Chapter 3

Nonparametric Regression

3.1 Introduction

We consider here nonparametric regression with one predictor variable. Practically relevant generalizations to more than one or two predictor variables are not so easy due to the curse of dimensionality mentioned in section 2.4.1 and often require different approaches, as will be discussed later in Chapter 7.

Figure 3.1 shows (several identical) scatter plots of (x_i, Y_i) $(i = 1, \ldots, n)$. We can model such data as

$$Y_i = m(x_i) + \varepsilon_i, \tag{3.1}$$

where $\varepsilon_1, \ldots, \varepsilon_n$ i.i.d. with $\mathbb{E}[\varepsilon_i] = 0$ and $m : \mathbb{R} \to \mathbb{R}$ is an "arbitrary" function. The function $m(\cdot)$ is called the nonparametric regression function and it satisfies $m(x) = \mathbb{E}[Y|x]$. The restriction we make for $m(\cdot)$ is that it fulfills some kind of smoothness conditions. The regression function in Figure 3.1 does not appear to be linear in x and linear regression is not a good model. The flexibility to allow for an "arbitrary" regression function is very desirable; but of course, such flexibility has its price, namely an inferior estimation accuracy than for linear regression.

3.2 The kernel regression estimator

We can view the regression function in (3.1) as

$$m(x) = \mathbb{E}[Y|X = x],$$

(assuming that X is random and $X_i = x_i$ are realized values of the random variables). We can express this conditional expectation as

$$\int_{\mathbb{R}} y f_{Y|X}(y|x)dy = \frac{\int_{\mathbb{R}} y f_{X,Y}(x,y)dy}{f_X(x)},$$

where $f_{Y|X},\ f_{X,Y},\ f_X$ denote the conditional, joint and marginal densities. We can now plug in the univariate and bivariate kernel density (all with the same univariate kernel K) estimates

$$\hat{f}_X(x) = \frac{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)}{nh},\ \hat{f}_{X,Y}(x,y) = \frac{\sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) K\left(\frac{y-Y_i}{h}\right)}{nh^2}$$

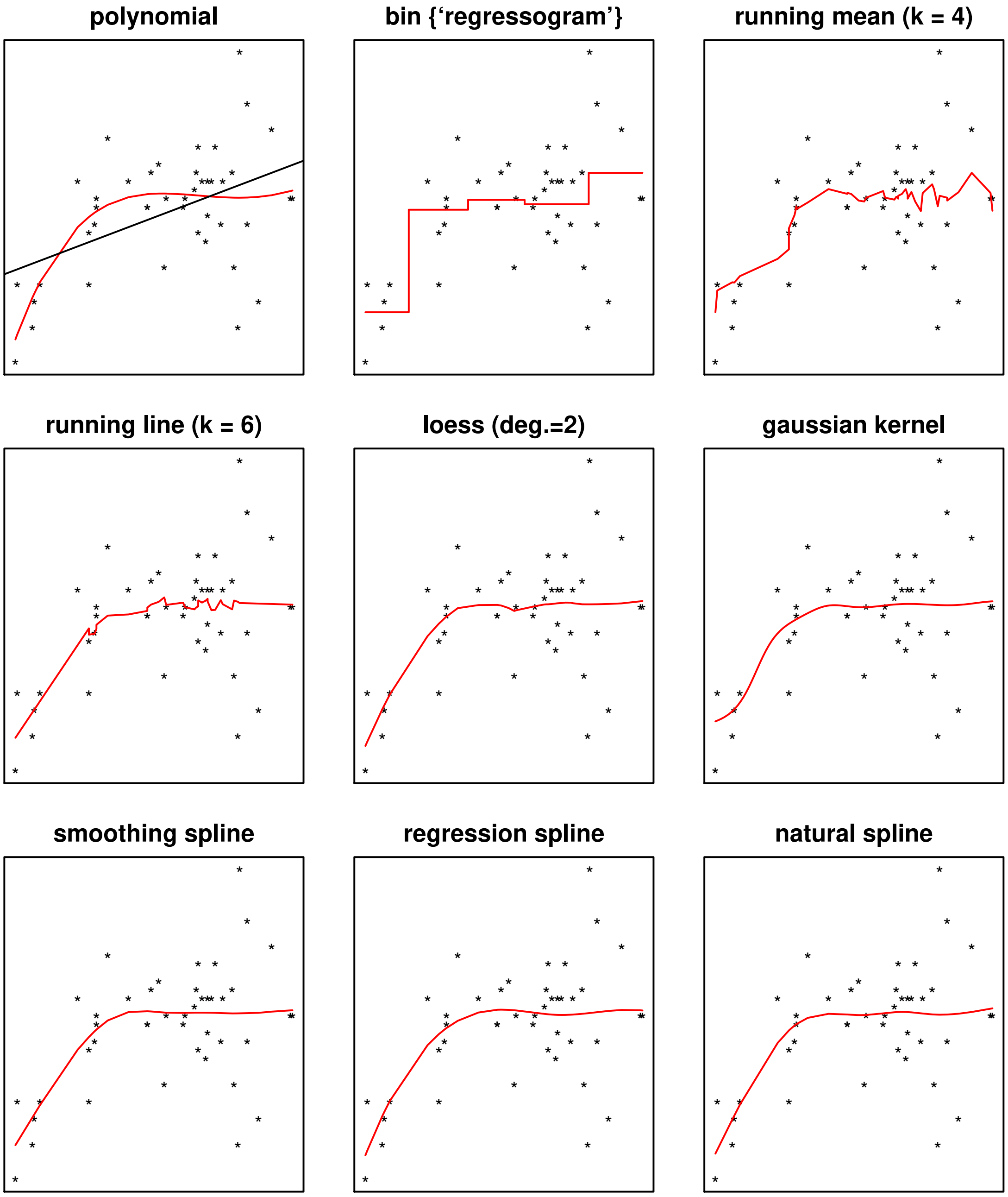

Figure 3.1: Various regression estimators in model $Y_i = m(x_i) + \varepsilon_i$ $(i = 1, \dots, 43)$ with response Y a log-concentration of a serum (in connection of Diabetes) and predictor variable x the age in months of children. See Hastie and Tibshirani (1990, p.10). Except for the linear regression fit (top left panel), all other estimators have about 5 degrees of freedom.

into the formula above which yields the so-called Nadaraya-Watson kernel estimator

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K((x - x_i)/h) Y_i}{\sum_{i=1}^{n} K((x - x_i)/h)}. \tag{3.2}$$

An interesting interpretation of the kernel regression estimator in (3.2) is

$$\hat{m}(x) = \arg\min_{m_x \in \mathbb{R}} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)(Y_i - m_x)^2. \tag{3.3}$$

This can be easily verified by solving $\frac{d}{dm_x}\sum_{i=1}^{n} K((x-x_i)/h)(Y_i - m_x)^2 = 0$. Thus, for every fixed x, we are searching for the best local constant m_x such that the localized sum of squares is minimized; localization is here described by the kernel and gives a large weight to those observations (x_i, Y_i) where x_i is close to the point x of interest.

3.2.1 The role of the bandwidth

Analogously as in section 2.3, the bandwidth h controls the bias-variance trade-off: a large bandwidth h implies high bias but small variance, resulting in a slowly varying curve, and vice-versa.

Local bandwidth selection

Similarly as in (2.7), there is a formula of the asymptotically best local bandwidth $h_{\text{opt}}(x)$ which depends on $m''(\cdot)$ and the error variance σ_ε^2:

$$h_{\text{opt}}(x) = n^{-1/5}\left(\frac{\sigma_\varepsilon^2 \int K^2(z)dz}{\{m''(x)\int z^2 K(z)dz\}^2}\right)^{1/5}. \tag{3.4}$$

The locally optimal bandwidth $h_{\text{opt}}(x)$ can then be estimated in an iterative way using

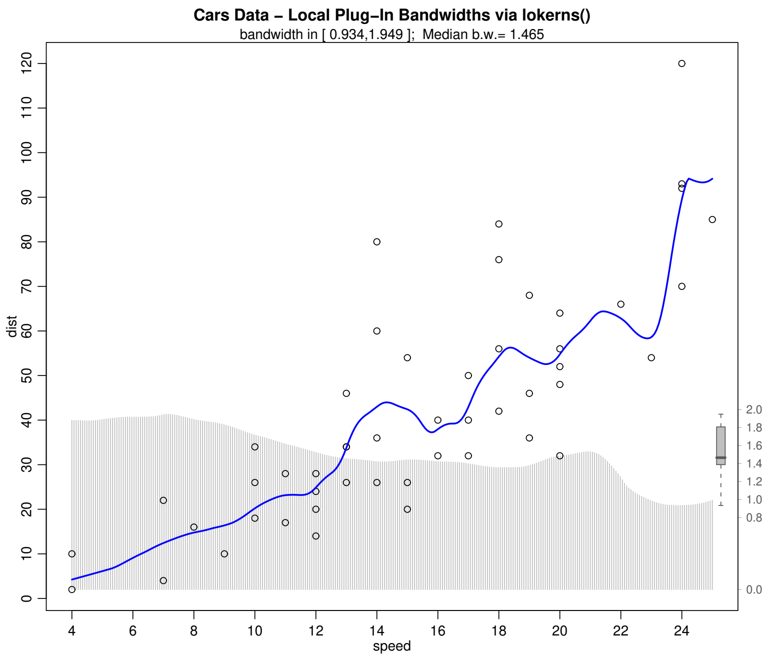

Figure 3.2: Nonparametric function estimate and locally varying bandwidths for distance of stopping as a function of speed of cars.

the plug-in principle. Roughly speaking, start with an initial bandwidth h_0 to estimate $m''(\cdot)$ (by using an inflated version $n^{1/10}h_0$) and σ_ε^2; these estimates can then be used to get a first estimate of $h_{\text{opt}}(x)$. Now use this first bandwidth estimate as the current bandwidth h_1 to estimate again $m''(\cdot)$ (by using the inflated version $n^{1/10}h_1$) and σ_ε^2, and then obtain new estimates for $h_{\text{opt}}(x)$; and so on, see Brockmann et al. (1993).

Such a procedure has been implemented in R with the function `lokerns` in the package `lokern`. The dataset `cars` contains the distance for stopping as a function of speed of a car. A nonparametric function estimate with locally varying bandwidth can then be obtained as follows:

```
library(lokern); lofit <- lokerns(cars$ speed, cars$ dist)
```

3.2.2 Inference for the underlying regression curve

We consider here the properties, in particular the variability, of the kernel estimator $\hat{m}(x_i)$ at an observed design point x_i.

The hat matrix

It is useful to represent the kernel estimator evaluated at the design points $\hat{m}(x_1), \ldots, \hat{m}(x_n)$ as a *linear* operator (a matrix):

$$\begin{aligned} \mathcal{S}: \quad & \mathbb{R}^n \to \mathbb{R}^n, \\ & (Y_1, \ldots, Y_n)^\top \mapsto (\hat{m}(x_1), \ldots, \hat{m}(x_n))^\top =: \hat{\mathbf{m}}(x.) = \hat{\mathbf{Y}}, \end{aligned}$$

i.e., $\hat{\mathbf{Y}} = \mathcal{S}\mathbf{Y}$ where $\mathcal{S}$ is the matrix representing the linear operator above. The kernel estimator in (3.2) is of the form

$$\hat{m}(x) = \sum_{i=1}^{n} w_i(x) Y_i, \;\; w_i(x) = \frac{K((x - x_i)/h)}{\sum_{j=1}^{n} K((x - x_j)/h)}.$$

Therefore, the matrix $\mathcal{S}$ which represents the operator above is

$$[\mathcal{S}]_{r,s} = w_s(x_r), \;\; r, s \in \{1, \ldots, n\},$$

since $\mathcal{S}\{(Y_1, \ldots, Y_n)^\top\} = (\hat{m}(x_1), \ldots, \hat{m}(x_n))^\top$. The "smoother" matrix $\mathcal{S}$ is also called the "*hat matrix*", since it yields the vector of fitted values (at the observed design points x_i). Note many other nonparametric regression methods (including those in the next two sections) can be seen to be linear in $\mathbf{Y}$ and hence be written as $\hat{\mathbf{Y}} = \mathcal{S}\mathbf{Y}$ where $\mathcal{S}$ depends on the x-design $(x_1, x_2, \ldots, x_n)$ and typically a smoothing parameter, say h. Algorithmically, for $\hat{\mathbf{Y}} = s(\mathbf{x}, \mathbf{Y}, h)$, the hat matrix can easily be computed columnwise as $\mathcal{S}_{.,j} = s(\mathbf{x}, \mathbf{e}_j, h)$ where $\mathbf{e}_j$ is the unit vector with $(\mathbf{e}_j)_i = \delta_{i,j} := \mathbf{1}(i = j)$.

Because of the elementary formula $\mathrm{Cov}(A\mathbf{X}) = A\,\mathrm{Cov}(\mathbf{X})A^\top$ (for a non-random matrix A and random vector $\mathbf{X}$), we get the covariance matrix

$$\mathrm{Cov}(\hat{\mathbf{m}}(x.)) = \sigma_\varepsilon^2 \mathcal{S}\mathcal{S}^\top, \tag{3.5}$$

i.e., $\mathrm{Cov}(\hat{m}(x_i), \hat{m}(x_j)) = \sigma_\varepsilon^2 (\mathcal{S}\mathcal{S}^\top)_{ij}$, and $\mathrm{Var}(\hat{m}(x_i)) = \sigma_\varepsilon^2 (\mathcal{S}\mathcal{S}^\top)_{ii}$.

Degrees of freedom

One way to assign degrees of freedom for regression estimators with a linear hat-operator $\mathcal{S}$ is given by

$$df = \mathrm{trace}(\mathcal{S}). \tag{3.6}$$

This definition coincides with the notion we have seen in the linear model: there, the fitted values $\hat{Y}_1, \ldots, \hat{Y}_n$ can be represented by the projection $P = X(X^\top X)^{-1}X^\top$, which is the hat matrix, and $\text{trace}(P) = \text{trace}((X^\top X)^{-1}X^\top X) = \text{trace}(I_p) = p$ equals the number of parameters in the model. Thus, the definition of degrees of freedom above can be viewed as a general concept for the number of parameters in a model fit with linear hat matrix.

Estimation of the error variance

Formula (3.5) requires knowledge of σ_ε^2. A plausible estimate is via the residual sum of squares,

$$\hat{\sigma}_\varepsilon^2 = \frac{\sum_{i=1}^n (Y_i - \hat{m}(x_i))^2}{n - df}.$$

We then get an estimate for the standard error of the kernel regression estimator at the design points via (3.5):

$$\widehat{s.e}(\hat{m}(x_i)) = \sqrt{\widehat{\text{Var}}(\hat{m}(x_i))} = \sqrt{\frac{\sum_{j=1}^n (Y_j - \hat{m}(x_j))^2}{n - df} (\mathcal{S}\mathcal{S}^\top)_{ii}}.$$

The estimated standard errors above are useful since under regularity conditions, $\hat{m}(x_i)$ is asymptotically normal distributed:

$$\hat{m}(x_i) \approx \mathcal{N}(\mathbb{E}[\hat{m}(x_i)], \text{Var}(\hat{m}(x_i))),$$

so that

$$I = \hat{m}(x_i) \pm 1.96 \cdot \widehat{s.e.}(\hat{m}(x_i))$$

yields approximate pointwise confidence intervals for $\mathbb{E}[\hat{m}(x_i)]$. Some functions in R (e.g. the function `gam` from package `mgcv`, see Chapter 7) supply such pointwise confidence intervals. Unfortunately, it is only a confidence interval for the expected value $\mathbb{E}[\hat{m}(x_i)]$ and not for the true underlying function $m(x_i)$. Correction of this interval is possible by subtracting a bias estimate: i.e., instead of the interval I above, we can use

$$I - \widehat{\text{bias}},$$

where $\widehat{\text{bias}}$ is an estimate of the bias (which is not so easy to construct; see also section 2.3).

3.3 Local polynomial nonparametric regression estimator

As a starting point, consider the kernel estimator which can be represented as a locally constant function as in (3.3). This can now be extended to functions which are locally polynomial. We aim to find local regression parameters $\boldsymbol{\beta}(x)$, defined as

$$\widehat{\boldsymbol{\beta}}(x) = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \left(Y_i - \beta_1 - \beta_2(x_i - x) - \ldots - \beta_p(x_i - x)^{p-1}\right)^2.$$

An even number p turns out to be better: in practice, we often choose $p = 2$ or $p = 4$. The estimated local regression parameter $\widehat{\boldsymbol{\beta}}(x)$ describes a local polynomial regression fit, localized and centered at x. The function estimator is then given by evaluating this local regression fit $\sum_{j=1}^{p} \hat{\beta}_j(x)(u-x)^{j-1}$ at $u = x$: due to the centering, only the local intercept remains and the local polynomial function estimator becomes

$$\hat{m}(x) = \hat{\beta}_1(x).$$

Note that due to (local) correlation among the $(x_i - x)^j$'s, $\hat{\beta}_1(x)$ is not the same as a local constant fit from (3.3).

The local polynomial estimator is often better at the edges than the locally constant Nadaraya-Watson kernel estimator. Another interesting property is that the method also immediately yields estimates for the derivatives of the function: when differentiating the local regression fit $\sum_{j=1}^{p} \hat{\beta}_j(x)(u-x)^{j-1}$ with respect to u and evaluating it at x, we obtain

$$\hat{m}^{(r)}(x) = r!\hat{\beta}_{r+1}(x) \ (r = 0, 1, \ldots p-1).$$

3.4 Smoothing splines and penalized regression

Function estimation could also be done by using higher order global polynomials, which is often not advisable, or by using splines which can be specified by choosing a set of knots. The latter is a more locally oriented approach and is called "regression splines". Here, we discuss a method based on splines *without* having to specify where to select the knots of the spline.

3.4.1 Penalized sum of squares

Consider the following problem: among all functions m with continuous second derivatives, find the one which minimizes the penalized residual sum of squares

$$\sum_{i=1}^{n}(Y_i - m(x_i))^2 + \lambda \int m''(z)^2 dz, \tag{3.7}$$

where $\lambda \geq 0$ is a smoothing parameter. The first term measures closeness to the data and the second term penalizes curvature ("roughness") of the function. The two extreme cases are:

- $\lambda = 0$: m is any function interpolating the data (but for $\lambda \to 0$, in the limit, $m_\lambda \to$ the well defined interpolating natural cubic spline).
- $\lambda = \infty$: the least squares fit for linear regression which fulfills $m''(x) \equiv 0$.

Thus, a large λ corresponds to a smooth function.

3.4.2 The smoothing spline solution

Remarkably, the minimizer of (3.7) is finite-dimensional, although the criterion to be minimized is over a Sobolev space of functions (function space for which the integral $\int m''^2$ is defined), an infinite-dimensional space.

The solution $\hat{m}_\lambda(\cdot)$ (e.g. the minimizer of (3.7)) is a natural **cubic spline** with knots at the predictors x_i: that is, $\hat{m}$ is a piecewise cubic polynomial in each interval $[x_i, x_{i+1})$ such that $\hat{m}_\lambda^{(k)}$ $(k = 0, 1)$ is continuous everywhere.

Knowing that the solution is a cubic spline, the solution can be obtained by linear algebra. The trick is to represent

$$\hat{m}_\lambda(x) = \sum_{j=1}^{n} \beta_j B_j(x), \tag{3.8}$$

where the $B_j(\cdot)$'s are basis functions for natural splines. The unknown coefficients can then be estimated from least squares in linear regression under side constraints. The criterion in (3.7) for $\hat{m}_\lambda$ as in (3.8) then becomes

$$\|Y - B\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^\top \Omega \boldsymbol{\beta},$$

where the design matrix B has jth column $(B_j(x_1), \ldots, B_j(x_n))^\top$ and $\Omega_{jk} = \int B_j''(z) B_k''(z) dz$. The solution can then be derived in a straightforward way,

$$\widehat{\boldsymbol{\beta}}_{n\times 1} = (B^\top B + \lambda\Omega)^{-1} B^\top Y. \tag{3.9}$$

This can be computed efficiently using fast linear algebra.

The fitted values are then $\hat{Y}_i = \hat{m}_\lambda(x_i)$ $(i = 1, \ldots, n)$ and

$$(\hat{Y}_1, \ldots, \hat{Y}_n)^\top = \mathcal{S}_\lambda Y, \quad \mathcal{S}_\lambda = B(B^\top B + \lambda\Omega)^{-1} B^\top.$$

The hat matrix $\mathcal{S}_\lambda = \mathcal{S}_\lambda^\top$ is here symmetric which implies elegant mathematical properties (real-valued eigen-decomposition).

3.4.3 Shrinking towards zero

At first sight, the smoothing spline solution in (3.8) looks heavily over-parameterized since we have to fit n unknown coefficients $\beta_1, \ldots, \beta_n$. However, the solution in (3.9) is not the least squares estimator but rather a Ridge-type version: the matrix $\lambda\Omega$ serves as a Ridge or shrinkage matrix so that the estimates $\hat{\beta}$ are shrunken towards zero: i.e., for large λ, the expression $(B^\top B + \lambda\Omega)^{-1}$ becomes small. Thus, since all the coefficients are shrunken towards zero, we gain on the variance part of each $\hat{\beta}_j$ by the square of the shrinkage factor, and the overall smoothing spline fit will be appropriate if λ is chosen suitably.

Note that λ can be chosen on the scale of equivalent degrees of freedom (df): df $=$ $trace(\mathcal{S}_\lambda)$. This provides an intuitive way to specify a smoothing spline: e.g. a smoothing spline with df=5 is as complex as a global polynomial of degree 4 (which has 5 parameters including the intercept), see also Figure 3.1.

3.4.4 Relation to equivalent kernels

It is interesting to note that there is a relationship between the smoothing spline estimate and a particular kernel estimator. The smoothing spline estimate $\hat{m}(x)$ is approximately

$$\begin{aligned}
\hat{m}_\lambda(x) &\approx \sum_{i=1}^{n} w_i(x) Y_i, \\
w_i(x) &= \frac{1}{n h(x) f_X(x)} K\left(\frac{x - x_i}{h(x)}\right), \\
h(x) &= \lambda^{1/4} n^{-1/4} f_X(x)^{-1/4}, \\
K(u) &= \frac{1}{2} \exp\left(-\frac{|u|}{\sqrt{2}}\right) \sin\left(\frac{|u|}{\sqrt{2}} + \frac{\pi}{4}\right).
\end{aligned}$$

See for example Green and Silverman (1994, Ch. 3.7).

The important fact is here that the bandwidth of the equivalent kernel estimator has a *local bandwidth*, depending on the density of the predictor variable x. In regions where the density of the predictor is low (observations are sparse), the bandwidth automatically adapts and becomes large: intuitively, this is the right behavior because we should use strong smoothing in regions where only few observations are available.

An example of a smoothing spline fit for real data is displayed in Figure 3.1. Finally, we illustrate on an artificial dataset the advantage of smoothing splines to adapt to the density of the predictor variables. Figure 3.3 shows the performance of smoothing splines in comparison with the Nadaraya-Watson Gaussian kernel estimator. The data has the following structure:

- the density of the predictor is high for positive values and low for negative values
- the true function is strongly oscillating where the predictor density is high and slowly oscillating where the predictors are sparse

The smoothing spline fit (using the GCV criterion for selecting the degrees of freedom, see section 4.5) yields a very good fit: it captures the strong oscillations because there are many data points with positive values of the predictors. On the other hand, the kernel estimator has been tuned such that it captures the strong oscillations, using a small bandwidth h (this was done by knowing the true underlying function – which is not feasible in practice): but the small bandwidth h then causes a much too rough and poor estimate for negative predictor values, although the underlying true function is smooth.

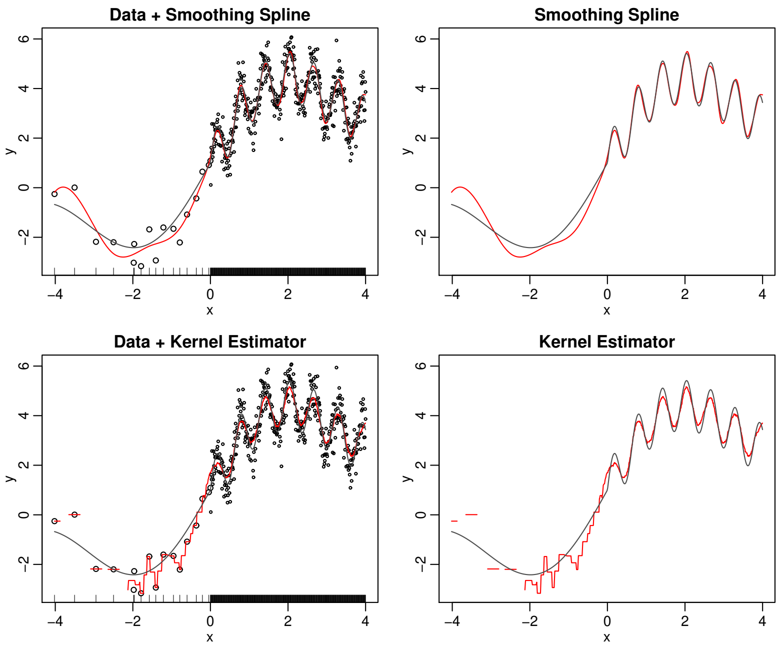

Figure 3.3: Curve estimation for synthetic dataset ($n = 417$). Left panels: scatterplot, overlaid with curve estimates (red) and true curve (gray); Right panels: curve estimates (red) and true curve (gray).
Top: Smoothing spline with GCV-selected df (= 9.23); Bottom: Nadaraya-Watson kernel estimator with bandwidth chosen for good estimation of the strong oscillations of the true function (giving $df = 25.6$).