21

Nonparametric Regression and Splines

21.1 Introduction

As discussed in Chap. 9, regression analysis estimates the conditional expectation of a response given predictor variables. The conditional expectation is called the regression function and is the best predictor of the response based upon the predictor variables, because it minimizes the expected squared prediction error.

There are three types of regression, linear, nonlinear parametric, and nonparametric. *Linear regression* assumes that the regression function is a linear function of the parameters and estimates the intercept and slopes (regression coefficients). *Nonlinear parametric regression*, which was discussed in Sect. 11.2, does not assume linearity but does assume that the regression function is of a *known* parametric form, for example, the Nelson-Siegel model. In this chapter, we study *nonparametric regression*, where the form of the regression function is also nonlinear but, unlike nonlinear parametric regression, not specified by a model but rather determined from the data. Nonparametric regression is used when we know, or suspect, that the regression function is curved, but we do not have a model for the curve.

There are many techniques for nonparametric regression, but local polynomial regression and splines are the most widely used, and only these will be discussed here. Local polynomial regression and splines generally work well and, since they usually give similar estimates, it is difficult to recommend one over the over. Local polynomial estimation might be somewhat simpler to understand. Splines are used in many areas of mathematics, such as, for interpolation, and so it is worthwhile to be familiar with them. Also, splines are useful as components in complex models. The `R` lab at the end of this chapter gives an example.

D. Ruppert, D.S. Matteson, *Statistics and Data Analysis for Financial Engineering*, Springer Texts in Statistics,
DOI 10.1007/978-1-4939-2614-5_21

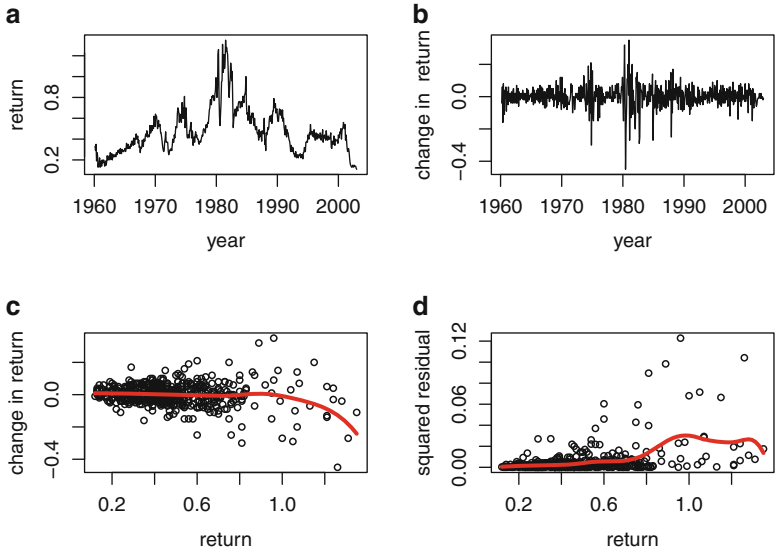

Fig. 21.1. *Risk-free monthly returns. The returns are* $1/12$*th the yearly rate. (a) Time series plot of the returns. (b) Time series plot of the changes in the returns. (c) Plot of changes in returns against lagged returns with a local linear estimate of the drift. (d) Plot of squared residuals against lagged returns with a local linear estimate of the squared diffusion coefficient.*

Models for the evolution of short-term interest rates are important in finance, for example, because they are needed for the pricing of interest rate derivatives. Figure 21.1 contains plots of the monthly risk-free returns[1] in the `Capm` data set in `R`'s `Ecdat` package. This data set has been used for various purposes in several previous chapters. Here we will use it to illustrate nonparametric regression. Panels (a) and (b) are time series plots of the returns and the changes in the returns.

A common model for changes in short-term interest rates is

$$\Delta r_t = \mu(r_{t-1}) + \sigma(r_{t-1})\epsilon_t, \tag{21.1}$$

where r_t is the rate at time t, $\Delta r_t = r_t - r_{t-1}$, $\mu(\cdot)$ is the drift function, $\sigma(\cdot)$ is the volatility function, also called the diffusion function, and ϵ_t is $N(0,1)$ noise. Many different parametric models have been proposed for $\mu(\cdot)$ and $\sigma(\cdot)$, for example, by Merton (1973), Vasicek (1977), Cox, Ingersoll, and Ross (1985), Yau and Kohn (2003), and Chan et al. (1992). The simplest model, due to Merton (1973), is that $\mu(\cdot)$ and $\sigma(\cdot)$ are constant. Chan et al. (1992) assume that $\mu(r) = \beta(r - \alpha)$ and $\sigma(r) = \theta r^\gamma$, where $\alpha > 0$, $\beta < 0$, $\theta > 0$, and γ are unknown parameters—this process reverts to a mean equal to α. Chan et al.'s model was used as an example of nonlinear regression in Sect. 11.12.1.

[1] The risk-free rate is called the risk-free return in the `Capm` package.

The approach of Yau and Kohn (2003) that is used here is to model both $\mu(\cdot)$ and $\sigma(\cdot)$ nonparametrically. Doing this allows one to check which parametric models, if any, fit the data and to have a nonparametric alternative if none of the parametric models fits well.

The solid red curves in Fig. 21.1c and d are estimates of $\mu(\cdot)$ and $\sigma^2(\cdot)$ by a nonparametric regression method *local linear regression*, a special case of *local polynomial regression*. By (21.1), $E(\Delta r_t) = \mu(r_{t-1})$ and $\text{Var}(\Delta r_t) = \sigma^2(r_{t-1})$, so $\widehat{\mu}(\cdot)$ is obtained by regressing Δr_t on r_{t-1} and $\widehat{\sigma}^2(\cdot)$ by regressing $\{\Delta r_t - \widehat{\mu}(r_{t-1})\}^2$ on r_{t-1}. The latter is an example of estimating a conditional variance; see Sect. 14.2.

The code to produce Fig. 21.1 is below. The local linear regression estimates were produced by the function `locpoly()` in the `KernSmooth` package. This function computes the fitted function on a grid of 401 equally spaced points; this is sufficient for plotting the fitted function as in lines 21 and 24 but not for computing the residuals. Instead, to compute squared residuals at line 11, the `spline()` function is used at line 10 to interpolate the fit from the 401-point grid to the values of the explanatory variable.

```
1  library(Ecdat)
2  library(KernSmooth)
3  data(Capm)
4  attach(Capm)
5  n = length(rf)
6  year = seq(1960.125, 2003, length = n)
7  diffrf = diff(Capm$rf)
8  rf_lag = rf[1:(n-1)]
9  ll_mu <- locpoly(rf_lag, diffrf, bandwidth = dpill(rf_lag, diffrf))
10 muhat = spline(ll_mu$x, ll_mu$y, xout = rf_lag)$y
11 epsilon_sqr = (diffrf-muhat)^2
12 ll_sig <- locpoly(rf_lag, epsilon_sqr,
13    bandwidth = dpill(rf_lag, epsilon_sqr) )
14 pdf("riskfree01.pdf", width = 6, height = 5)
15 par(mfrow=c(2, 2))
16 plot(year, rf, ylab = "return", main = "(a)", type = "l" )
17 plot(year[2:n], diffrf, ylab = "change in  return", main = "(b)",
18    type = "l", xlab = "year")
19 plot(rf_lag, diffrf, ylab = "change in return", xlab = "return",
20    main="(c)",type="p",cex=.7)
21 lines(ll_mu$x, ll_mu$y, lwd = 4, col = "red")
22 plot(rf_lag, (diffrf - muhat)^2, xlab = "return",
23    ylab = "squared residual", main = "(d)", cex = 0.7)
24 lines(ll_sig$x, ll_sig$y, lwd = 4, col = "red")
25 graphics.off()
```

21.2 Local Polynomial Regression

Local polynomial regression is based on the principle that a smooth function can be approximated locally by a low-degree polynomial. Suppose we have a sample (X_i, Y_i), $i = 1, \ldots, n$, and $E(Y|X = x) = \mu(x)$ for a smooth function μ. The function μ will be estimated on a grid of x-values, $x_1, \ldots, x_M$. These could, but need not, be the same values $X_1, \ldots, X_n$, as where we observe Y.

The estimation is done one point at a time on the grid $x_1, \ldots, x_M$. To estimate μ at x_ℓ, one fits a pth-degree polynomial using only (X_i, Y_i) with X_i near x_ℓ. This is done using weights determined by a kernel function K. K is a probability density function symmetric about 0 and such that $K(x)$ decreases as $|x|$ increases, for instance, a normal density with mean 0. We have seen kernels used for density estimation in Sect. 4.2.

The regression function at x_ℓ is estimated by kernel-weighted least squares, which minimizes

$$\sum_{i=1}^{n} \left[Y_i - \{\beta_0 + \beta_1(X_i - x_\ell) + \cdots + \beta_p(X_i - x_\ell)^p\} \right]^2 K\{(X_i - x_\ell)/h\} \quad (21.2)$$

and then $\widehat{\mu}(x_\ell) = \widehat{\beta}_0$ since the regression model $\beta_0 + \beta_1(x - x_\ell) + \cdots + \beta_p(x - x_\ell)^p$ equals β_0 at $x = x_\ell$. The weights $K\{(X_i - x_\ell)/h\}$ decrease as $|X_i - x_\ell|$ increases, so only the data near x_ℓ are used. The parameter h is called the bandwidth and determines how much data are used for estimation; the larger the value of h, the more data used.

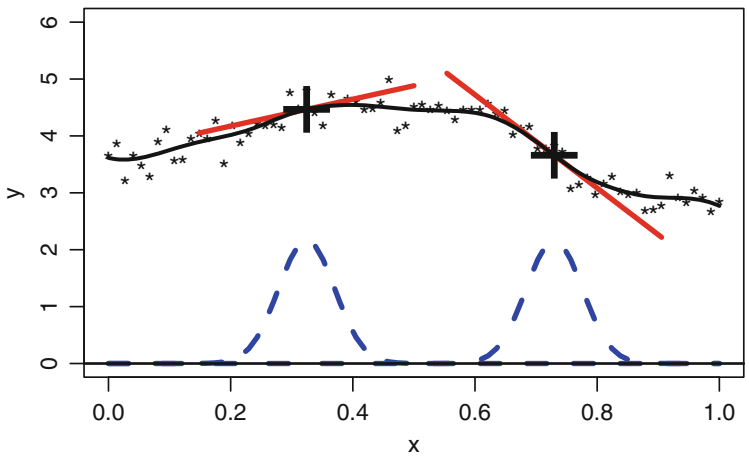

Fig. 21.2. *Local linear fit (solid curve) to 75 data points (asterisks) with bandwidth chosen by the direct plug-in method. The regression function μ is estimated at each of the 75 points and the estimates are connected to create the solid curve. Estimation at $x_{25} = 0.32$ and $x_{55} = 0.72$ is illustrated by the kernels (dashed curves), the linear fits (solid lines), and the fitted points (large +).*

Local linear estimation, where $p = 1$, is illustrated in Fig. 21.2. The kernel functions are shown as blue dashed curves at two points, $x_{25} = 0.32$ and $x_{75} = 0.72$. Above each kernel, the local linear fit is shown as a red line and the large "+" is placed at $\{x, \widehat{\mu}(x)\}$. The black curve $\widehat{\mu}$ is obtained by finding local fits on a grid of 75 x_ℓ-values and plotting $\{x_\ell, \widehat{\mu}(x_\ell)\}$ for all x_ℓ on this grid. For example, the curve in Fig. 21.2 used the `R` function `locpoly()` in `R`'s `KernSmooth` package and has a grid of 401 equally spaced x-values (the default). Often the grid is simply the observed X-values, $X_1, \ldots, X_n$.

The bandwidth h is called a "smoothing parameter" because it determines the smoothness of $\widehat{\mu}$. A larger value of h gives a smoother curve. The choice of h is important. If h is too large, then the polynomial approximation may be poor and the estimate of $\mu(x)$ will be badly biased. Conversely, if h is too small, then too few data are used and the estimate of μ will be too variable. A good choice of the bandwidth minimizes the mean squared error of the estimator, which is the variance plus the squared bias. Both the squared bias and variance of the estimator are unknown and must be estimated, or at least their sum must be estimated. Automatic bandwidth selection, which either directly or indirectly estimates and attempts to minimize the mean-squared error, has been an area of intense research and a number of data-based bandwidth selectors are available. The curve in Fig. 21.2 used the bandwidth chosen by the popular direct plug-in (dpi) bandwidth selector of Ruppert, Sheather and Wand (1995). The dpi selector estimates the mean integrated squared error (MISE) of $\widehat{\mu}$, which is

$$E\left[\int_{\min(X_i)}^{\max(X_i)} \{\mu(x) - \widehat{\mu}(x)\}^2 \, dx\right], \tag{21.3}$$

and finds the bandwidth that minimizes the estimated MISE.

Nonparametric regression estimators are also called *smoothers* because they smooth out the noise in the data. Using a bandwidth that is too small causes *overfitting*, which is *undersmoothing*. Conversely, a bandwidth that is too large will result in *underfitting*, which is *oversmoothing*—see Sect. 4.2 for further discussion of under- and oversmoothing in the context of kernel density estimation.

Figure 21.3 illustrates the effect of varying the bandwidth. The solid black curve uses the dpi bandwidth, the dashed red curve uses three times the dpi bandwidth, and the dotted-and-dashed blue curve uses one-third the dpi bandwidth. The dashed red curve is too smooth to follow the data closely, that is, it underfits, while the dotted-and-dashed blue curve is wiggly because it is tracking random noise in the data, that is, it overfits. In this example, the data were simulated, so the true regression function, $\mu(x) = 3.6 + 0.1x + \sin(5\,x^{1.5})$, is known and it is possible to calculate the average squared error, $\sum_{i=1}^{n}\{\widehat{\mu}(X_i) - \mu(X_i)\}^2$, for each bandwidth. The average squared errors are 1.34 and 2.27 times larger using 3*dpi and dpi/3, respectively, compared to using dpi.

Besides the dpi bandwidth selector, the bandwidth can also be chosen by minimizing either the AIC or GCV (generalized cross-validation) criterion. The definition of AIC for a parametric model uses the number of parameters in the model, but local polynomial estimation is not parametric, so one cannot count parameters. Nonetheless, it is possible to define the "effective number of parameters" and this is done in Sect. 21.3.1. GCV is defined in Sect. 21.3.2.

Example 21.1. Local polynomial estimation of forward rates.

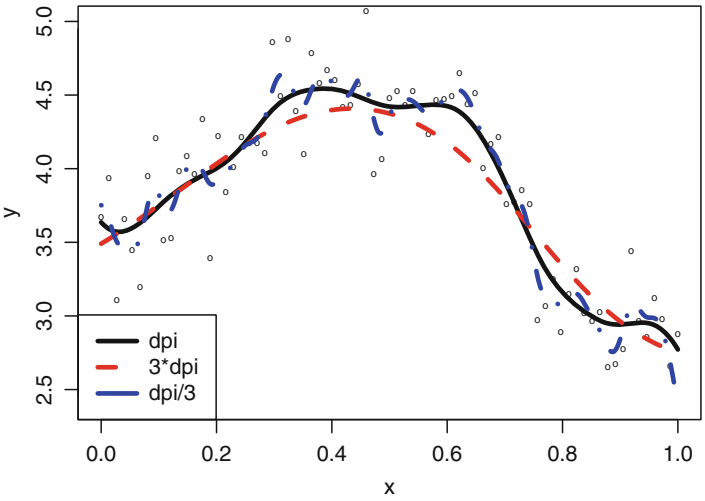

Fig. 21.3. *Local linear estimators with three bandwidths: dpi (direct plug-in), which gives an appropriate amount of smoothing; three times the dpi, which oversmooths (underfits); and one-third the dpi, which undersmooths (overfits). Simulated data.*

The `R` code in this section computes two estimates of the forward rate function, a parametric estimate based on the Nelson-Siegel model and a nonparametric estimates using local polynomial smoothing. The program is split into several chucks of code with discussion between the chucks.

Line 1 reads in prices of STRIPS on Dec 31, 1995. This data set was used in Sect. 11.3. There are 29 1/4 years of quarterly maturities T, for a total of 117 (= 29.25 * 4) prices.

The price of a zero-coupon bond as a percentage of par is price $= 100 \exp\left\{-\int_0^T f(s)ds\right\}$ so that $-\log(\text{price}) = \int_0^T f(s)ds - \log(100)$. Thus, the integrated forward rate, called `Int_F`, is defined on line 7 to be $-\log(\text{price}) + \log(100)$. Lines 4–7 use the `order()` function to order the maturities `T` from smallest to largest and to order the prices accordingly. Ordering the data by `T` is needed when the plotted points are connected by lines. If they were not ordered by `T`, then the plot could look like a spider web.

```
dat = read.table("strips_dec95.txt", header = T)
T = dat$T
n = length(T)
ord = order(T)
T = T[ord]
price = dat$price[ord]
Int_F = - log(price) + log(100)
```

In lines 8–9, the function `locfit()` in the `locfit` package estimates the first derivative of `Int_F` by local cubic fitting to produce an estimate of the forward rate (since `Int_F` estimates the integrated forward rate). Note that `deriv = 1` specifies estimation of the first derivative and `deg = 3` specifies using cubic polynomial fitting. The function `locfit()` is similar to `locpoly()` used in Sect. 21.1 to create the curves in Fig. 21.1. We could have used `locpoly()` again here but wanted to illustrate both local polynomial regression functions.

```
library(locfit)
fit_loc_Int_F = locfit(Int_F ~ T, deriv = 1, deg = 3)
```

The function `Nelson-Siegel()` in lines 10–18 returns a list containing the forward rate, called `f`, and the integrated forward rate, called `int_f`. Lines 19–24 estimate the parameters of the Nelson-Siegel model by fitting the Nelson-Siegel integrated forward rate to `Int_F`. On line 21, `Yhat` is the integrated forward rate because "`[[2]]`" is the second element of the list that is output by `NelsonSiegel()`.

```
NelsonSiegel = function(theta){
   ####  f = forward rate and int_f = intergrated forward rate  ####
   f = theta[1] + (theta[2] + theta[3] * T) * exp(-theta[4] * T)
   int_f = theta[1] * T - theta[2] / theta[4]
   * (exp(-theta[4] * T) - 1) -
   theta[3] * (T * exp(-theta[4] * T) / theta[4] +
   (exp(-theta[4] * T) - 1) / theta[4]^2)
    list("f" = f, "inf_t" = int_f)
}
fit_NS = optim(c(0.05, 0.001, 0.001, 0.08),
fn = function(theta){
   Yhat = NelsonSiegel(theta)[[2]]
   sum((Int_F - Yhat)^2)},
   control = list(maxit=30000, reltol = 1e-10))
NS_yhat = NelsonSiegel(fit_NS$par)[[1]]
```

Figure 21.4 is produced by lines 25–36. Notice that the Nelson-Siegel fit (in blue) tends to overestimate the forward rate when $5 < T < 15$ and $T > 25$ and to underestimate when $T < 3$ and $15 < T < 25$. In comparison, the local polynomial estimates show no bias.

```
pdf("strips02.pdf",width=6,height=5)
par(mfrow=c(1,1))
```

```
plot(fit_loc_Int_F, ylim = c(.025,.075), ylab = "Forward rate",
   col = "red", lwd = 3)
lines(fit_loc_Int_F, col = "red", lwd=3) # to widen to linewidth = 3
lines(T, NS_yhat, col = "blue", lwd = 3, lty = 4)
points(T[2:n], diff(Int_F) / diff(T))
legend("bottomleft", c("local cubic",
   "Nelson-Siegel", "Empirical"),
   lty=c(1, 4,NA),pch=c(NA, NA, "o"),
   col=c("red", "blue", "black"), lwd = c(3, 3, NA))
graphics.off()
```

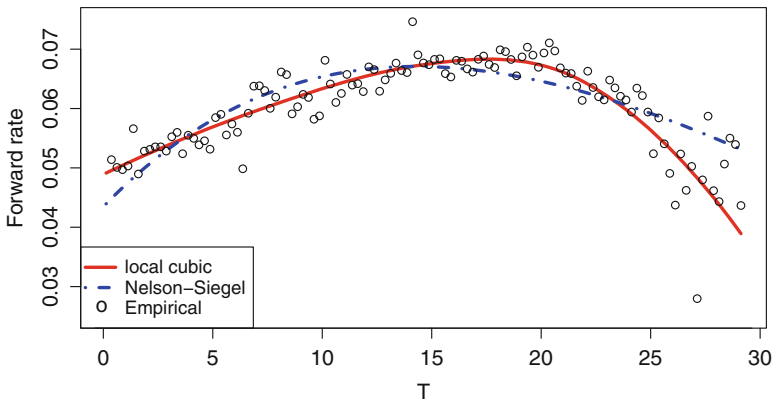

Fig. 21.4. Local cubic (nonparametric), Nelson-Siegel (parametric), and empirical estimates of the forward rate curve. Notice that the nonparametric estimates are much closer than the parametric estimates to the empirical estimates.

□

21.2.1 Lowess and Loess

Loess and its earlier version lowess are local polynomial smoothers with spatially varying bandwidths controlled by a parameter called *span*. Span is the fraction of the data used for estimation at each point. The bandwidth, call it $h(x, \text{span})$, for estimation at a point x is adjusted so that whenever span $\leq$ 1 then $K\{(X_i - x)/h(x, \text{span})\}$ is nonzero for span $\times$ 100 % of the X_i.

If span = 1, then all of the data are used for estimation at each point, but the data farthest from X_i get small weights. Because of these small weights, for small data sets, a lowess (or loess) smooth with a span of 1 might not be smooth enough. To solve this problem, span is defined for values greater than 1 by

$$h(x, \text{span}) = \text{span} \times h(x, 1).$$

As span increases beyond 1, the weights $K\{(X_i - x)/h(x, \text{span})\}$ become more and more equal. As span $\to \infty$, the weights converge to a constant, $K(0)$, and the lowess (or loess) fit converges to a polynomial regression fit.

21.3 Linear Smoothers

Local polynomial regression as well as penalized spline regression—to be covered soon—are examples of linear smoothers. A linear smoother has an $n \times n$ smoother matrix $\boldsymbol{H}$, which does not depend on $\boldsymbol{Y}$, such that

$$\widehat{\boldsymbol{Y}} = \boldsymbol{H}\boldsymbol{Y}, \tag{21.4}$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\mathsf{T}$ is the vector of responses and $\widehat{\boldsymbol{Y}} = (\widehat{Y}_1, \ldots, \widehat{Y}_n)^\mathsf{T}$ is the vector of fitted values. Equation (21.4) can be written as

$$\widehat{Y}_i = \sum_{j=1}^{n} H_{ij} Y_j, \quad i = 1, \ldots, n. \tag{21.5}$$

The smoother matrix will depend on a smoothing parameter, which for local polynomial regression is the bandwidth. We will let λ denote the smoothing parameter and denote the smoother matrix by $\boldsymbol{H}(\lambda)$. The smoother matrix is an analog of the hat matrix of linear regression and is, itself, often called a hat matrix.

21.3.1 The Smoother Matrix and the Effective Degrees of Freedom

In a parametric model, the number of parameters quantifies the ability of the model to fit the data. In nonparametric estimation, the potential to fit (and overfit) can be quantified by the *effective number of parameters* or the *effective degrees of freedom of the fit*. Conceptually, the effective number of parameters is similar to the Bayesian p_D in Sect. 20.7.6.

By (21.5), the hat diagonal $H(\lambda)_{ii}$ gives the *leverage* or *self-influence* of the Y_i since it is the weight given to Y_i when calculating $\widehat{Y}_i$. A large value of $H(\lambda)_{ii}$ means a high potential for overfitting. The effective number of parameters is the sum of the leverages:

$$p_{\text{eff}} = \sum_{i=1}^{n} H(\lambda)_{ii} = \text{tr}\{\boldsymbol{H}(\lambda)\}. \tag{21.6}$$

If p_{eff} is too small (too large), then the data are underfit (overfit).

The residual mean sum of squares is

$$\sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 = \|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|^2 = \|\{\boldsymbol{I} - \boldsymbol{H}(\lambda)\}\boldsymbol{Y}\|^2, \tag{21.7}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix. The noise variance is estimated by

$$\widehat{\sigma}(\lambda)^2 = \frac{\|\{\boldsymbol{I} - \boldsymbol{H}(\lambda)\}\boldsymbol{Y}\|^2}{n - p_{\text{eff}}}, \tag{21.8}$$

which is a direct analog of (9.16).

21.3.2 AIC, CV, and GCV

For linear regression models, AIC is

$$\text{AIC} = n \log(\widehat{\sigma}^2) + 2(1 + p),$$

where $1 + p$ is the number of parameters (intercept plus p slopes). For a linear smoother, AIC uses p_{eff} in place of $p + 1$, so that

$$\text{AIC}(\lambda) = n \log\{\widehat{\sigma}^2(\lambda)\} + 2\, p_{\text{eff}}.$$

We can then select λ by minimizing AIC.

The cross-validation or CV statistic is

$$\text{CV}(\lambda) = \sum_{i=1}^{n} \{Y_i - \widehat{Y}_{-i}(\lambda)\}^2$$

where, to prevent overfitting, $\widehat{Y}_{-1}(\lambda)$ is the ith fitted value computed with the ith observation deleted. One, of course, should choose a λ with a small value of $\text{CV}(\lambda)$.

The generalized cross-validation statistic (GCV) is

$$\text{GCV}(\lambda) = \frac{\|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}(\lambda)\|^2}{(n - p_{\text{eff}})^2}. \tag{21.9}$$

$\text{GCV}(\lambda)$ can be computed more quickly than $\text{CV}(\lambda)$ and $\text{GCV}(\lambda)$ is a good approximation to $\text{CV}(\lambda)/n^2$ so minimizing GCV is another way to choose λ.

AIC and GCV can both be computed very quickly and usually give essentially the same amount of smoothing. In fact, it has been shown theoretically that both criteria should give similar estimates. Therefore, it does not matter much which is used, but GCV is more commonly used than AIC in nonparametric regression.

21.4 Polynomial Splines

The use of polynomial splines in nonparametric regression, as well as many other areas of mathematics, is based on the same principle as local polynomial regression—a smooth function can be accurately approximated locally by a low-degree polynomial. A pth-degree polynomial spline is constructed by piecing together pth-degree polynomials, so that they join together at specified locations called *knots*. The polynomials are spliced together, so that the spline has $p - 1$ continuous derivatives. The pth derivative of the spline is constant between knots and can jump at the knots.

21.4.1 Linear Splines with One Knot

We start simple, a linear spline with one knot. Figure 21.5a illustrates such a spline. This spline is defined as

$$f(x) = \begin{cases} 0.5 + 0.2x, & x < 2, \\ -0.5 + 0.7x, & x \geq 2. \end{cases}$$

Because $0.5 + 0.2x = 0.9 = -0.5 + 0.7x$ when $x = 2$, the two linear components are equal at the point $x = 2$, so that they join together there.

The point $x = 2$ where the spline switches from one linear function to the other is called a *knot*. A linear spline with a knot at the point t can be constructed as follows. The spline is defined to be $s(x) = a + bx$ for $x < t$ and $s(x) = c + dx$ for $x > t$. The parameters a, b, c, and d can be chosen arbitrarily except that they must satisfy the equality constraint

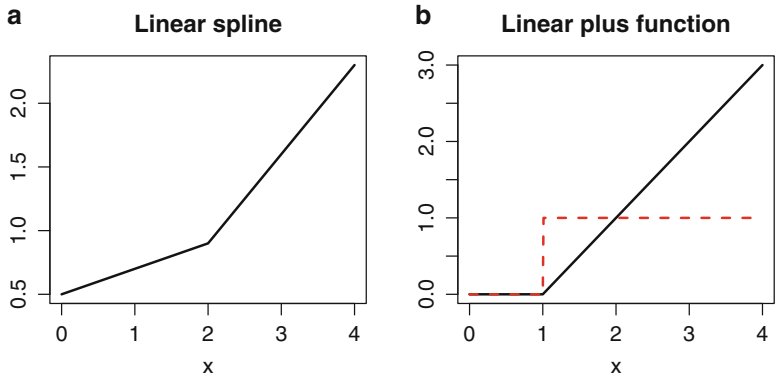

Fig. 21.5. *(a) Example of a linear spline with a knot at* 2*. (b) The linear plus function* $(x - 1)_+$ *with a knot at* 1 *(black) and its first derivative (red).*

$$a + bt = c + dt, \tag{21.10}$$

which assures us that the two lines join together at $x = t$. Solving for c in (21.10), we get $c = a + (b - d)t$. Substituting this expression for c into the definition of $s(x)$ and doing some rearranging, we have

$$s(x) = \begin{cases} a + bx, & x < t, \\ a + bx + (d - b)(x - t), & x \geq t. \end{cases} \tag{21.11}$$

Recall the definition that for any number y,

$$(y)_+ = \begin{cases} 0, & y < 0, \\ y, & y \geq 0. \end{cases}$$

By this definition,

$$(x-t)_+ = \begin{cases} 0, & x < t, \\ x-t, & x \geq t. \end{cases}$$

We call $(x-t)_+$ a linear *plus function* with a knot at t. It is also called a truncated line, though we will stick with "plus function." The spline $s(x)$ in (21.11) can be written using this plus function:

$$s(x) = a + bx + (d-b)(x-t)_+.$$

The plus function simplifies the problem of keeping the spline continuous at t. Figure 21.5b illustrates a linear plus function with a knot at 1 and its first derivative. Notice that

$$\frac{d}{dx}(x-t)_+ = \begin{cases} 0, & x < t, \\ 1, & x > t. \end{cases}$$

21.4.2 Linear Splines with Many Knots

Plus functions are very convenient when defining linear splines with more than one knot, because plus functions automatically join the component linear functions together so that the spline is continuous. For example, suppose we want a linear spline to have K knots, $t_1 < \cdots < t_K$, for the spline to equal $s(x) = \beta_0 + \beta_1 x$ for $x < t_1$, and for the first derivative of the spline to jump by the amount b_k at knot t_k, for $k = 1, \ldots, K$. Then the spline can be constructed from linear plus functions, one for each knot:

$$s(x) = \beta_0 + \beta_1 x + b_1(x-t_1)_+ + b_2(x-t_2)_+ + \cdots + b_K(x-t_K)_+.$$

Because the plus functions are continuous, the spline is the sum of continuous functions and is therefore continuous itself.

21.4.3 Quadratic Splines

A linear spline is continuous but has "kinks" at its knots, where its first derivative jumps. If we want a function without these kinks, we cannot use a linear spline. A quadratic spline is a function obtained by piecing together quadratic polynomials. More precisely, $s(x)$ is a quadratic spline with knots $t_1 < \cdots < t_K$ if $s(x)$ equals one quadratic polynomial to the left of t_1 and equals a second quadratic polynomial between t_1 and t_2, and so on. The quadratic polynomials are pieced together, so that the spline is continuous and, to guarantee no kinks, its first derivative is also continuous. Figure 21.6a shows a quadratic spline with a knot at 1. Notice that the function does not have a kink at the knot but changes from convex to concave there.

As with linear splines, continuity can be enforced by using plus functions. Define the quadratic plus function

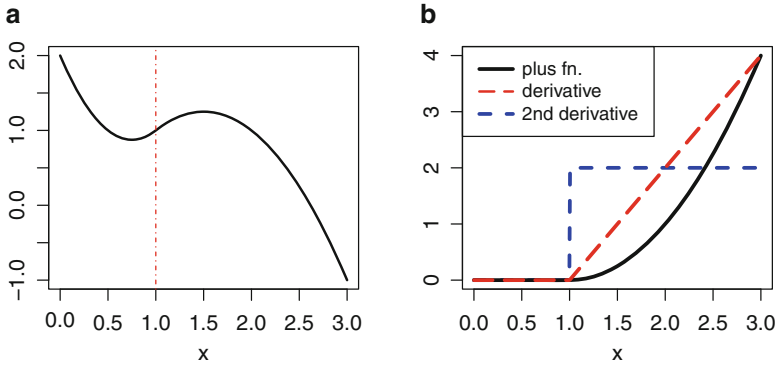

Fig. 21.6. *(a) Quadratic spline with a knot at* 1*. The dotted red vertical line marks the knot's location. (b) The quadratic plus function* $(x-1)_+^2$ *with a knot at* 1 *(black) and its first (red) and second (blue) derivatives.*

$$(x-t)_+^2 = \begin{cases} 0, & x < t, \\ (x-t)^2, & x \geq t. \end{cases}$$

Notice that $(x-t)_+^2$ equals $\{(x-t)_+\}^2$, not $\{(x-t)^2\}_+ = (x-t)^2$.

Figure 21.6b shows a quadratic plus function and its first and second derivatives. One can see that for $x > t$

$$\frac{d}{dx}(x-t)_+^2 = 2(x-t)_+$$

and

$$\frac{d^2}{dx^2}(x-t)_+^2 = 2(x-t)_+^0,$$

where $(x-t)_+^0 = \{(x-t)_+\}^0$, so that $(x-t)_+^0$ is the 0th-degree plus function

$$(x-t)_+^0 = \begin{cases} 0, & x < t, \\ 1, & x \geq t. \end{cases}$$

Therefore, the second derivative of $(x-t)_+^2$ jumps from 0 to 2 at the knot t.

A quadratic spline with knots $t_1 < \cdots < t_K$ can be written as

$$s(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + b_1(x-t_1)_+^2 + b_2(x-t_2)_+^2 + \cdots + b_K(x-t_K)_+^2.$$

The second derivative of s jumps by the amount $2b_k$ at knot t_k for $k = 1, \ldots, K$.

21.4.4 *p*th Degree Splines

The way to define a general pth-degree spline with knots $t_1 < \cdots < t_K$ should now be obvious:

$$s(x) = \beta_0 + \beta_1 x + \cdots + \beta_p x^p + b_1(x - t_1)_+^p + \cdots + b_K(x - t_K)_+^p, \quad (21.12)$$

where, as we have seen for the specific case of $p = 2$, $(x-t)_+^p$ equals $\{(x-t)_+\}^p$. The first $p-1$ derivatives of s are continuous while the pth derivative takes a jump equal to $p!\, b_k$ at the kth knot.

21.4.5 Other Spline Bases

Given a degree p and knots $\kappa_1, \ldots, \kappa_K$, the polynomials $1, x, \ldots, x^p$ and plus functions $(x - \kappa_1)_+^p, \ldots, (x - \kappa_K)_+^p$ form a spline basis. What this means is that any pth degree spline with knots $\kappa_1, \ldots, \kappa_K$ is a linear combination of these basis functions. The basis of polynomials and plus functions is simple to understand, but is known to be numerically unstable if the number of knots is large. For this reason, other bases are often used for numerical computations. The B-spline basis is particular popular. It is assumed here that the reader will not be programming spline estimators from scratch but rather will be using spline software. Therefore, B-splines and other bases will not be covered here, but see Sect. 21.6 for further reading.

21.5 Penalized Splines

Because a pth degree spline with K knots has $1 + p + K$ parameters, an ordinary least-squares fit will usually overfit the data unless both p and K are kept small, for instance, $1 + p + K \leq 6$. (There is nothing especial about the number 6 and it is just being used as a rule of thumb. Any number between 5 and 10 would be equally good.) An example is the quadratic spline with one knot (so $1 + p + K = 4$) used as a forward-rate curve in Example 11.3. However, a spline with p and K both small is essentially a parametric model. To have the flexibility of a nonparametric model, that is, a wide range of potential values of p_{eff}, we need to have K large and find another way to avoid overfitting. Penalized least-squares estimation does this.

Let $\mu(x; \boldsymbol{\beta}) = \boldsymbol{B}(x)^\mathsf{T}\boldsymbol{\beta}$ be a spline, where $\boldsymbol{\beta}$ is a vector of coefficients and $\boldsymbol{B}(x) = (B_1(x), \ldots, B_{1+p+K}(x))^\mathsf{T}$ is a spline basis. For example, $\boldsymbol{B}(x) = (1, x, \ldots, x_p, (x - \kappa_1)_+^p, \ldots, (x - \kappa_K)_+^p)$ if we use model (21.12). A penalized least-squares estimator minimizes over $\boldsymbol{\beta}$ the penalized sum of squares

$$\sum_{i=1}^{n} \{Y_i - \mu(X_i; \boldsymbol{\beta})\}^2 + \lambda \boldsymbol{\beta}^\mathsf{T} \boldsymbol{D} \boldsymbol{\beta}, \quad (21.13)$$

where $\boldsymbol{D}$ is a positive semidefinite matrix and $\lambda > 0$ is a penalty parameter.

A common choice of $\boldsymbol{D}$ has the i, jth element equal to

$$\int_a^b B_i^{(2)}(x) B_j^{(2)}(x) dx \quad (21.14)$$

for some $a < b$, such as, $a = \min(X_i)$ and $b = \max(X_i)$. Here $B_i^{(2)}(x)$ is the second derivative of $B_i(x)$. With this $\boldsymbol{D}$,

$$\lambda \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{D} \boldsymbol{\beta} = \lambda \int_a^b \left\{ \mu^{(2)}(x; \boldsymbol{\beta}) \right\}^2 dx, \tag{21.15}$$

Since $\mu^{(2)}(x)$ is the amount of curvature of μ at x, this choice of $\boldsymbol{D}$ penalizes wiggly functions and, if λ is chosen appropriately, prevents overfitting. If $\lambda = 0$, then there is no penalization and the effective number of parameters is $1 + p + K$. With this $\boldsymbol{D}$, in the limit as $\lambda \to \infty$, any curvature at all receives an infinite penalty, so the estimator converges to a linear polynomial fit and the effective number of parameters converges to 2. Any value of p_{eff} between 2 and $1 + p + K$ is achievable by the some value of λ between the extremes of 0 and ∞.

Let $\boldsymbol{X}$ be the $n \times (1 + p + K)$ matrix with i, jth element $B_j(X_i)$ and let $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^{\mathsf{T}}$. The penalized least-squares estimate is

$$\widehat{\boldsymbol{\beta}}(\lambda) = \left(\boldsymbol{X}^{\mathsf{T}} \boldsymbol{X} + \lambda \boldsymbol{D} \right)^{-1} \boldsymbol{X}^{\mathsf{T}} \boldsymbol{Y}, \tag{21.16}$$

which is obtained by setting the gradient of (21.13) equal to zero and solving. The fitted values are

$$\widehat{\boldsymbol{Y}}(\lambda) = \boldsymbol{X} \widehat{\boldsymbol{\beta}}(\lambda) = \left\{ \boldsymbol{X} (\boldsymbol{X}^{\mathsf{T}} \boldsymbol{X} + \lambda \boldsymbol{D})^{-1} \boldsymbol{X}^{\mathsf{T}} \right\} \boldsymbol{Y} = \boldsymbol{H}(\lambda) \boldsymbol{Y}, \tag{21.17}$$

where $\boldsymbol{H}(\lambda) = \left\{ \boldsymbol{X} (\boldsymbol{X}^{\mathsf{T}} \boldsymbol{X} + \lambda \boldsymbol{D})^{-1} \boldsymbol{X}^{\mathsf{T}} \right\}$ is the smoother matrix.

21.5.1 Cubic Smoothing Splines

A very widely used nonparametric regression estimator is the cubic smoothing spline. This estimator uses a knot at each unique value of $\{X_1, \ldots, X_n\}$ and the second-derivative penalty in (21.15). Using this many knots is not really necessary, and a variation on the cubic smoothing spline also uses penalty (21.15) but fewer knots. The knots could be equally-spaced or at selected quantiles of $\{X_1, \ldots, X_n\}$.

The function `smooth.spline()` in `R` fits a cubic smoothing spline if the argument `all.knots` is `TRUE` or if $n < 50$. If `all.knots = FALSE` and $n > 49$, then it uses less than the full set of knots.

21.5.2 Selecting the Amount of Penalization

The penalty parameter λ determines the amount of smoothing and can be chosen by AIC or GCV. Another popular method for choosing λ is REML (restricted maximum likelihood). REML is based on a so-called mixed model, where some of the spline coefficients are random variables. A description of mixed models and REML is beyond the scope of this book, but the interested reader may consult the references in Sect. 21.6.

Example 21.2. Estimating the drift and volatility for the evolution of the risk-free returns

In this example, we return to estimating the drift and squared volatility functions for the evolution of the risk-free returns. Three estimators will be used: local linear, local quadratic, and a penalized spline. The R code is:

```
library(Ecdat)
library(KernSmooth)
library(locfit)
library(mgcv)
data(Capm)
attach(Capm)
n = length(rf)
year = seq(1960.125,2003,length=n)
diffrf=diff(Capm$rf)
rf_lag = rf[1:(n-1)]
log_rf_lag = log(rf_lag)
ll_mu <- locpoly(rf_lag, diffrf, bandwidth = dpill(rf_lag,diffrf))
muhat = spline(ll_mu$x, ll_mu$y, xout = rf_lag)$y
epsilon_sqr = (diffrf - muhat)^2
ll_sig <- locpoly(rf_lag, epsilon_sqr,
   bandwidth = dpill(rf_lag, epsilon_sqr) )
gam_mu = gam(diffrf ~ s(rf_lag, bs = "cr"), method = "REML")
epsilon_sqr = (diffrf-gam_mu$fit)^2
gam_sig = gam(epsilon_sqr ~ s(rf_lag, bs = "cr"), method = "REML")
locfit_mu = locfit(diffrf ~ rf_lag)
epsilon_sqr = (diffrf - fitted(locfit_mu))^2
locfit_sig = locfit(epsilon_sqr ~ rf_lag)
std_res = (diffrf - fitted(locfit_mu)) / sqrt(fitted(locfit_sig))
min(rf_lag[(gam_mu$fit < 0)])
orrf = order(rf_lag)
pdf("riskfree02.pdf", width = 8, height = 4)
par(mfrow=c(1, 2))
plot(rf_lag[orrf], gam_mu$fit[orrf], type = "l", lwd = 3, lty = 1,
   xlab = "lagged rate", ylab = "change in rate", main = "(a)")
lines(ll_mu$x,ll_mu$y, lwd = 3, lty = 2, col = "red")
lines(locfit_mu, lwd = 3, lty = 3, col = "blue")
legend(0.1, -0.05, c("spline", "local linear", "local quadratic"),
   lty = c(1, 2, 3), cex = 0.85, lwd = 3,
   col = c("black", "red", "blue"))
rug(rf_lag)
abline(h = 0, lwd = 2)
plot(rf_lag[orrf], gam_sig$fit[orrf], type="l", lwd = 3, lty = 1,
   ylim = c(0, 0.03), xlab = "lagged rate",
   ylab = "squared residual", main = "(b)")
lines(ll_sig$x,ll_sig$y, lwd = 3, lty = 2, col = "red")
lines(locfit_sig, lwd = 3, lty = 3, col = "blue")
abline(h = 0, lwd = 2)
```

```
43 legend("topleft", c("spline", "local linear", "local quadratic"),
44    lty = c(1, 2, 3), cex = 0.85, lwd = 3,
45    col = c("black", "red", "blue"))
46 rug(rf_lag)
47 graphics.off()
```

The first estimator, local linear, is computed at line 12 using the function `locpoly()` in R's `KernSmooth` package. The dpi plug-in bandwidth selector is computed using the function `dpill()` in this package.[2]

In the R code, the changes in the risk-free returns (`diffrf`) are regressed on the lagged returns (`rf_lag`) to estimate the drift. The local linear estimator is computed on an equally-spaced grid, and to compute residuals the function `spline()` is used at line 13 to interpolate the fit to the observed values of `rf_lag`. Finally, the squared residuals (`epsilon_sqr`) computed at line 14 are regressed at lines 15–16 on the lagged returns to estimate the squared volatility function. The estimated drift function is in the object `ll_mu` and the estimated squared volatility function is in `ll_sig`.

The penalized spline estimator is computed at line 17 by the `gam()` function in the `mgcv` package. The specification `bs = "cr"` requests a cubic spline fit with penalty (21.15). The REML method is used to select the amount of smoothing.

The local quadratic estimator is computed at line 20 with the function `locfit()` in R's `locfit` package. Spline interpolation is not necessary here, since with `locfit()` the fitted values can be computed with the `fitted` function.

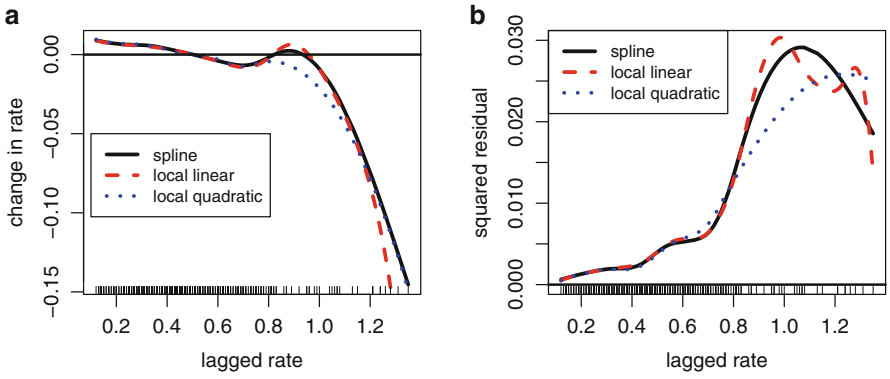

Fig. 21.7. *Risk-free monthly returns. (a) Estimates of the drift function. (b) Estimates of the squared volatility function.*

[2] "dpill" means "direct plug-in, local linear."

All three estimated drift functions are shown in Fig. 21.7a and the squared volatility function estimates are in Fig. 21.7b.

The drift functions have a general decreasing trend and are negative to the right of 0.51 (approximately), except that the estimates have humps around 0.9–1.0 and the spline and local linear estimates are slightly positive at this hump. It is likely that the hump is due to random variation, which increases as one moves from left to right (see Fig. 21.1). If we use the local quadratic fit, then the estimated drift is positive to the left of 0.51 and negative to the right of 0.51. The drift will cause reversion to a mean of 0.51, which is an annual rate of $6.12\,\% = (12)(0.51)\,\%$. The Chan et al. (1992) drift function, $\mu(r) = \beta(r - \alpha)$, is also mean-reverting, but linear. In contrast, the local quadratic estimated drift function in Fig. 21.7 is nonlinear and shows much faster reversion to the mean when the rate is high.

The squared volatility estimates show that volatility increases with the rate, at least to a point. For very high rates, the estimated volatility function becomes decreasing. There is not enough data with extremely high rates to tell if this phenomenon is "real" or due to random estimation error. The extremely high rates occurred only for the brief period in the early 1980s; see Fig. 21.1a.

The standardized residuals $\{\Delta r_t - \widehat{\mu}(r_{t-1})\}/\widehat{\sigma}(r_{t-1})$ show negative serial correlation and GARCH-type volatility clustering; see Fig. 21.8. Neither of these is surprising. Negative lag-1 autocorrelation is common in a differenced series and volatility clustering is certainly to be expected in any financial time series. This case study could be continued by fitting an ARMA/GARCH model to the standardized residuals. □

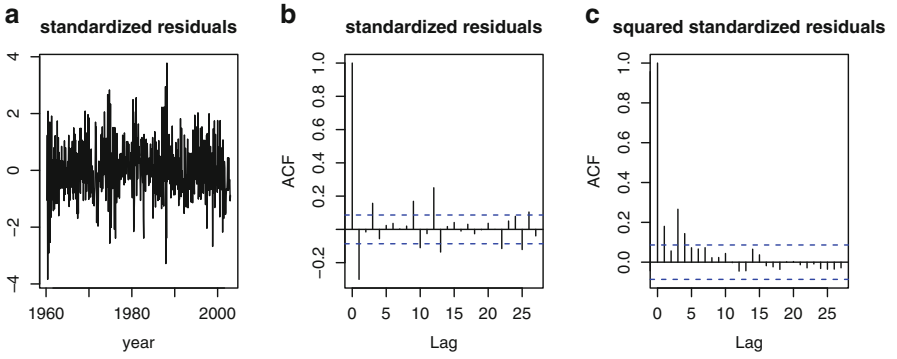

Fig. 21.8. *Risk-free monthly returns. Residual analysis. (a) Time series plot of standardized residuals. (b) ACF of standardized residuals. (c) ACF of squared standardized residuals.*

Example 21.3. Spline estimation of a forward rate

This example used the STRIPS data that were already analyzed in Example 11.3. In that example, an unpenalized spline was fit to the bond prices by nonlinear regression, and smoothness was controlled by using only knot.

The function `gam()` in the `mgcv` is a powerful tool that can fit a wide variety of spline models with penalties. In this example, `gam()` is used to fit a cubic spline to the empirical forward rates that are defined at the beginning of Sect. 11.3. The code is below.

```
dat = read.table("strips_dec95.txt", header = T)
T = dat$T
n = length(T)
ord = order(T)
T = T[ord]
price = dat$price[ord]
Int_F = - log(price) + log(100)
emp_forward = diff(Int_F)/diff(T)
library(mgcv)
X = T[-1]
fit_gam = gam(emp_forward ~ s(X, bs = "cr"))
pred_gam = predict(fit_gam, as.data.frame(X)  )
pdf("forward_spline.pdf", width = 6, height = 5)
plot(X, emp_forward, xlab = "maturity", ylab = "forward rate")
lines(X, pred_gam, col = "red", lwd = 2)
graphics.off()
```

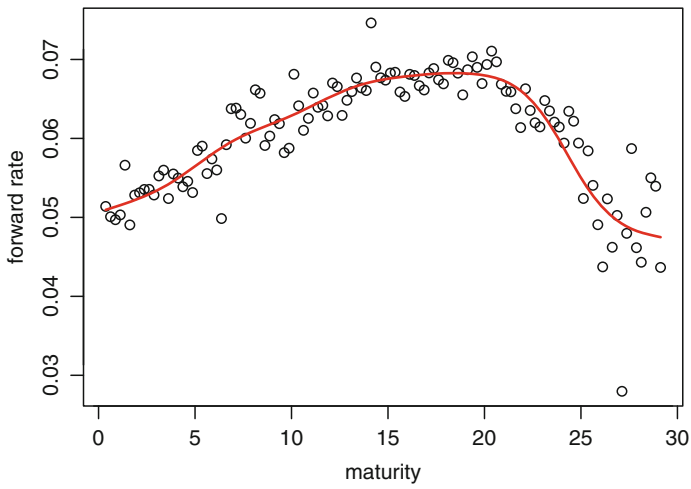

Fig. 21.9. *Empirical forward rates (circle) with a spline fit.*

Figure 21.9 shows the empirical forward rates and the cubic spline fit to them. Notice that the spline goes through the empirical forward rates with little sign of bias and yet is smooth. □

21.6 Bibliographic Notes

Ruppert, Wand, and Carroll (2003) and Wood (2006) offer comprehensive introductions to nonparametric and semiparametric modeling and their applications. Wand and Jones (1995) and Fan and Gijbels (1996) are good sources of information about local polynomial regression. REML is discussed in detail by Ruppert, Wand, and Carroll (2003) and Wood (2006) . Wasserman (2006) is an interesting modern synthesis of nonparametric estimation. Wood (2006) is a good introduction to his `mgcv` package.

21.7 R Lab

21.7.1 Additive Model for Wages, Education, and Experience

This section uses the Current Population Survey data in the `CPS1988` data set introduced in Sect. 10.4.1. We will fit spline effects for both predictors, `education` and `experience`. This is easily done with the `gam()` function in the `mgcv` package. The model being fit is

$$\log(\text{wage}) = \beta_0 + s_1(\texttt{education}) + s_2(\texttt{experience}) + \beta_1 \texttt{ethnicity} + \epsilon_i,$$

where β_0 is the intercept, s_1 and s_2 are splines, `ethnicity` is 0 for Caucasians and 1 for African Americans, and ϵ_i is white noise. To fit this model, print its summary, and plot the estimates of s_1 and s_2, run:

```
library(AER)
library(mgcv)
data(CPS1988)
attach(CPS1988)
fitGam = gam(log(wage)~s(education)+s(experience)+ethnicity)
summary(fitGam)
par(mfrow=c(1,2))
plot(fitGam)
```

Problem 1 *What are the estimates of β_0 and β_1?*

Problem 2 *Describe the shapes of s_1 and s_2.*

21.7.2 An Extended CKLS Model for the Short Rate

In this section, we use splines to extend the CKLS model in Sect. 11.12 by letting the drift parameters a and θ vary with time so that

$$\mu(t, r) = a(t)\{\theta(t) - r\}. \tag{21.18}$$

One could also let the volatility parameters σ and γ vary as well with t, but, for simplicity, we will not do that here. We will fit this model with $a(t)$ being linear in time and $\theta(t)$ being a piecewise linear spline. [Letting both $a(t)$ and $\theta(t)$ be splines can lead to unstable estimates, so we will restrict $a(t)$ to be linear.] First, read in the data, and then create the knots and the truncated line basis functions.

```
#  CKLS, extended
library(Ecdat)
data(Irates)
r1 = Irates[,1]
n = length(r1)
lag_r1 = lag(r1)[-n]
delta_r1 = diff(r1)
n = length(lag_r1)
knots = seq(from=1950,to=1985,length=10)
t = seq(from=1946,to =1991+2/12,length=n)
X1 = outer(t,knots,FUN="-")
X2 = X1 * (X1>0)
X3 = cbind(rep(1,n), (t - 1946),X2)
m2 = dim(X3)[2]
m = m2 - 1
```

Problem 3 *How many knots are being used here? What does the* `outer()` *function do here? What is done by the statement* `X2 = X1 * (X1>0)`*? Describe what is in the variable* `X3`*.*

Now fit the CKLS model with time-varying drift.

```
nlmod_CKLS_ext = nls(delta_r1 ~ X3[,1:2]%*%a *
   (X3%*%theta-lag_r1),
   start=list(theta = c(10,rep(0,m)),
   a=c(.01,0)),control=list(maxiter=200))
AIC(nlmod_CKLS_ext)
param4 = summary(nlmod_CKLS_ext)$parameters[,1]
par(mfrow=c(1,3))
plot(t,X3%*%param4[1:m2],ylim=c(0,16),ylab="rate",
   main="(a)",col="red",type="l",lwd=2)
lines(t,lag_r1)
legend("topleft",c("theta(t)","lagged rate"),lwd=c(2,1),
   col=c("red","black"))
```

```
plot(t,X3[,1:2]%*%param4[(m2+1):(m2+2)],ylab="a(t)",
   col="red",type="l",lwd=2,main="(b)")

res_sq = residuals(nlmod_CKLS_ext)^2
nlmod_CKLS_ext_res <- nls(res_sq ~  A*lag_r1^B,
   start=list(A=.2,B=1/2) )

plot(lag_r1,sqrt(res_sq),pch=5,ylim=c(0,6),ylab="",main="(c)")
lines(lag_r1,sqrt(fitted(nlmod_CKLS_ext_res)),
   lw=3,col="red",type="l")
legend("topleft",c("abs res","volatility fn"),lty=c(NA,1),
   pch=c(5,NA),col=c("black","red"),lwd=1:2)
```

Problem 4 *Explain why* `X3[,1:2]%*%a` *is a linear function but* `X3%*%theta` *is a spline.*

Problem 5 *What is the interpretation of a time-varying θ? Note that in panel (a), θ seems to track the interest rate. Does this make sense? Why or why not?*

Problem 6 *Would you accept or reject the null hypothesis that $a(t)$ is constant, that is, that the slope of the linear function $a(t)$ is zero? Justify your answer.*

21.8 Exercises

1. A linear spline $s(t)$ has knots at 1, 2, and 3. Also, $s(0) = 1$, $s(1) = 1.3$, $s(2) = 5.5$, $s(4) = 6$, and $s(5) = 6$.
   (a) What is $s(0.5)$?
   (b) What is $s(3)$?
   (c) What is $\int_2^4 s(t)\,dt$?
2. Suppose that (21.1) holds with $\mu(r) = 0.1(0.035 - r)$ and $\sigma(r) = 2.3r$.
   (a) What is the expected value of r_t given that $r_{t-1} = 0.04$?
   (b) What is the variance of r_t given that $r_{t-1} = 0.02$?
3. Let the spline $s(x)$ be defined as

$$s(x) = (x)_+ - 3(x-1)_+ + (x-2)_+.$$

   (a) Is $s(x)$ either a probability density function (pdf) or a cumulative distribution function (cdf)? Explain your answer.
   (b) If X is a random variable and s is its pdf or cdf [whichever is the correct answer in (a)], then what is the 90th percentile of X?
4. Let s be the spline

$$s(x) = 1 + 0.65x + x^2 + (x-1)_+^2 + 0.6(x-2)_+^2.$$

   (a) What are $s(1.5)$ and $s'(1.5)$?
   (b) What is $s''(2.2)$?

References

Chan, K. C., Karolyi, G. A., Longstaff, F. A., and Sanders, A. B. (1992) An empirical comparison of alternative models of the short-term interest rate. *Journal of Finance*, **47**, 1209–1227.

Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985) A theory of the term structure of interest rates. *Econometrica*, **53**, 385–407.

Fan, J., and Gijbels, I. (1996) *Local Polynomial Modelling and Its Applications*, Chapman & Hall, London.

Merton, R. C. (1973) Theory of rational option pricing. *Bell Journal of Economics and Management Science*, **4**, 141–183.

Ruppert, D., Sheather, S., and Wand, M. P. (1995) An effective bandwidth selector for local least squares kernel regression, *Journal of the American Statistical Association*, **90**, 1257–1270.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003) *Semiparametric Regression*, Cambridge University Press, Cambridge.

Vasicek, O. A. (1977) An equilibrium characterization of the term structure. *Journal of Financial Economics*, **5**, 177–188.

Wand, M. P., and Jones, M. C. (1995) *Kernel Smoothing*, Chapman & Hall, London.

Wasserman, L. (2006) *All of Nonparametric Statistics*, Springer, New York.

Wood, S. (2006) *Generalized Additive Models: An Introduction with R*, Chapman & Hall, Boca Raton, FL.

Yau, P., and Kohn, R. (2003) Estimation and variable selection in nonparametric heteroskedastic regression. *Statistics and Computing*, **13**, 191–208.