d$ polynomial under the constraint that it (and possibly its first $d - 1$ derivatives) be continuous? It turns out that we can use the basis model (7.7) to represent a regression spline. A cubic spline with $K$ knots can be modeled as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i, \tag{7.9}$$

for an appropriate choice of basis functions $b_1, b_2, \ldots, b_{K+3}$. The model (7.9) can then be fit using least squares.

Just as there were several ways to represent polynomials, there are also many equivalent ways to represent cubic splines using different choices of basis functions in (7.9). The most direct way to represent a cubic spline using (7.9) is to start off with a basis for a cubic polynomial—namely, $x, x^2$, and $x^3$—and then add one *truncated power basis* function per knot.

truncated power basis

[3] Cubic splines are popular because most human eyes cannot detect the discontinuity at the knots.

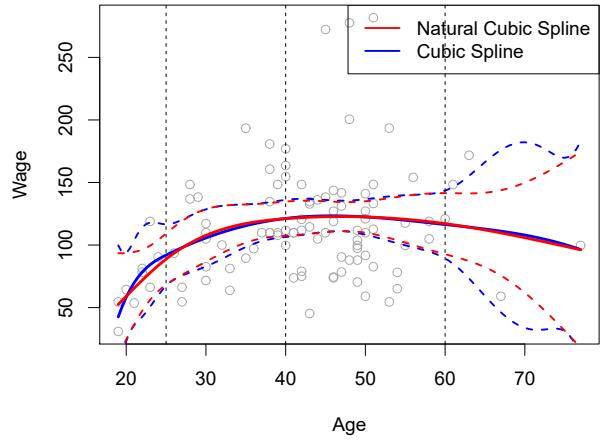


**FIGURE 7.4.** *A cubic spline and a natural cubic spline, with three knots, fit to a subset of the* `Wage` *data. The dashed lines denote the knot locations.*

A truncated power basis function is defined as

$$h(x,\xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise,} \end{cases} \tag{7.10}$$

where $\xi$ is the knot. One can show that adding a term of the form $\beta_4 h(x,\xi)$ to the model (7.8) for a cubic polynomial will lead to a discontinuity in only the third derivative at $\xi$; the function will remain continuous, with continuous first and second derivatives, at each of the knots.

In other words, in order to fit a cubic spline to a data set with $K$ knots, we perform least squares regression with an intercept and $3+K$ predictors, of the form $X, X^2, X^3, h(X,\xi_1), h(X,\xi_2), \ldots, h(X,\xi_K)$, where $\xi_1, \ldots, \xi_K$ are the knots. This amounts to estimating a total of $K+4$ regression coefficients; for this reason, fitting a cubic spline with $K$ knots uses $K+4$ degrees of freedom.

Unfortunately, splines can have high variance at the outer range of the predictors—that is, when $X$ takes on either a very small or very large value. Figure 7.4 shows a fit to the `Wage` data with three knots. We see that the confidence bands in the boundary region appear fairly wild. A *natural spline* is a regression spline with additional *boundary constraints*: the function is required to be linear at the boundary (in the region where $X$ is smaller than the smallest knot, or larger than the largest knot). This additional constraint means that natural splines generally produce more stable estimates at the boundaries. In Figure 7.4, a natural cubic spline is also displayed as a red line. Note that the corresponding confidence intervals are narrower.

natural spline

### *7.4.4 Choosing the Number and Locations of the Knots*

When we fit a spline, where should we place the knots? The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly. Hence, one

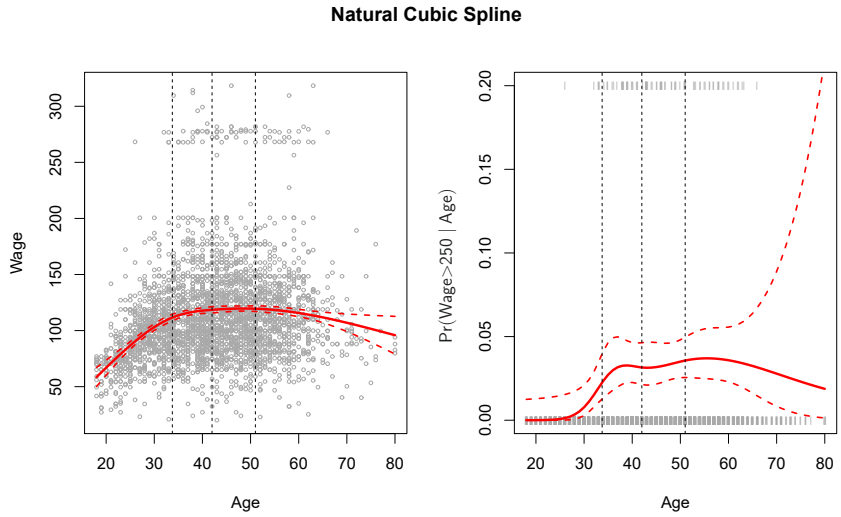


**FIGURE 7.5.** *A natural cubic spline function with four degrees of freedom is fit to the* `Wage` *data.* Left: *A spline is fit to* `wage` *(in thousands of dollars) as a function of* `age`. Right: *Logistic regression is used to model the binary event* `wage>250` *as a function of* `age`. *The fitted posterior probability of* `wage` *exceeding* $250,000 *is shown. The dashed lines denote the knot locations.*

option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable. While this option can work well, in practice it is common to place knots in a uniform fashion. One way to do this is to specify the desired degrees of freedom, and then have the software automatically place the corresponding number of knots at uniform quantiles of the data.

Figure 7.5 shows an example on the `Wage` data. As in Figure 7.4, we have fit a natural cubic spline with three knots, except this time the knot locations were chosen automatically as the 25th, 50th, and 75th percentiles of `age`. This was specified by requesting four degrees of freedom. The argument by which four degrees of freedom leads to three interior knots is somewhat technical.[4]

How many knots should we use, or equivalently how many degrees of freedom should our spline contain? One option is to try out different numbers of knots and see which produces the best looking curve. A somewhat more objective approach is to use cross-validation, as discussed in Chapters 5 and 6. With this method, we remove a portion of the data (say 10 %), fit a spline with a certain number of knots to the remaining data, and then use the spline to make predictions for the held-out portion. We repeat this process multiple times until each observation has been left out once, and

[4]There are actually five knots, including the two boundary knots. A cubic spline with five knots has nine degrees of freedom. But natural cubic splines have two additional *natural* constraints at each boundary to enforce linearity, resulting in $9 - 4 = 5$ degrees of freedom. Since this includes a constant, which is absorbed in the intercept, we count it as four degrees of freedom.

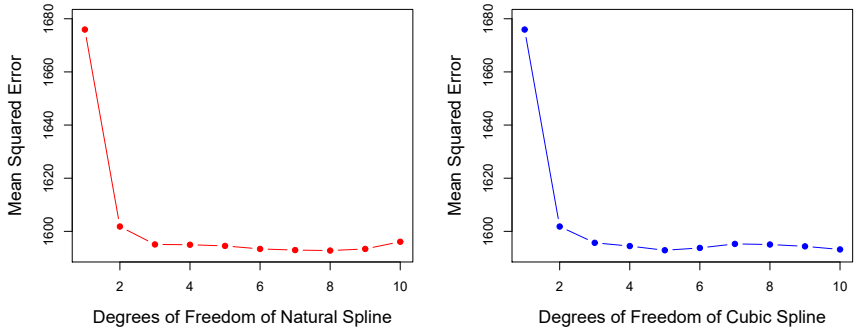


**FIGURE 7.6.** *Ten-fold cross-validated mean squared errors for selecting the degrees of freedom when fitting splines to the* `Wage` *data. The response is* `wage` *and the predictor* `age`. Left: *A natural cubic spline.* Right: *A cubic spline.*

then compute the overall cross-validated RSS. This procedure can be repeated for different numbers of knots $K$. Then the value of $K$ giving the smallest RSS is chosen.

Figure 7.6 shows ten-fold cross-validated mean squared errors for splines with various degrees of freedom fit to the `Wage` data. The left-hand panel corresponds to a natural cubic spline and the right-hand panel to a cubic spline. The two methods produce almost identical results, with clear evidence that a one-degree fit (a linear regression) is not adequate. Both curves flatten out quickly, and it seems that three degrees of freedom for the natural spline and four degrees of freedom for the cubic spline are quite adequate.

In Section 7.7 we fit additive spline models simultaneously on several variables at a time. This could potentially require the selection of degrees of freedom for each variable. In cases like this we typically adopt a more pragmatic approach and set the degrees of freedom to a fixed number, say four, for all terms.

### *7.4.5 Comparison to Polynomial Regression*

Figure 7.7 compares a natural cubic spline with 15 degrees of freedom to a degree-15 polynomial on the `Wage` data set. The extra flexibility in the polynomial produces undesirable results at the boundaries, while the natural cubic spline still provides a reasonable fit to the data. Regression splines often give superior results to polynomial regression. This is because unlike polynomials, which must use a high degree (exponent in the highest monomial term, e.g. $X^{15}$) to produce flexible fits, splines introduce flexibility by increasing the number of knots but keeping the degree fixed. Generally, this approach produces more stable estimates. Splines also allow us to place more knots, and hence flexibility, over regions where the function $f$ seems to be changing rapidly, and fewer knots where $f$ appears more stable.

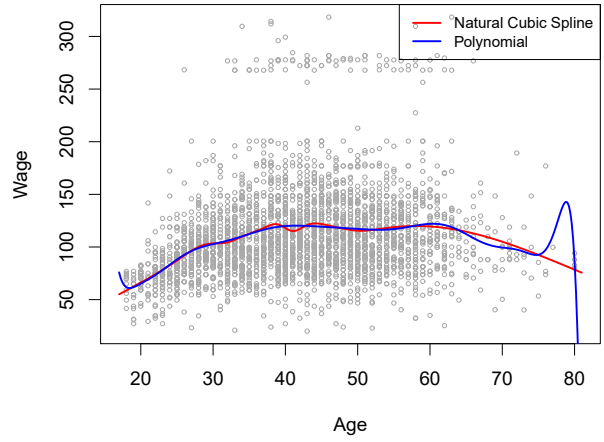


**FIGURE 7.7.** *On the* `Wage` *data set, a natural cubic spline with 15 degrees of freedom is compared to a degree-15 polynomial. Polynomials can show wild behavior, especially near the tails.*

## 7.5 Smoothing Splines

In the last section we discussed regression splines, which we create by specifying a set of knots, producing a sequence of basis functions, and then using least squares to estimate the spline coefficients. We now introduce a somewhat different approach that also produces a spline.

### *7.5.1 An Overview of Smoothing Splines*

In fitting a smooth curve to a set of data, what we really want to do is find some function, say $g(x)$, that fits the observed data well: that is, we want RSS $= \sum_{i=1}^{n}(y_i - g(x_i))^2$ to be small. However, there is a problem with this approach. If we don't put any constraints on $g(x_i)$, then we can always make RSS zero simply by choosing $g$ such that it *interpolates* all of the $y_i$. Such a function would woefully overfit the data—it would be far too flexible. What we really want is a function $g$ that makes RSS small, but that is also *smooth*.

How might we ensure that $g$ is smooth? There are a number of ways to do this. A natural approach is to find the function $g$ that minimizes

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt \tag{7.11}$$

where $\lambda$ is a nonnegative *tuning parameter*. The function $g$ that minimizes (7.11) is known as a *smoothing spline*.

smoothing spline

What does (7.11) mean? Equation 7.11 takes the "Loss+Penalty" formulation that we encounter in the context of ridge regression and the lasso in Chapter 6. The term $\sum_{i=1}^{n}(y_i - g(x_i))^2$ is a *loss function* that encourages $g$ to fit the data well, and the term $\lambda \int g''(t)^2 dt$ is a *penalty term* that penalizes the variability in $g$. The notation $g''(t)$ indicates the second derivative of the function $g$. The first derivative $g'(t)$ measures the slope

loss function

of a function at $t$, and the second derivative corresponds to the amount by which the slope is changing. Hence, broadly speaking, the second derivative of a function is a measure of its *roughness*: it is large in absolute value if $g(t)$ is very wiggly near $t$, and it is close to zero otherwise. (The second derivative of a straight line is zero; note that a line is perfectly smooth.) The $\int$ notation is an *integral*, which we can think of as a summation over the range of $t$. In other words, $\int g''(t)^2 dt$ is simply a measure of the total change in the function $g'(t)$, over its entire range. If $g$ is very smooth, then $g'(t)$ will be close to constant and $\int g''(t)^2 dt$ will take on a small value. Conversely, if $g$ is jumpy and variable then $g'(t)$ will vary significantly and $\int g''(t)^2 dt$ will take on a large value. Therefore, in (7.11), $\lambda \int g''(t)^2 dt$ encourages $g$ to be smooth. The larger the value of $\lambda$, the smoother $g$ will be.

When $\lambda = 0$, then the penalty term in (7.11) has no effect, and so the function $g$ will be very jumpy and will exactly interpolate the training observations. When $\lambda \to \infty$, $g$ will be perfectly smooth—it will just be a straight line that passes as closely as possible to the training points. In fact, in this case, $g$ will be the linear least squares line, since the loss function in (7.11) amounts to minimizing the residual sum of squares. For an intermediate value of $\lambda$, $g$ will approximate the training observations but will be somewhat smooth. We see that $\lambda$ controls the bias-variance trade-off of the smoothing spline.

The function $g(x)$ that minimizes (7.11) can be shown to have some special properties: it is a piecewise cubic polynomial with knots at the unique values of $x_1, \ldots, x_n$, and continuous first and second derivatives at each knot. Furthermore, it is linear in the region outside of the extreme knots. In other words, *the function $g(x)$ that minimizes (7.11) is a natural cubic spline with knots at $x_1, \ldots, x_n$!* However, it is not the same natural cubic spline that one would get if one applied the basis function approach described in Section 7.4.3 with knots at $x_1, \ldots, x_n$—rather, it is a *shrunken* version of such a natural cubic spline, where the value of the tuning parameter $\lambda$ in (7.11) controls the level of shrinkage.

### 7.5.2 Choosing the Smoothing Parameter $\lambda$

We have seen that a smoothing spline is simply a natural cubic spline with knots at every unique value of $x_i$. It might seem that a smoothing spline will have far too many degrees of freedom, since a knot at each data point allows a great deal of flexibility. But the tuning parameter $\lambda$ controls the roughness of the smoothing spline, and hence the *effective degrees of freedom*. It is possible to show that as $\lambda$ increases from 0 to $\infty$, the effective degrees of freedom, which we write $df_\lambda$, decrease from $n$ to 2.

effective degrees of freedom

In the context of smoothing splines, why do we discuss *effective* degrees of freedom instead of degrees of freedom? Usually degrees of freedom refer to the number of free parameters, such as the number of coefficients fit in a polynomial or cubic spline. Although a smoothing spline has $n$ parameters and hence $n$ nominal degrees of freedom, these $n$ parameters are heavily constrained or shrunk down. Hence $df_\lambda$ is a measure of the flexibility of the smoothing spline—the higher it is, the more flexible (and the lower-bias but higher-variance) the smoothing spline. The definition of effective degrees of

freedom is somewhat technical. We can write

$$\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y}, \tag{7.12}$$

where $\hat{\mathbf{g}}_\lambda$ is the solution to (7.11) for a particular choice of $\lambda$—that is, it is an $n$-vector containing the fitted values of the smoothing spline at the training points $x_1, \ldots, x_n$. Equation 7.12 indicates that the vector of fitted values when applying a smoothing spline to the data can be written as a $n \times n$ matrix $\mathbf{S}_\lambda$ (for which there is a formula) times the response vector $\mathbf{y}$. Then the effective degrees of freedom is defined to be

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}, \tag{7.13}$$

the sum of the diagonal elements of the matrix $\mathbf{S}_\lambda$.

In fitting a smoothing spline, we do not need to select the number or location of the knots—there will be a knot at each training observation, $x_1, \ldots, x_n$. Instead, we have another problem: we need to choose the value of $\lambda$. It should come as no surprise that one possible solution to this problem is cross-validation. In other words, we can find the value of $\lambda$ that makes the cross-validated RSS as small as possible. It turns out that the *leave-one-out* cross-validation error (LOOCV) can be computed very efficiently for smoothing splines, with essentially the same cost as computing a single fit, using the following formula:

$$\text{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n} \left[ \frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}} \right]^2 .$$

The notation $\hat{g}_\lambda^{(-i)}(x_i)$ indicates the fitted value for this smoothing spline evaluated at $x_i$, where the fit uses all of the training observations except for the $i$th observation $(x_i, y_i)$. In contrast, $\hat{g}_\lambda(x_i)$ indicates the smoothing spline function fit to all of the training observations and evaluated at $x_i$. This remarkable formula says that we can compute each of these *leave-one-out* fits using only $\hat{g}_\lambda$, the original fit to *all* of the data![5] We have a very similar formula (5.2) on page 205 in Chapter 5 for least squares linear regression. Using (5.2), we can very quickly perform LOOCV for the regression splines discussed earlier in this chapter, as well as for least squares regression using arbitrary basis functions.

Figure 7.8 shows the results from fitting a smoothing spline to the `Wage` data. The red curve indicates the fit obtained from pre-specifying that we would like a smoothing spline with 16 effective degrees of freedom. The blue curve is the smoothing spline obtained when $\lambda$ is chosen using LOOCV; in this case, the value of $\lambda$ chosen results in 6.8 effective degrees of freedom (computed using (7.13)). For this data, there is little discernible difference between the two smoothing splines, beyond the fact that the one with 16 degrees of freedom seems slightly wigglier. Since there is little difference between the two fits, the smoothing spline fit with 6.8 degrees of freedom

[5]The exact formulas for computing $\hat{g}(x_i)$ and $\mathbf{S}_\lambda$ are very technical; however, efficient algorithms are available for computing these quantities.

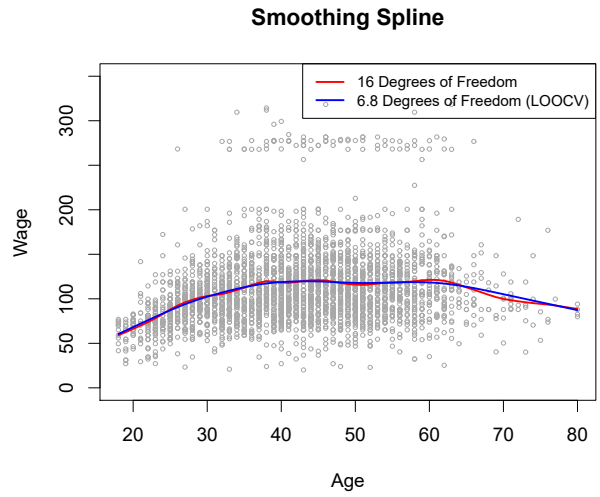


**FIGURE 7.8.** *Smoothing spline fits to the* `Wage` *data. The red curve results from specifying* 16 *effective degrees of freedom. For the blue curve,* $\lambda$ *was found automatically by leave-one-out cross-validation, which resulted in* 6.8 *effective degrees of freedom.*

is preferable, since in general simpler models are better unless the data provides evidence in support of a more complex model.

## 7.6 Local Regression

*Local regression* is a different approach for fitting flexible non-linear functions, which involves computing the fit at a target point $x_0$ using only the nearby training observations. Figure 7.9 illustrates the idea on some simulated data, with one target point near 0.4, and another near the boundary at 0.05. In this figure the blue line represents the function $f(x)$ from which the data were generated, and the light orange line corresponds to the local regression estimate $\hat{f}(x)$. Local regression is described in Algorithm 7.1.

local regression

Note that in Step 3 of Algorithm 7.1, the weights $K_{i0}$ will differ for each value of $x_0$. In other words, in order to obtain the local regression fit at a new point, we need to fit a new weighted least squares regression model by minimizing (7.14) for a new set of weights. Local regression is sometimes referred to as a *memory-based* procedure, because like nearest-neighbors, we need all the training data each time we wish to compute a prediction. We will avoid getting into the technical details of local regression here—there are books written on the topic.

In order to perform local regression, there are a number of choices to be made, such as how to define the weighting function $K$, and whether to fit a linear, constant, or quadratic regression in Step 3. (Equation 7.14 corresponds to a linear regression.) While all of these choices make some difference, the most important choice is the *span* $s$, which is the proportion of points used to compute the local regression at $x_0$, as defined in Step 1 above. The span plays a role like that of the tuning parameter $\lambda$ in smooth-

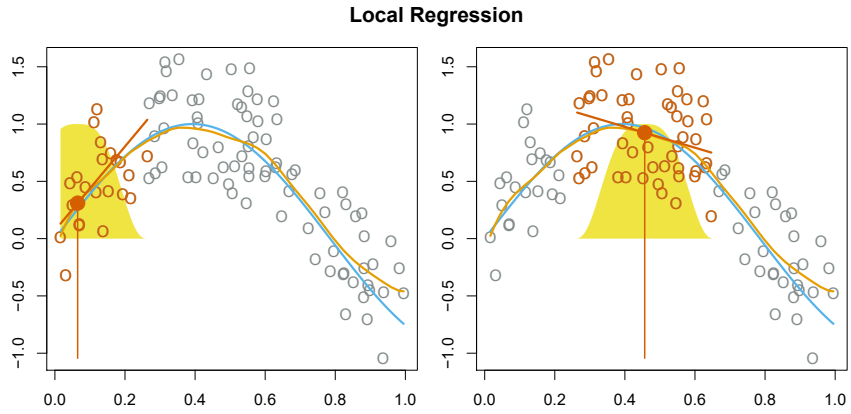


**FIGURE 7.9.** *Local regression illustrated on some simulated data, where the blue curve represents $f(x)$ from which the data were generated, and the light orange curve corresponds to the local regression estimate $\hat{f}(x)$. The orange colored points are local to the target point $x_0$, represented by the orange vertical line. The yellow bell-shape superimposed on the plot indicates weights assigned to each point, decreasing to zero with distance from the target point. The fit $\hat{f}(x_0)$ at $x_0$ is obtained by fitting a weighted linear regression (orange line segment), and using the fitted value at $x_0$ (orange solid dot) as the estimate $\hat{f}(x_0)$.*

ing splines: it controls the flexibility of the non-linear fit. The smaller the value of $s$, the more *local* and wiggly will be our fit; alternatively, a very large value of $s$ will lead to a global fit to the data using all of the training observations. We can again use cross-validation to choose $s$, or we can specify it directly. Figure 7.10 displays local linear regression fits on the `Wage` data, using two values of $s$: 0.7 and 0.2. As expected, the fit obtained using $s = 0.7$ is smoother than that obtained using $s = 0.2$.

The idea of local regression can be generalized in many different ways. In a setting with multiple features $X_1, X_2, \ldots, X_p$, one very useful generalization involves fitting a multiple linear regression model that is global in some variables, but local in another, such as time. Such *varying coefficient models* are a useful way of adapting a model to the most recently gathered data. Local regression also generalizes very naturally when we want to fit models that are local in a pair of variables $X_1$ and $X_2$, rather than one. We can simply use two-dimensional neighborhoods, and fit bivariate linear regression models using the observations that are near each target point in two-dimensional space. Theoretically the same approach can be implemented in higher dimensions, using linear regressions fit to $p$-dimensional neighborhoods. However, local regression can perform poorly if $p$ is much larger than about 3 or 4 because there will generally be very few training observations close to $x_0$. Nearest-neighbors regression, discussed in Chapter 3, suffers from a similar problem in high dimensions.

varying coefficient model

**Algorithm 7.1** *Local Regression At $X = x_0$*

1. Gather the fraction $s = k/n$ of training points whose $x_i$ are closest to $x_0$.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from $x_0$ has weight zero, and the closest has the highest weight. All but these $k$ nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the $y_i$ on the $x_i$ using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at $x_0$ is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

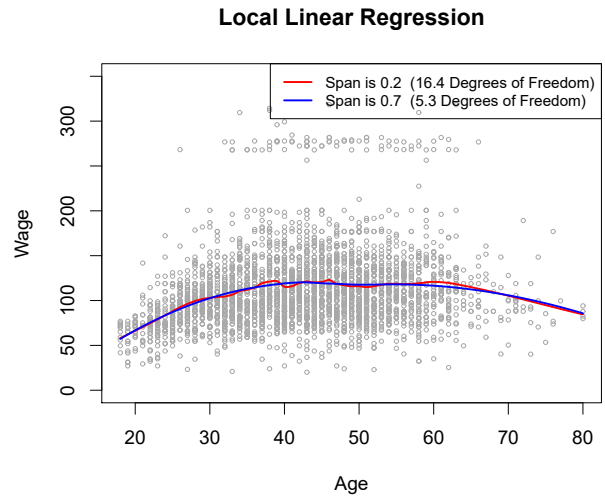


**FIGURE 7.10.** *Local linear fits to the* `Wage` *data. The span specifies the fraction of the data used to compute the fit at each target point.*

## 7.7 Generalized Additive Models

In Sections 7.1–7.6, we present a number of approaches for flexibly predicting a response $Y$ on the basis of a single predictor $X$. These approaches can be seen as extensions of simple linear regression. Here we explore the problem of flexibly predicting $Y$ on the basis of several predictors, $X_1, \ldots, X_p$. This amounts to an extension of multiple linear regression.

*Generalized additive models* (GAMs) provide a general framework for extending a standard linear model by allowing non-linear functions of each of the variables, while maintaining *additivity*. Just like linear models, GAMs can be applied with both quantitative and qualitative responses. We first

generalized additive model

additivity

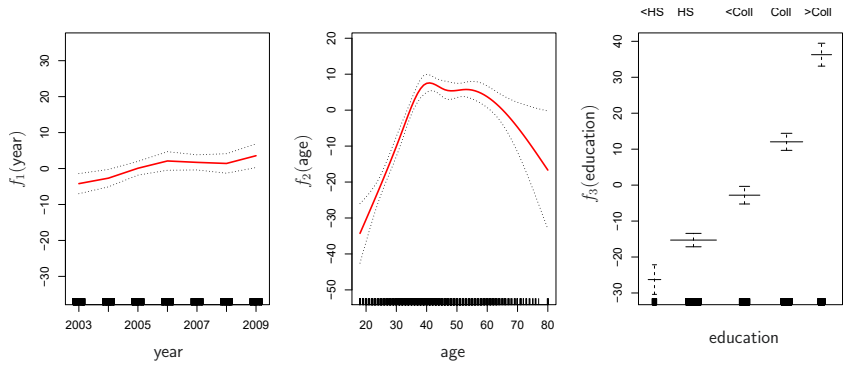


**FIGURE 7.11.** *For the* Wage *data, plots of the relationship between each feature and the response,* wage*, in the fitted model (7.16). Each plot displays the fitted function and pointwise standard errors. The first two functions are natural splines in* year *and* age*, with four and five degrees of freedom, respectively. The third function is a step function, fit to the qualitative variable* education*.*

examine GAMs for a quantitative response in Section 7.7.1, and then for a qualitative response in Section 7.7.2.

### 7.7.1 GAMs for Regression Problems

A natural way to extend the multiple linear regression model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

in order to allow for non-linear relationships between each feature and the response is to replace each linear component $\beta_j x_{ij}$ with a (smooth) non-linear function $f_j(x_{ij})$. We would then write the model as

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i. \end{aligned} \tag{7.15}$$

This is an example of a GAM. It is called an *additive* model because we calculate a separate $f_j$ for each $X_j$, and then add together all of their contributions.

In Sections 7.1–7.6, we discuss many methods for fitting functions to a single variable. The beauty of GAMs is that we can use these methods as building blocks for fitting an additive model. In fact, for most of the methods that we have seen so far in this chapter, this can be done fairly trivially. Take, for example, natural splines, and consider the task of fitting the model

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon \tag{7.16}$$

on the Wage data. Here year and age are quantitative variables, while the variable education is qualitative with five levels: <HS, HS, <Coll, Coll, >Coll, referring to the amount of high school or college education that an individual has completed. We fit the first two functions using natural splines. We

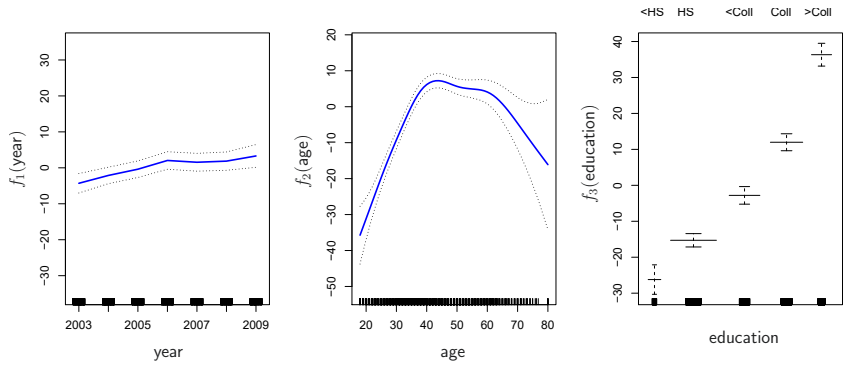


**FIGURE 7.12.** *Details are as in Figure 7.11, but now $f_1$ and $f_2$ are smoothing splines with four and five degrees of freedom, respectively.*

fit the third function using a separate constant for each level, via the usual dummy variable approach of Section 3.3.1.

Figure 7.11 shows the results of fitting the model (7.16) using least squares. This is easy to do, since as discussed in Section 7.4, natural splines can be constructed using an appropriately chosen set of basis functions. Hence the entire model is just a big regression onto spline basis variables and dummy variables, all packed into one big regression matrix.

Figure 7.11 can be easily interpreted. The left-hand panel indicates that holding `age` and `education` fixed, `wage` tends to increase slightly with `year`; this may be due to inflation. The center panel indicates that holding `education` and `year` fixed, `wage` tends to be highest for intermediate values of `age`, and lowest for the very young and very old. The right-hand panel indicates that holding `year` and `age` fixed, `wage` tends to increase with `education`: the more educated a person is, the higher their salary, on average. All of these findings are intuitive.

Figure 7.12 shows a similar triple of plots, but this time $f_1$ and $f_2$ are smoothing splines with four and five degrees of freedom, respectively. Fitting a GAM with a smoothing spline is not quite as simple as fitting a GAM with a natural spline, since in the case of smoothing splines, least squares cannot be used. However, standard software such as the `Python` package `pygam` can be used to fit GAMs using smoothing splines, via an approach known as *backfitting*. This method fits a model involving multiple predictors by repeatedly updating the fit for each predictor in turn, holding the others fixed. The beauty of this approach is that each time we update a function, we simply apply the fitting method for that variable to a *partial residual*.[6]

`pygam`

backfitting

The fitted functions in Figures 7.11 and 7.12 look rather similar. In most situations, the differences in the GAMs obtained using smoothing splines versus natural splines are small.

[6] A partial residual for $X_3$, for example, has the form $r_i = y_i - f_1(x_{i1}) - f_2(x_{i2})$. If we know $f_1$ and $f_2$, then we can fit $f_3$ by treating this residual as a response in a non-linear regression on $X_3$.

We do not have to use splines as the building blocks for GAMs: we can just as well use local regression, polynomial regression, or any combination of the approaches seen earlier in this chapter in order to create a GAM. GAMs are investigated in further detail in the lab at the end of this chapter.

### Pros and Cons of GAMs

Before we move on, let us summarize the advantages and limitations of a GAM.

- ▲ GAMs allow us to fit a non-linear $f_j$ to each $X_j$, so that we can automatically model non-linear relationships that standard linear regression will miss. This means that we do not need to manually try out many different transformations on each variable individually.
- ▲ The non-linear fits can potentially make more accurate predictions for the response $Y$.
- ▲ Because the model is additive, we can examine the effect of each $X_j$ on $Y$ individually while holding all of the other variables fixed.
- ▲ The smoothness of the function $f_j$ for the variable $X_j$ can be summarized via degrees of freedom.
- ◆ The main limitation of GAMs is that the model is restricted to be additive. With many variables, important interactions can be missed. However, as with linear regression, we can manually add interaction terms to the GAM model by including additional predictors of the form $X_j \times X_k$. In addition we can add low-dimensional interaction functions of the form $f_{jk}(X_j, X_k)$ into the model; such terms can be fit using two-dimensional smoothers such as local regression, or two-dimensional splines (not covered here).

For fully general models, we have to look for even more flexible approaches such as random forests and boosting, described in Chapter 8. GAMs provide a useful compromise between linear and fully nonparametric models.

## *7.7.2 GAMs for Classification Problems*

GAMs can also be used in situations where $Y$ is qualitative. For simplicity, here we assume $Y$ takes on values 0 or 1, and let $p(X) = \Pr(Y = 1|X)$ be the conditional probability (given the predictors) that the response equals one. Recall the logistic regression model (4.6):

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p. \tag{7.17}$$

The left-hand side is the log of the odds of $P(Y = 1|X)$ versus $P(Y = 0|X)$, which (7.17) represents as a linear function of the predictors. A natural way to extend (7.17) to allow for non-linear relationships is to use the model

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p). \tag{7.18}$$

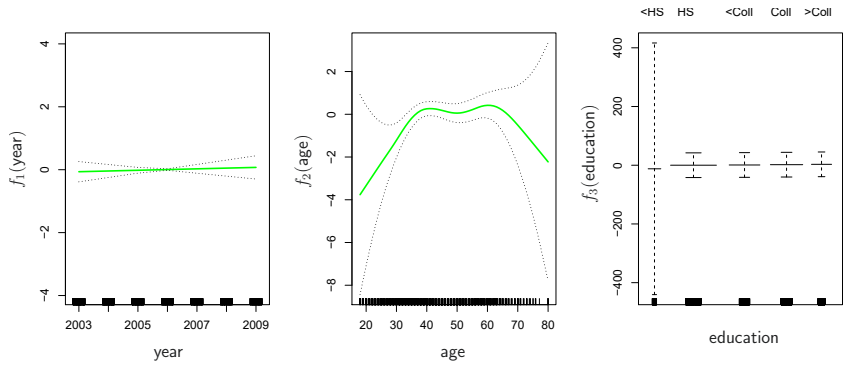


**FIGURE 7.13.** *For the* `Wage` *data, the logistic regression GAM given in (7.19) is fit to the binary response* `I(wage>250)`*. Each plot displays the fitted function and pointwise standard errors. The first function is linear in* `year`*, the second function a smoothing spline with five degrees of freedom in* `age`*, and the third a step function for* `education`*. There are very wide standard errors for the first level* `<HS` *of* `education`*.*

Equation 7.18 is a logistic regression GAM. It has all the same pros and cons as discussed in the previous section for quantitative responses.

We fit a GAM to the `Wage` data in order to predict the probability that an individual's income exceeds \$250,000 per year. The GAM that we fit takes the form

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 \times \texttt{year} + f_2(\texttt{age}) + f_3(\texttt{education}), \quad (7.19)$$

where

$$p(X) = \Pr(\texttt{wage} > 250 | \texttt{year}, \texttt{age}, \texttt{education}).$$

Once again $f_2$ is fit using a smoothing spline with five degrees of freedom, and $f_3$ is fit as a step function, by creating dummy variables for each of the levels of education. The resulting fit is shown in Figure 7.13. The last panel looks suspicious, with very wide confidence intervals for level `<HS`. In fact, no response values equal one for that category: no individuals with less than a high school education make more than \$250,000 per year. Hence we refit the GAM, excluding the individuals with less than a high school education. The resulting model is shown in Figure 7.14. As in Figures 7.11 and 7.12, all three panels have similar vertical scales. This allows us to visually assess the relative contributions of each of the variables. We observe that `age` and `education` have a much larger effect than `year` on the probability of being a high earner.

## 7.8 Lab: Non-Linear Modeling

In this lab, we demonstrate some of the nonlinear models discussed in this chapter. We use the `Wage` data as a running example, and show that many of the complex non-linear fitting procedures discussed can easily be implemented in `Python`.