Nonparametric Regression in R

An Appendix to *An R Companion to Applied Regression*, third edition

John Fox & Sanford Weisberg

last revision: 2018-09-26

Abstract

In traditional parametric regression models, the functional form of the model is specified before the model is fit to data, and the object is to estimate the parameters of the model. In nonparametric regression, in contrast, the object is to estimate the regression function directly without specifying its form explicitly. In this appendix to Fox and Weisberg (2019), we describe how to fit several kinds of nonparametric-regression models in R, including scatterplot smoothers, where there is a single predictor; models for multiple regression; additive regression models; and generalized nonparametric-regression models that are analogs to generalized linear models.

1 Nonparametric Regression Models

The traditional nonlinear regression model that is described in the on-line appendix to the *R Companion* on nonlinear regression fits the model

$$y = m(\mathbf{x}, \boldsymbol{\theta}) + \varepsilon$$

where $\boldsymbol{\theta}$ is a vector of parameters to be estimated, and $\mathbf{x}$ is a vector of predictors;[1] the errors ε are assumed to be normally and independently distributed with mean 0 and constant variance σ^2. The function $m(\mathbf{x}, \boldsymbol{\theta})$, relating the average value of the response y to the predictors, *is specified in advance*, as it is in a linear regression model.

The *general nonparametric regression model* is written in a similar manner, but the function m is left unspecified:

$$\begin{aligned} y &= m(\mathbf{x}) + \varepsilon \\ &= m(x_1, x_2, \ldots, x_p) + \varepsilon \end{aligned}$$

for the p predictors $\mathbf{x} = (x_1, x_2, \ldots, x_p)'$. Moreover, the object of nonparametric regression is to estimate the regression function $m(\mathbf{x})$ directly, rather than to estimate parameters. Most methods of nonparametric regression implicitly assume that m is a smooth, continuous function.[2] As in nonlinear regression, it is standard to assume that $\varepsilon \sim \text{NID}(0, \sigma^2)$.

An important special case of the general model is nonparametric simple regression, where there is only one predictor:

$$y = m(x) + \varepsilon$$

[1] If you're unfamiliar with vector notation, simply think of $\mathbf{x}$ is the collection of predictors and $\boldsymbol{\theta}$ as the collection of regression parameters.

[2] An exception to the implicit assumption of smoothness is wavelet regression, not discussed in this appendix, which is implemented in R, e.g., in the **wavethresh** package (Nason, 2016); see Nason and Silverman (1994, 2000); Nason (2008).

Nonparametric simple regression is often called "scatterplot smoothing" because an important application is to tracing a smooth curve through a scatterplot of y against x. We frequently use nonparametric regression in this manner in the body of the *R Companion*, and discuss it in Sections 3.6 and 9.2 of the text.

Because it is difficult to fit the general nonparametric regression model when there are many predictors, and because it is difficult to display the fitted model when there are more than two or three predictors, more restrictive models have been developed. One such model is the *additive regression model*,

$$y = \beta_0 + m_1(x_1) + m_2(x_2) + \cdots + m_p(x_p) + \varepsilon$$

where the partial-regression functions $m_j(x_j)$ are assumed to be smooth, and are to be estimated from the data. This model is much more restrictive than the general nonparametric regression model, but less restrictive than the linear regression model, which assumes that all of the partial-regression functions are linear.

Variations on the additive regression model include semiparametric models, in which some of the predictors enter linearly, for example,

$$y = \beta_0 + \beta_1 x_1 + m_2(x_2) + \cdots + m_p(x_p) + \varepsilon$$

This is particularly useful when some of the predictors are factors, represented in the model as dummy regressors or other contrasts. A more general version allows for selected interactions,, for example

$$y = \beta_0 + m_{12}(x_1, x_2) + m_3(x_3) + \cdots + m_p(x_p) + \varepsilon$$

All of these models extend to *generalized nonparametric regression*, much as linear models extend to generalized linear models as discussed in Chapter 6 of the *R Companion*. The random and link components are as in generalized linear models, but the linear predictor of the GLM

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

is replaced, for example, by an unspecified smooth function of the predictors

$$\eta = m(x_1, x_2, \ldots, x_p)$$

for the most general case, or by a sum of smooth partial-regression functions

$$\eta = \beta_0 + m_1(x_1) + m_2(x_2) + \cdots + m_p(x_p)$$

in the *generalized additive model*.

2 Estimation

There are several approaches to estimating nonparametric regression models, of which we will describe two: local polynomial regression and smoothing splines. With respect to implementation of these methods in R, there is an embarrassment of riches:

- Local polynomial regression is performed by the standard R functions `lowess()` (`l`ocally `we`ighted `s`catterplot `s`moother, for the simple-regression case) and `loess()` (`l`ocal `regress`ion, more generally).
- Simple-regression smoothing-spline estimation is performed by the standard R function `smooth.spline()`.

- Generalized nonparametric regression by local likelihood estimation, of which local regression is a special case for models with normal errors, is implemented in the **locfit** (**loc**al **fit**ting) package (Loader, 1999), which also performs density estimation.

- Generalized additive models may be fit with Hastie and Tibshirani's (1990) `gam()` function (in the **gam** package), which uses spline or local-regression smoothers. The `gam()` function in Wood's (2000, 2001, 2006) **mgcv** package, which is part of the standard R distribution, also fits this class of models using spline smoothers, and features automatic selection of smoothing parameters. (The name of the package comes from the method used to pick the smoothing parameters: **m**ultiple **g**eneralized **c**ross-**v**alidation.)

- There are several other R packages for nonparametric regression, including Bowman and Azzalini's (1997) **sm** (**sm**oothing) package, which performs local-regression and local-likelihood estimation, and which also includes facilities for nonparametric density estimation; and Gu's (2000) **gss** (**g**eneral **s**moothing **s**plines) package, which fits various smoothing-spline regression and generalized regression models. This is not an exhaustive list!

2.1 Local Polynomial Regression

2.1.1 Simple Regression

We are looking to fit the model

$$y = m(x) + \varepsilon$$

Focus first on evaluating the regression function at a particular x-value, x_0. Ultimately, we will fit the model at a representative range of values of x or simply at the n cases, x_i. We proceed to perform a pth-order weighted-least-squares polynomial regression of y on x,

$$y = b_0 + b_1(x - x_0) + b_2(x - x_0)^2 + \cdots + b_p(x - x_0)^p + e$$

weighting the cases in relation to their distance from the focal value x_0; a common weight function to use is the *tricube function*:

$$W(z) = \begin{cases} (1 - |z|^3)^3 & \text{for } |z| < 1 \\ 0 & \text{for } |z| \geq 1 \end{cases}$$

In the present context, $z = (x - x_0)/h$, where h determines the smoothness of the fit and so is called a smoothing parameter. The fitted value at x_0, that is, the estimated height of the regression curve, is simply $\widehat{y}_0 = b_0$, produced conveniently by having centered the predictor x at the focal value x_0.

The value of h is either fixed for all values of x_0 to provide a fixed window width, or else it is adjusted for each x_0 to include a fixed fraction s of the data. In this case s may be called the *span* of the local-regression smoother. The larger the span, the smoother the result; in contrast, the larger the order of the local regressions p, the more flexible the smooth, so the span and the order of the local regressions can be traded off against one-another.

The process of fitting a local regression is illustrated in Figure 1, using the Canadian occupational-prestige data introduced in Chapter 3 of the *Companion*, which reside in the `Prestige` data set in the **carData** package. We examine the regression of `prestige` on `income`, focusing initially on the case with the $80th$ largest `income` value, $x_{(80)}$, represented in Figure 1 by the vertical solid line.[3]

- A window including the 50 nearest x-neighbors of $x_{(80)}$ (i.e., for span $s = 50/102 \approx 1/2$) is shown in Figure 1 (a).

- The tricube weights for cases in this neighborhood appear in Figure 1 (b).

[3]We draw a very similar graph for different data in Figure 9.13 the *R Companion*.

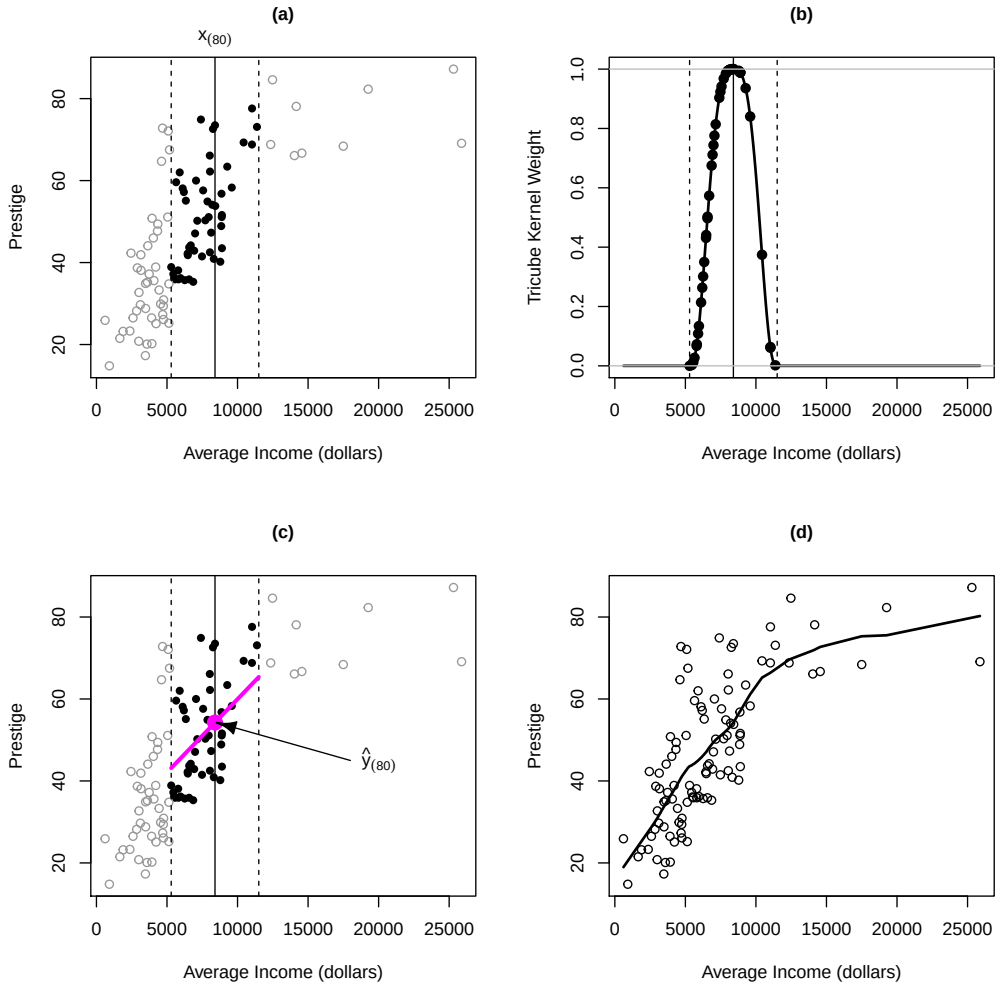

Figure 1: Local linear regression of `prestige` on `income` for the Canadian occupational-prestige data: (a) The broken lines delimit the 50 nearest neighbors of $x_{(80)}$ (at the solid vertical line). (b) Tricube weights for cases in the neighborhood of $x_{(80)}$. (c) Locally weighted linear regression in the neighborhood of $x_{(80)}$; the larger magenta solid dot is the fitted value $\widehat{y}_{(80)}$ above $x_{(80)}$. (d) The completed locally linear regression, connecting fitted values across the range of x.

- Figure 1 (c) shows the locally weighted regression line fit to the data in the neighborhood of x_0 (i.e., a local polynomial regression of order $p = 1$); the fitted value $\widehat{y}|x_{(80)}$ is represented in this graph as a larger solid dot.
- Finally, in Figure 1 (d), local regressions are estimated for a range of x-values, and the fitted values are connected in a nonparametric-regression curve.

Figure 1 (d) is produced by the following R commands, using the `lowess` function[4]:

```
library("carData") # for data sets
plot(prestige ~ income, xlab="Average Income", ylab="Prestige", data=Prestige)
with(Prestige, lines(lowess(income, prestige, f=0.5, iter=0), lwd=2))
```

The argument `f` to `lowess()` gives the span of the local-regression smoother; `iter=0` specifies that the local regressions should *not* be refit to down-weight outlying cases.[5]

2.1.2 Multiple Regression

The nonparametric multiple regression model is

$$\begin{aligned} y &= f(\mathbf{x}) + \varepsilon \\ &= f(x_1, x_2, \ldots, x_p) + \varepsilon \end{aligned}$$

Extending the local-polynomial approach to multiple regression is simple conceptually, but can run into practical difficulties.

- The first step is to define a multivariate neighborhood around a focal point $\mathbf{x}_0' = (x_{01}, x_{02}, \ldots, x_{0k})$. The default approach in the `loess()` function is to used scaled Euclidean distances:

$$D(\mathbf{x}_i, \mathbf{x}_0) = \sqrt{\sum_{j=1}^{k} (z_{ij} - z_{0j})^2}$$

  where the z_j are the standardized predictors,

$$z_{ij} = \frac{x_{ij} - \overline{x}_j}{s_j}$$

  Here, $\mathbf{x}_i$ is the predictor vector for the ith case; x_{ij} is the value of the jth predictor for the ith case; $\overline{x}_j$ is the mean of the jth predictor; and s_j is its standard deviation.

- Weights are defined using the scaled distances:

$$w_i = W\left[\frac{D(\mathbf{x}_i, \mathbf{x}_0)}{h}\right]$$

  where $W(\cdot)$ is a suitable weight function, such as the tricube, in which case h is the half-width (i.e., radius) of the neighborhood. As in local simple regression, h may be adjusted to define a neighborhood including the $[ns]$ nearest neighbors of $\mathbf{x}_0$ (where the square brackets denote rounding to the nearest integer).

[4]Most R functions used but not described in this appendix are discussed in Fox and Weisberg (2019). All the R code used in this appendix can be downloaded from `http://tinyurl.com/carbook` or via the `carWeb()` function in the **car** package.

[5]By default, `lowess()` performs `iter=3` robustness iterations, using a bisquare weight function. The idea of weighting cases to obtain robust regression estimators, and the bisquare weight function, are described in the on-line appendix on robust regression. Alternatively, one can use the `loess.smooth()` function to get the coordinates for a local-polynomial smooth, or `scatter.smooth()` to draw the graph.

- Perform a weighted polynomial regression of y on the xs; for example, a local linear fit takes the following form:

$$y = b_0 + b_1(x_1 - x_{01}) + b_2(x_2 - x_{02}) + \cdots + b_k(x_k - x_{0k}) + e$$

  The fitted value at $\mathbf{x}_0$ is then simply $\widehat{y}_0 = b_0$.

- The procedure is repeated for representative combinations of predictor values to build up a picture of the regression surface.

Extending the illustration of the previous section, and using the `loess()` function, regress `prestige` on both the `income` and `education` levels of the occupations:

```
mod.lo <- loess(prestige ~ income + education, span=.5, degree=1, data=Prestige)
summary(mod.lo)
```

```
Call:
loess(formula = prestige ~ income + education, data = Prestige,
    span = 0.5, degree = 1)

Number of Observations: 102
Equivalent Number of Parameters: 8.03
Residual Standard Error: 6.91
Trace of smoother matrix: 10.5  (exact)

Control settings:
  span     :  0.5
  degree   :  1
  family   :  gaussian
  surface  :  interpolate          cell = 0.2
  normalize:  TRUE
 parametric:  FALSE FALSE
drop.square:  FALSE FALSE
```

Specifying `degree=1` fits locally linear regressions; the default is `degree=2` (i.e., locally quadratic regressions). To see the full range of arguments for the `loess()` function, consult `?loess`. The `summary()` output includes the standard deviation of the residuals under the model and an estimate of the equivalent number of parameters (or degrees of freedom) employed by the model — in this case, about 8 parameters. In contrast, a standard linear regression model would have used 3 parameters (the constant and 2 slopes).

As in nonparametric simple regression, there are no parameters estimates. To see the result of the regression, we have to examine the fitted regression surface graphically, as in Figure 2, produced by the following R commands:[6]

```
inc <- with(Prestige, seq(min(income), max(income), len=25))
ed <- with(Prestige, seq(min(education), max(education), len=25))
newdata <- expand.grid(income=inc, education=ed)
fit.prestige <- matrix(predict(mod.lo, newdata), 25, 25)
persp(inc, ed, fit.prestige, theta=45, phi=30, ticktype="detailed",
    xlab="Income", ylab="Education", zlab="Prestige", expand=2/3,
    shade=0.5)
```

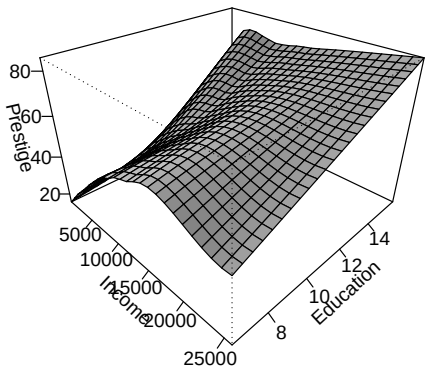

Figure 2: Fitted surface for the local linear multiple regression of `prestige` on `income` and `education`.

We use the `expand.grid()` function to create a data frame containing combinations of values of the two predictors, `income` and `education`; for each predictor, we take 25 values, evenly spaced along the range of the variable. Corresponding fitted values on the regression surface are computed by `predict()`. These predicted values are reshaped into a 25 by 25 matrix, which is passed to the `persp()` function, along with the values of the predictors (`inc` and `ed`) used to generate the regression surface. The arguments `theta` and `phi` to `persp()` control the orientation of the plot; `expand` controls the relative length of the z axis; and `shade` controls the shading of the plotted surface. See `?persp` for details.

The relationship of `prestige` to `education` and `income` appears to be nonlinear, especially in the direction of `income` (look at the grid lines on the regression surface). The partial regression in the direction of each predictor does not appear to change very much as the other predictor varies, suggesting that an additive model may be appropriate for these data. We consider such a model below.

We can also address the contribution of each predictor by dropping it from the model and performing an approximate incremental F-test for the change in the residual sum of squares. In fitting these separate models, we set the span of the local simple regressions to $0.7 \simeq \sqrt{0.5}$:

```
mod.lo.inc <- loess(prestige ~ income, span=.7, degree=1,
    data=Prestige)  # omitting education
mod.lo.ed <- loess(prestige ~ education, span=.7, degree=1,
    data=Prestige)  # omitting income
anova(mod.lo.inc, mod.lo)  # test for education

Model 1: loess(formula = prestige ~ income, data = Prestige,
    span = 0.7, degree = 1)
Model 2: loess(formula = prestige ~ income + education, data = Prestige,
    span = 0.5, degree = 1)
```

[6]There are alternative graphical representations of the regression surface, such as contour plots and coplots. The latter can be used when there are more than two predictors.

```
Analysis of Variance:   denominator df 90.66

      ENP   RSS F-value  Pr(>F)
[1,] 3.85 12006
[2,] 8.03  4246    20.8 4.8e-16

anova(mod.lo.ed, mod.lo)  # test for income

Model 1: loess(formula = prestige ~ education, data = Prestige,
    span = 0.7, degree = 1)
Model 2: loess(formula = prestige ~ income + education, data = Prestige,
    span = 0.5, degree = 1)

Analysis of Variance:   denominator df 90.66

      ENP  RSS F-value  Pr(>F)
[1,] 2.97 7640
[2,] 8.03 4246    7.79 7.1e-08
```

The contributions of both `income` and `education`, therefore, are associated with very small p-values.

2.2 Smoothing Splines

Smoothing splines arise as the solution to the following simple-regression problem: Find the function $\widehat{m}(x)$ with two continuous derivatives that minimizes the *penalized sum of squares*,[7]

$$\text{SS}^*(h) = \sum_{i=1}^{n} [y_i - m(x_i)]^2 + h \int_{x_{\min}}^{x_{\max}} [m''(x)]^2 \, dx \tag{1}$$

where h is a smoothing parameter, analogous to the neighborhood-width of the local-polynomial estimator.

- The first term in Equation 1 is the residual sum of squares.
- The second term is a *roughness penalty,* which is large when the integrated second derivative of the regression function $m''(x)$ is large — that is, when $m(x)$ is 'rough' (with rapidly changing slope). The endpoints of the integral enclose the data.
- At one extreme, when the smoothing constant is set to $h = 0$ (and if all the x-values are distinct), $\widehat{m}(x)$ simply interpolates the data; this is similar to a local-regression estimate with span $= 1/n$.
- At the other extreme, if h is very large, then $\widehat{m}$ will be selected so that $\widehat{m}''(x)$ is everywhere 0, which implies a globally linear least-squares fit to the data (equivalent to local regression with infinitely wide neighborhoods).

The function $\widehat{m}(x)$ that minimizes Equation 1 is a natural cubic spline with knots at the distinct observed values of x.[8] Although this result seems to imply that n parameters are required (when

[7] If you're unfamiliar with calculus, the bottom line here is that the first term in $\text{SS}^*(h)$ is the residual sum of squares and the second term is a penalty for "wiggliness."

[8] *Splines* are piece-wise polynomial functions that fit together (at *knots*); for cubic splines, the first and second derivatives are also continuous at the knots. Natural splines place two additional knots at the ends of the data, and constrain the function to be linear beyond these points. See Section 4.4.2 of the *R Companion*.

all x-values are distinct), the roughness penalty imposes additional constraints on the solution, typically reducing the *equivalent number of parameters* for the smoothing spline substantially, and preventing $\widehat{m}(x)$ from interpolating the data. Indeed, it is common to select the smoothing parameter h indirectly by setting the equivalent number of parameters for the smoother.

Because there is an explicit objective-function to optimize, smoothing splines are more elegant mathematically than local regression. It is more difficult, however, to generalize smoothing splines to multiple regression,[9] and smoothing-spline and local-regression fits with the same equivalent number of parameters are usually very similar.

An illustration appears in Figure 3, comparing a smoothing spline with a local-linear fit employing the same number of equivalent parameters (degrees of freedom). We use the `smooth.spline()` function along with a previous `loess()` model to show alternative fits (each with 3.85 equivalent parameters) to the relationship of `prestige` to `income`:

```
mod.lo.inc  # previously fit loess model

Call:
loess(formula = prestige ~ income, data = Prestige, span = 0.7,
    degree = 1)

Number of Observations: 102
Equivalent Number of Parameters: 3.85
Residual Standard Error: 11.1

plot(prestige ~ income, data=Prestige)
inc.100 <- with(Prestige, seq(min(income), max(income), len=100))  # 100 x-values
pres <- predict(mod.lo.inc, data.frame(income=inc.100))  # fitted values
lines(inc.100, pres, lty=2, lwd=2)  # loess curve
lines(with(Prestige, smooth.spline(income, prestige, df=3.85),
    lwd=2))  # smoothing spline
```

We graph the local-linear regression by using `predict()` to calculate 100 fitted values over the range of `income`. The two smooths are very similar: The broken line is the local-linear fit; the solid line is the smoothing spline.

2.3 Selecting the Smoothing Parameter

Both local-polynomial regression and smoothing splines have an adjustable smoothing parameter. This parameter may be selected by visual trial and error, picking a value that balances smoothness against fidelity to the data. More formal methods of selecting smoothing parameters typically try to minimize the mean-squared error of the fit, either by employing a formula approximating the mean-square error (e.g., so-called *plug-in* estimates), or by some form of cross-validation.

In cross-validation, the data are divided into subsets (possibly comprising the individual cases); the model is successively fit omitting each subset in turn; and then the fitted model is used to 'predict' the response for the left-out subset. Trying this procedure for different values of the smoothing parameter suggests a value that minimizes the cross-validation estimate of the mean-squared error. Because cross-validation is very computationally intensive, approximations and generalizations are often employed (see, e.g., Wood, 2000, 2004).

[9] More complicated variants, such as *thin-plate splines*, generalize more easily to multiple regression. See, e.g., Gu (2000).

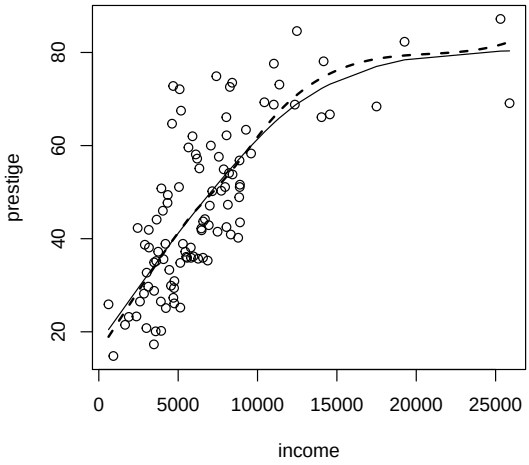

Figure 3: Local-regression (broken line) and smoothing-spline (solid line) fits for the regression of `prestige` on `income`. Both models use 3.85 equivalent parameters.

2.4 Additive Nonparametric Regression

The additive nonparametric regression model is

$$y = \beta_0 + m_1(x_1) + m_2(x_2) + \cdots + m_k(x_k) + \varepsilon$$

where the partial-regression functions m_j are fit using a simple-regression smoother, such as local polynomial regression or smoothing splines. We illustrate for the regression of `prestige` on `income` and `education`, employing the `gam()` function in the **mgcv** package (Wood, 2000, 2001, 2004, 2017):

```
library("mgcv")
```

```
Loading required package: nlme
```

```
This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.
```

```
mod.gam <- gam(prestige ~ s(income) + s(education), data=Prestige)
summary(mod.gam)
```

```

Family: gaussian
Link function: identity

Formula:
prestige ~ s(income) + s(education)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   46.833      0.689      68   <2e-16
```

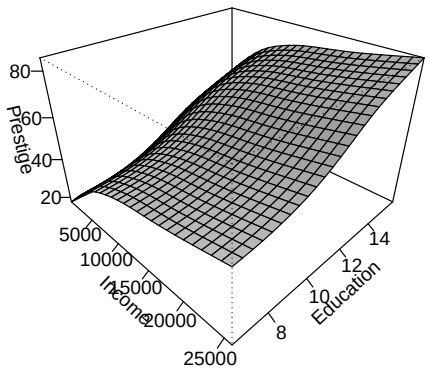

Figure 4: Fitted surface for the additive nonparametric regression of `prestige` on `income` and `education`.

```
Approximate significance of smooth terms:
               edf Ref.df    F p-value
s(income)    3.12   3.88 14.6 1.5e-09
s(education) 3.18   3.95 38.8 < 2e-16

R-sq.(adj) =  0.836   Deviance explained = 84.7%
GCV = 52.143  Scale est. = 48.414    n = 102
```

The `s()` function, used in specifying the model formula, indicates that each term is to be fit with a smoothing spline. The degrees of freedom for each term are found by generalized cross validation:[10] In this case, the equivalent of 3.118 parameters are used for the `income` term, and 3.177 for the `education` term; the degrees of freedom for the model are the sum of these plus 1 for the regression constant.

The additive regression surface is plotted in Figure 4:

```
fit.prestige <- matrix(predict(mod.gam, newdata), 25, 25)
persp(inc, ed, fit.prestige, theta=45, phi=30, ticktype="detailed",
    xlab="Income", ylab="Education", zlab="Prestige", expand=2/3,
    shade=0.5)
```

The data frame `newdata`, used to find predicted values on the regression surface, was calculated earlier to draw Figure 2 (page 7) for the *general* nonparametric multiple-regression model fit to these data. The two fits are quite similar. Moreover, because slices of the additive-regression surface in

[10]The smoothing parameters are estimated along with the rest of the model, minimizing the *generalized cross-validation criterion*,

$$\frac{n\widehat{\sigma}^2}{n - \mathrm{df}_{\mathrm{mod}}}$$

where $\widehat{\sigma}^2$ is the estimated error variance and $\mathrm{df}_{\mathrm{mod}}$ is the equivalent degrees of freedom for the model, including both parametric and smooth terms. In the *generalized* additive model (considered below) the estimated dispersion $\widehat{\phi}$ replaces the estimated error variance.

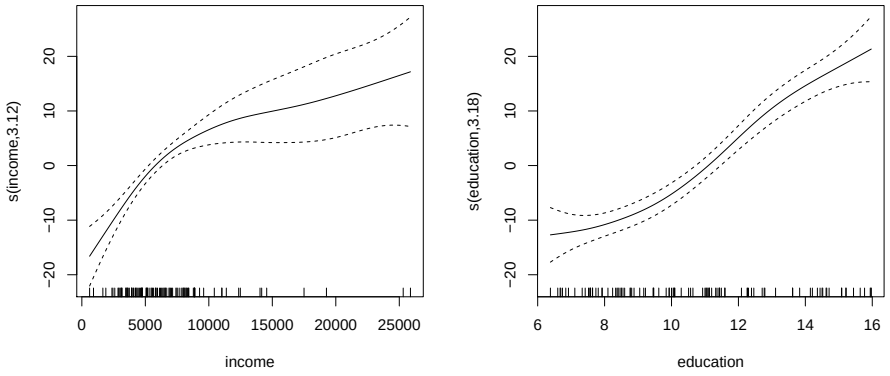

Figure 5: Partial-regression functions for the additive regression of `prestige` on `income` and `education`. The broken lines give point-wise 95-percent confidence envelopes around the fit.

the direction of one predictor (holding the other predictor constant) are parallel, it suffices to graph each partial-regression function separately. This is the practical virtue of the additive-regression model: It reduces a multidimensional (in this case, only three-dimensional) regression problem to a series of two-dimensional partial-regression graphs. The `plot()` method for `"gam"` objects produces these graphs, showing a point-wise 95-percent confidence envelope around the fit (Figure 5):

```
plot(mod.gam)
```

The `gam` function is considerably more general than this example illustrates:

- The model can include smooth (interaction) terms in two or more predictors, for example, of the form `s(income, education)`.
- The model can be semi-parametric, including linear terms — for example, `prestige ~ s(income) + education`.
- Certain technical options, such as the kinds of splines employed, may be selected by the user, and the user can fix the degrees of freedom for smooth terms.
- As its name implies (GAM = *generalized* additive model), the `gam()` function is not restricted to models with normal errors and an identity link (see below).

Hastie and Tibshirani's (1990) `gam` function in the **gam** package predates and differs from the `gam` function in the `mgcv` package: First, it is possible to fit partial-regression functions by local polynomial regression, using the `lo` function in a model formula, as well as by smoothing splines, using `s`. Second, the smoothing parameter for a term (the span for a local regression, or the degrees of freedom for a smoothing spline) is specified directly rather than determined by generalized cross-validation. As in the **mgcv** package, the `gam` function in the **gam** package can also fit *generalized* additive models.

3 Generalized Nonparametric Regression

We will illustrate generalized nonparametric regression by fitting a logistic semiparametric additive regression model to Mroz's labor-force participation data (described in Chapter 6 of the *R Companion*, and included in the **carData** package). Recall that the response variable in this data set, `lfp`,

is a factor coded `yes` for women in the labor-force, and `no` for those who are not. The predictors include number of children five years of age or less (`k5`); number of children between the ages of six and 18 (`k618`); the woman's `age`, in years; factors indicating whether the woman (`wc`) and her husband (`hc`) attended college, `yes` or `no`; and family income (`inc`), excluding the wife's income and given in $1000s. We ignore the remaining variable in the data set, the log of the wife's expected wage rate, `lwg`; as explained in the *Companion*, the peculiar definition of `lwg` makes its use problematic.

Because `k5` and `k618` are discrete, with relatively few distinct values, we will treat these predictors as factors, modeling them parametrically, along with the factors `wc` and `hc`; as well, because there are only 3 individuals with 3 children under 5, and only 3 with more than 5 children between 6 and 18, we use the `recode()` function in the **car** package to recode these unusual values:

```
library("car")
remove(list=objects())  # clean up everything
Mroz$k5f <- factor(Mroz$k5)
Mroz$k618f <- factor(Mroz$k618)
Mroz$k5f <- recode(Mroz$k5f, "3 = 2")
Mroz$k618f <- recode(Mroz$k618f, "6:8 = 5")
mod.1 <- gam(lfp ~ s(age) + s(inc) + k5f + k618f + wc + hc,
    family=binomial, data=Mroz)
summary(mod.1)
```

```
Family: binomial
Link function: logit

Formula:
lfp ~ s(age) + s(inc) + k5f + k618f + wc + hc

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.542      0.180    3.01   0.0026
k5f1          -1.521      0.251   -6.07  1.3e-09
k5f2          -2.820      0.500   -5.64  1.7e-08
k618f1        -0.342      0.227   -1.51   0.1316
k618f2        -0.279      0.248   -1.12   0.2608
k618f3        -0.333      0.284   -1.17   0.2415
k618f4        -0.531      0.440   -1.21   0.2269
k618f5        -0.491      0.609   -0.81   0.4206
wcyes          0.980      0.223    4.39  1.1e-05
hcyes          0.159      0.206    0.77   0.4422

Approximate significance of smooth terms:
        edf Ref.df Chi.sq p-value
s(age) 1.67   2.09   26.2 2.9e-06
s(inc) 1.74   2.19   17.6 0.00023

R-sq.(adj) =  0.128   Deviance explained = 10.9%
UBRE = 0.25363  Scale est. = 1         n = 753
```

Summarizing the `"gam"` object shows coefficient estimates and standard errors for the parametric part of the model; the degrees of freedom used for each smooth term (in the example, for `age` and

inc), and a hypothesis test for the term; and several summary statistics, including the UBRE score for the model.[11]

The anova() function applied to a single "gam" object reports Wald tests for the terms in the model:

```
anova(mod.1)
```

```
Family: binomial
Link function: logit

Formula:
lfp ~ s(age) + s(inc) + k5f + k618f + wc + hc

Parametric Terms:
      df Chi.sq p-value
k5f    2  55.61 8.4e-13
k618f  5   3.28    0.66
wc     1  19.26 1.1e-05
hc     1   0.59    0.44

Approximate significance of smooth terms:
        edf Ref.df Chi.sq p-value
s(age) 1.67   2.09   26.2 2.9e-06
s(inc) 1.74   2.19   17.6 0.00023
```

The plot() method for "gam" objects graphs the smooth terms in the model, along with pointwise 95-percent confidence envelopes (Figure 6):

```
plot(mod.1)
```

The departures from linearity are not great. Moreover, the regression function for inc is very imprecisely estimated at the right, where data values are sparse, and we would probably have done well do transform inc by taking logs prior to fitting the model.

One use of additive regression models, including generalized additive models, is to test for nonlinearity: We may proceed by contrasting the deviance for a model that fits a term nonparametrically with the deviance for an otherwise identical model that fits the term linearly. To illustrate, we replace the smooth term for age in the model with a linear term:

```
mod.2 <- gam(lfp ~ age + s(inc) + k5f + k618f + wc + hc,
    family=binomial, data=Mroz)
anova(mod.2, mod.1, test="Chisq")

Analysis of Deviance Table

Model 1: lfp ~ age + s(inc) + k5f + k618f + wc + hc
Model 2: lfp ~ s(age) + s(inc) + k5f + k618f + wc + hc
  Resid. Df Resid. Dev   Df Deviance Pr(>Chi)
1       740        919
2       739        917 1.16     2.21     0.17
```

[11] For a binomial model, by default gam() minimizes the UBRE (unbiased risk estimator) criterion (Wahba, 1990) in place of the GCV criterion.

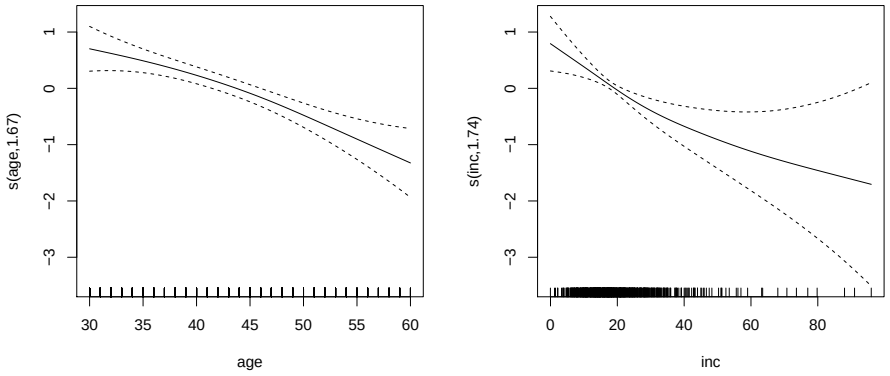

Figure 6: Smooth terms for `age` and `inc` in a semi-parametric generalized additive model for Mroz's labor-force participation data.

Likewise, we can test for nonlinearity in the `inc` (income) effect:

```
mod.3 <- gam(lfp ~ s(age) + inc + k5f + k618f + wc + hc,
    family=binomial, data=Mroz)
anova(mod.3, mod.1, test="Chisq")
```

```
Analysis of Deviance Table

Model 1: lfp ~ s(age) + inc + k5f + k618f + wc + hc
Model 2: lfp ~ s(age) + s(inc) + k5f + k618f + wc + hc
  Resid. Df Resid. Dev   Df Deviance Pr(>Chi)
1       740        920
2       739        917 1.26     2.55     0.15
```

Both tests have p-values between 0.05 and 0.1, providing only weak evidence against the null hypotheses of linearity.[12]

Similarly, we can test the statistical significance of a term in the model by dropping it and noting the change in the deviance. For example, to test the `age` term:

```
mod.4 <- update(mod.1, . ~ . - s(age))
anova(mod.4, mod.1, test="Chisq")
```

```
Analysis of Deviance Table

Model 1: lfp ~ s(inc) + k5f + k618f + wc + hc
Model 2: lfp ~ s(age) + s(inc) + k5f + k618f + wc + hc
  Resid. Df Resid. Dev   Df Deviance Pr(>Chi)
1       741        945
2       739        917 1.84     27.5  8.1e-07
```

Thus, the `age` effect is associated with a very small p-value.[13] Compare this with the similar result

[12] We invite the reader to perform tests for nonlinearity for the factors `k5f` and `k618f`.

[13] Applied to models in which the degree of smoothing is selected by GCV or UBRE, rather than fixed, these tests tend to exaggerate the "statistical significance" of terms in the model.

for the Wald test for `age`, presented above. We leave it to the reader to perform similar tests for the other predictors in the model, including `inc` and the parametric terms.

4 Complementary Reading and References

Nonparametric regression is taken up in Fox (2016, Chap. 18).

All of the nonparametric regression models discussed in this appendix (and some others, such as *projection-pursuit regression*, and *classification and regression trees*) are described in Fox (2000a,b), from which the examples appearing in the appendix are adapted.

The excellent and wide-ranging books by Hastie and Tibshirani (1990) and Wood (2017) are associated respectively with the **gam** and **mgcv** packages, the latter a part of the standard R distribution. A briefer treatment of GAMs and the `gam()` function in the **gam** package appears in a paper by Hastie (1992).

References

Bowman, A. W. and Azzalini, A. (1997). *Applied Smoothing Techniques for Data Analysis: The Kernel Approach With S-Plus Illustrations*. Oxford University Press, Oxford.

Fox, J. (2000a). *Multiple and Generalized Nonparametric Regression*. Thousand Oaks, CA.

Fox, J. (2000b). *Nonparametric Simple Regression: Smoothing Scatterplots*. Thousand Oaks, CA.

Fox, J. (2016). *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks CA, third edition.

Fox, J. and Weisberg, S. (2019). *An R Companion to Applied Regression*. Sage, Thousand Oaks, CA, third edition.

Gu, C. (2000). Multidimensional smoothing with smoothing splines. In Schmiek, M. G., editor, *Smoothing and Regression: Approaches, Computation, and Applications*. Wiley, New York.

Hastie, T. J. (1992). Generalized additive models. In Chambers, J. M. and Hastie, T. J., editors, *Statistical Models in S*, pages 421–454. Wadsworth, Pacific Grove, CA.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*. Chapman and Hall, London.

Loader, C. (1999). *Local Regression and Likelihood*. Springer, New York.

Nason, G. (2008). *Wavelet Methods in Statistics with R*. New York.

Nason, G. (2016). *wavethresh: Wavelets Statistics and Transforms*. R package version 4.6.8.

Nason, G. P. and Silverman, B. W. (1994). The discrete wavelet transform in S. *Journal of Computational and Graphical Statistics*, 3:163–191.

Nason, G. P. and Silverman, B. W. (2000). Wavelets for regression and other statistical problems. In Schmiek, M. G., editor, *Smoothing and Regression: Approaches, Computation, and Applications*. Wiley, New York.

Wahba, G. (1990). *Spline Models for Observational Data*. SIAM, Philadelphia.

Wood, S. N. (2000). Modelling and smoothing parameter estimation with multiple quadratic penalties. *Journal of the Royal Statistical Socieity, Series B*, 62:413–428.

Wood, S. N. (2001). mgcv: GAMS and generalized ridge regression for R. *R News*, 1(2):20–25.

Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistics Association*, 99:673–686.

Wood, S. N. (2017). *Generalized Additive Models: An Introduction with R*. Chapman and Hall, Boca Raton, FL, second edition.