7 Moving Beyond Linearity

So far we have mostly focused on linear models. Linear models are relatively simple to describe and implement, and have advantages over other approaches in terms of interpretation and inference. However, standard linear regression can have significant limitations in terms of predictive power. This is because the linearity assumption is almost always an approximation, and sometimes a poor one. In this chapter we relax the linearity assumption while still attempting to maintain as much interpretability as possible. We do this by examining very simple extensions of linear models like polynomial regression as well as more sophisticated approaches such as additive models and more generally generalized additive models (GAMs).

- Polynomial regression extends the linear model by adding extra predictors, obtained by raising each of the original predictors to a power. For example, a cubic regression uses three variables, X, X^2, and X^3, as predictors. This approach provides a simple way to provide a nonlinear fit to data.
- GAM is an additive modeling technique where the impact of the predictive variables is captured through smooth functions.

7.1 Polynomial regression

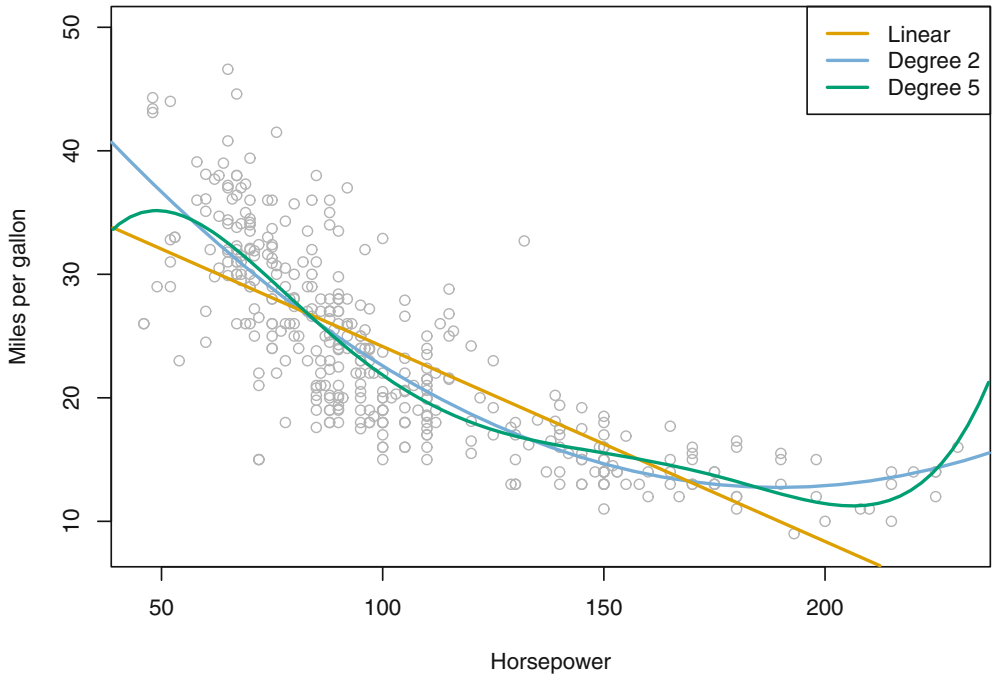

Figure 25: The `Auto` data set. For a number of cars, `mpg` and `horsepower` are shown. The linear regression fit is shown in orange. The linear regression fit for a model that includes `horsepower`^{2} is shown as a blue curve. The linear regression fit for a model that includes all polynomials of `horsepower` up to fifth-degree is shown in green.

Consider Figure 25, in which the `mpg` (gas mileage in miles per gallon) versus `horsepower` is shown for a number of cars in the `Auto` data set. The orange line represents the linear regression fit. There

	Coefficient	Std. error	t-statistic	p-value
`Intercept`	56.9001	1.8004	31.6	< 0.0001
`horsepower`	-0.4662	0.0311	-15.0	< 0.0001
`horsepower`^{2}	0.0012	0.0001	10.1	< 0.0001

Table 17: For the `Auto` data set, least squares coefficient estimates associated with the regression of `mpg` onto `horsepower` and `horsepower`^{2}.

is a pronounced relationship between `mpg` and `horsepower`, but it seems clear that this relationship is in fact non-linear: the data suggest a curved relationship. A simple approach for incorporating non-linear associations in a linear model is to include transformed versions of the predictors in the model. For example, the points in Figure 25 seem to have a quadratic shape, suggesting that a model of the form

$$\texttt{mpg} = \beta_0 + \beta_1 \texttt{horsepower} + \beta_2 \texttt{horsepower}^2 + \epsilon \tag{32}$$

may provide a better fit. Equation (32) involves predicting `mpg` using a non-linear function of `horsepower`. But it is still a linear model! That is, (32) is simply a multiple linear regression model with $X_1 =$ `horsepower` and $X_2 =$ `horsepower`^{2}. So we can use standard linear regression software to estimate β_0, β_1 and β_2 in order to produce a non-linear fit. The blue curve in Figure 25 shows the resulting quadratic fit to the data. The quadratic fit appears to be substantially better than the fit obtained when just the linear term is included. The R^2 of the quadratic fit is 0.688, compared to 0.606 for the linear fit, and the p-value in Table 17 for the quadratic term is highly significant.

If including `horsepower`^{2} led to such a big improvement in the model, why not include `horsepower`^{3}, `horsepower`^{4}, or even `horsepower`^{5}? The green curve in Figure 25 displays the fit that results from including all polynomials up to fifth degree in the model (32). The resulting fit seems unnecessarily wiggly - that is, it is unclear that including the additional terms really has led to a better fit to the data.

The approach that we have just described for extending the linear model to accommodate non-linear relationships is known as polynomial regression, since we have included polynomial functions of the predictors in the regression model.

In general a polynomial regression can be written as

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$

For large enough degree d, a polynomial regression allows us to produce an extremely non-linear curve. Notice that the coefficients can be easily estimated using least squares linear regression. Generally speaking, it is unusual to use d greater than 3 or 4 because for large values of d, the polynomial curve can become overly flexible and can take on some very strange shapes. This is especially true near the boundary of the X variable. In particular, polynomials do not seem to have good properties in terms of approximating an underlying function over its whole domain.

7.1.1 Lab I: Polynomial regression

The `lm()` function can also accommodate non-linear transformations of the predictors. For instance, given a predictor X, we can create a predictor X^2 using `I(X^2)`. The function `I()` is needed since the `^` has a special meaning in a formula; wrapping as we do allows the standard usage in `R`, which is to raise X to the power 2. We now perform a regression of `medv` onto `lstat` and `lstat^2`.

```
library(MASS)
lm.fit2 <- lm(medv ~ lstat +I(lstat ^2), data = Boston)
summary(lm.fit2)
```

The near-zero p-value associated with the quadratic term suggests that it leads to an improved model. We use the `anova()` function to further quantify the extent to which the quadratic fit is superior to the linear fit

```
lm.fit <- lm(medv ~ lstat, data = Boston)
anova(lm.fit, lm.fit2)
```

Here Model 1 represents the linear submodel containing only one predictor, `lstat`, while Model 2 corresponds to the larger quadratic model that has two predictors, `lstat` and `lstat^2`. The `anova()` function performs a hypothesis test comparing the two models. The null hypothesis is that the two models fit the data equally well, and the alternative hypothesis is that the full model is superior. Here the F-statistic is 135 and the associated p-value is virtually zero. This provides very clear evidence that the model containing the predictors `lstat` and `lstat^2` is far superior to the model that only contains the predictor `lstat`. This is not surprising, since earlier we saw evidence for non-linearity in the relationship between `medv` and `lstat`. If we type

```
par(mfrow = c(2,2))
plot(lm.fit2)
```

then we see that when the `lstat^2` term is included in the model, there is little discernible pattern in the residuals.

In order to create a cubic fit, we can include a predictor of the form `I(X^3)`. However, this approach can start to get cumbersome for higher order polynomials. A better approach involves using the `poly()` function to create the polynomial within `lm()`. For example, the following command produces a fifth-order polynomial fit:

```
lm.fit5 <- lm(medv ~ poly(lstat ,5), data = Boston)
summary(lm.fit5)
```

This suggests that including additional polynomial terms, up to fifth order, leads to an improvement in the model fit! However, further investigation of the data reveals that no polynomial terms beyond fifth order have significant p-values in a regression fit.

7.1.2 Lab II: Polynomial regression

This section shows using simulated data polynomials behaviour. Let's consider a simple example. First, let's create a data frame and fill it with some simulated data with an obvious non-linear trend and compare how well some models fit to that data.

```
set.seed(100)
x <- seq(0, pi * 2, 0.1)
sin_x <- sin(x)
```

```
y <- sin_x + rnorm(n = length(x), mean = 0, sd = sd(sin_x / 2))
Sample_data <- data.frame(y,x)

plot(x,y)
```

We are interested in modelling `y` as a function of `x`. We fit a polynomial model. Perhaps we should add square, cubic, etc. terms and stop when we think we have a good fit. This can be done by using

```
degree.p <- 2 # or 3, 4 ...
lm.poly2 <- lm(y ~ poly(x, degree.p), data = Sample_data)
```

Let's fit polynomial models with different degrees.

```
x11()
par(mfrow = c(2, 3))

degree.p <- c(2, 4, 6, 8, 10, 20)

for(i in 1:length(degree.p)){

m2 <- lm(y  ~ poly(x, degree.p[i]), data = Sample_data)

plot(Sample_data$x, Sample_data$y)
lines(sort(Sample_data$x), fitted(m2)[order(Sample_data$x)], lwd = 2)

}
```

Polynomials do not seem to have good properties in terms of approximating an underlying function over its whole domain.

7.2 Additive Models

7.2.1 Introducing a Smooth Function

Let us rewrite our simple regression model in a slight more general way.

So replace

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

with

$$y_i = \beta_0 + f(x_i) + \epsilon_i, \tag{33}$$

where f is a smooth function of x. We can approximate $f(x_i)$ by

$$f(x_i) = \sum_{k=1}^{d} \beta_k b_k(x_i),$$

where the b_k are known basis functions, and β_k unknown regression parameters.

If $f(x_i)$ is believed to be a 3^{th} order polynomial (so $d = 3$) then

$$b_1(x_i) = x_i,\ b_2(x_i) = x_i^2,\ b_3(x_i) = x_i^3,$$

which brings us back to a polynomial model;

$$f(x_i) = \sum_{k=1}^{3} \beta_k b_k(x_i) = \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3.$$

However, we have already seen that polynomials have problems when approximating the true function. In particular their approximations of unknown true functions are concerned with getting good approximations in the vicinity of some particular point of interest, approximation will eventually become very poor as we move away from that point. For this reason we can use bases that have better theoretical and numerical properties. An example of basis with good theoretical and computational properites are the thin plate regression spline (TPRS) basis.

A Thin Plate Regression Spline (TPRS) Basis

Since the expression of the TPRS are very complex, we only report a representation of a thin plate regression spline basis functions forming a smooth function of one variable (see Figure 25). The first 7 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth curve in the lower right panel. The first two basis functions span the space of functions that are completely smooth, according to the wiggliness measure. The remaining basis functions represent the wiggly component of the smooth curve.

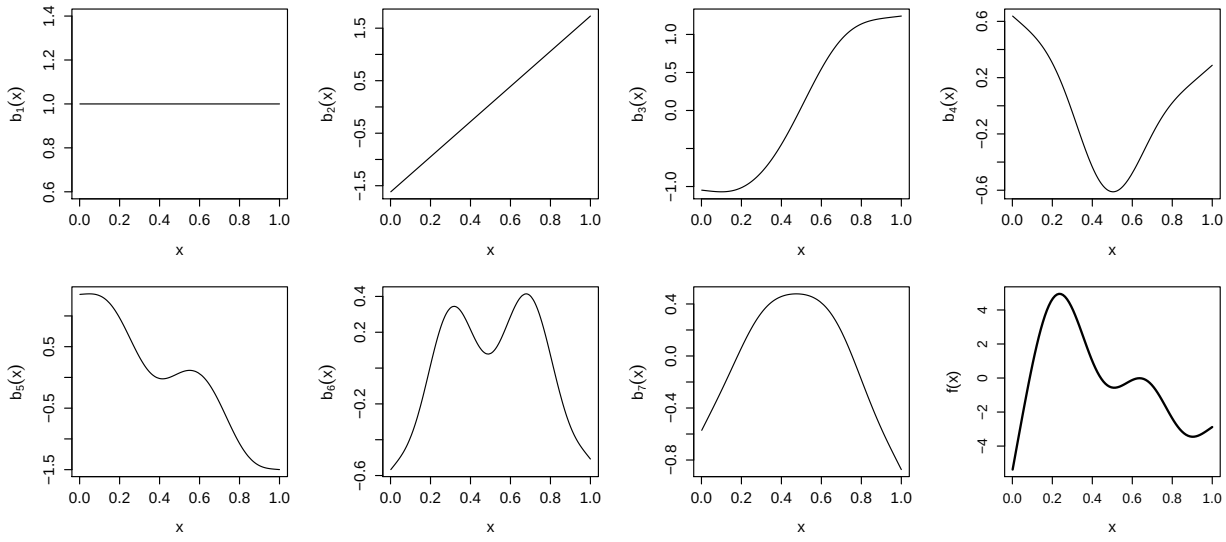

Figure 26: TPRS: seven basis functions and a smooth function of one variable.

There are many types of spline bases; e.g., B-splines, cubic regression splines. They all typically lead to the same results in the one-dimensional smooth function case.

But how should we choose d, the number of basis? A convenient and theoretically founded way of choosing the degree of smoothing is to keep the basis dimension d fixed at a reasonable large number

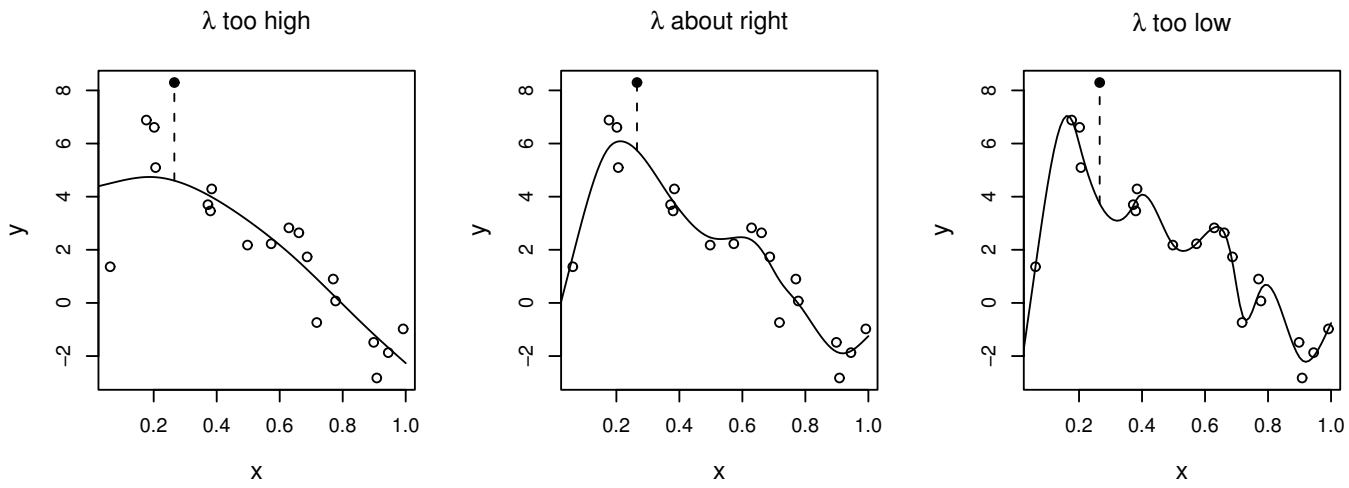

Figure 27: Illustration of the principle behind cross validation.

and add a roughness penalty to the least squares fitting objective. That is, minimise

$$\sum_{i=1}^{n} \{y_i - \beta_0 - f(x_i)\}^2 + \lambda \int [f''(x)]^2 dx.$$

The second term measure the roughness of the smooth functions, $\lambda \geq 0$ is the smoothing parameter.

7.2.2 Smoothing Parameter

The trade off between fit and smoothness is controlled by smoothing parameter λ. In particular:

- $\lambda \to \infty$ leads to a straight line estimate; all parameters associated with the wiggly bases will be estimated as 0.
- $\lambda = 0$ gives an un-penalized (potentially very wiggly) regression spline estimate; the parameters of all wiggly bases will be kept in the model.

So it is really crucial to be able to tune λ such that it reaches a balance in terms of fit and smoothness.

Smoothing Parameter Estimation

Estimation of the smoothing parameter is achieved using cross validation. Specifically, minimise

$$\frac{1}{n} \sum_{i=1}^{n} \left\{ y_i - \beta_0 - \hat{f}^{[-i]}(x_i) \right\}^2,$$

where $\hat{f}^{[-i]}$ refers to the model fitted to all data except y_i.

Figure 26 illustrates the principle of cross validation in this context. The fifth datum (•) has been omitted from fitting and the continuous curve shows a penalized regression spline fitted to the remaining data (○). When the smoothing parameter is too high the spline fits many of the data poorly and

does no better with the missing point. When λ is too low the spline fits the noise as well as the signal and the consequent extra variability causes it to predict the missing datum poorly. For intermediate λ the spline is fitting the underlying signal quite well, but smoothing through the noise: hence, the missing datum is reasonably well predicted. Cross validation leaves out each datum from the data in turn and considers the average ability of models fitted to the remaining data to predict the omitted data.

7.2.3 Some Remarks

- The number of effective degrees of freedom (*edf*) of a spline estimate is defined in such a way that the role of the penalty in estimation is acknowledged. For example, if a smooth function is represented using 9 bases then the *edf* of the curve estimate will be a real value between 1 and 9. The higher the *edf* value the higher the complexity of the estimated smooth function. For example, $edf = 1$ indicates that the estimated function is linear.
- If λ is chosen larger than needed then the resulting estimated function may miss genuine wiggles in the underlying function. Therefore, there can be a bit of bias in the results. As a consequence, confidence intervals may not be reliable although they can still provide a useful indication of uncertainty.
- Using $\hat{f}$ and an estimate of the mean squared error for $\hat{f}$, point-wise Bayesian confidence intervals at the points on the estimated function can be easily constructed.
- Testing null hypothesis that a linear model is appropriate, $H_0 : f = 0$, can be achieved by employing an F-test.
- Model comparison can be achieved using AIC

  $$\text{AIC} = n \log(\text{RSS}/n) + 2edf + C,$$

  where $\text{RSS} = \sum_{i=1}^{n} \left\{ y_i - \hat{\beta}_0 - \hat{f}(x_i) \right\}^2$ and C is a constant that can be ignored in model comparisons.
- Model assumptions can be tested in a similar fashion as traditionally done for linear models using residual analysis.
- Many of the definitions and results showed for model (33) can be extended to the case where there are multiple covariates. In this case model (33)can be replaced with:

  $$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_k(x_{ik}) + \epsilon_i,$$

  where $f_1(\cdot), f_2(\cdot), \ldots, f_k(\cdot)$ are smooth functions as defined in section 7.2.1.

7.2.4 Lab: Thin Plate Regression Spline

Let's see the TPRS in action within `R` using the previous simulated dataset `Sample_data`. The function we use to fit an additive model is `gam()` from package `mgcv`.

```
library(mgcv)
no.b <- 3 # or 4, 5 ...
gam(y ~ s(x, bs = "tp", k = no.b, fx = TRUE))
```

where `s()` denotes the smooth function, `bs` indicates the kind of spline basis employed and `k` the number of bases, `fx=TRUE` means that no penalty is applied. Let's fit TPRS models with several values for `k`.

```
x11()
par(mfrow = c(2, 3))

no.b <- c(2, 3, 4, 5, 6, 7) + 1

for(i in 1:length(no.b)){

m3 <- gam(y  ~ s(x, bs = "tp", k = no.b[i], fx = TRUE), data = Sample_data)

plot(Sample_data$x, Sample_data$y )
lines(sort(Sample_data$x), fitted(m3)[order(Sample_data$x)], lwd = 2)

}
```

Notice how the estimated smooth function becomes more flexible as we add more basis functions.

Let's now look at the behaviour of λ. Let's fit penalised spline models with several values for λ.

```
x11()
par(mfrow = c(1, 3))

sps <- c(0.00001, 1, 1000)

for(i in 1:length(sps)){

m4 <- gam(y  ~ s(x, bs = "tp", k = 30), sp = sps[i], data = Sample_data)

plot(Sample_data$x, Sample_data$y )
lines(sort(Sample_data$x), fitted(m4)[order(Sample_data$x)], lwd = 2)

}
```

We can see that for small values of λ we obtain estimated functions that are too wiggly (no penalty is applied), for big values of λ we get function that are too smooth (we are penalizing too much). If we choose the right amount of smoothness we get functions that are neither too wiggly nor a too smooth.

Let's fit a penalised spline model with automatic λ estimation (based on cross-validation criterion).

```
par(mfrow = c(1, 1))

m5 <- gam(y  ~ s(x, bs = "tp", k = 30), data = Sample_data)

plot(Sample_data$x, Sample_data$y)
lines(sort(Sample_data$x), fitted(m5)[order(Sample_data$x)], lwd = 2)
```

```
m5$sp # estimated smoothing parameter

# Confidence intervals and p-values

plot(m5, residuals = TRUE, shade = TRUE)
summary(m5)
AIC(m5)
```

We can see that $\hat{\lambda} = 0.04$.

7.3 Generalized Additive Models

If the response variable follows an exponential family distribution and we want to model non-linearities with smooth functions, then we can use the generalised additive models (GAMs). All the description and results that we have seen for the additive models can be adapted to GAMs. However, because we are dealing with non-Gaussian responses, then the penalised least squares fitting objective

$$\sum_{i=1}^{n} \{y_i - \beta_0 - f(x_i)\}^2 + \lambda \int [f''(x)]^2 dx,$$

can be replaced with

$$\ell + \lambda \int [f''(x)]^2 dx,$$

where ℓ is the likelihood function of the model. Maximization of the penalised likelihood produces estimates which can be used to construct the estimated curve. λ can be estimated using modifications of the methods seen previously. Of course, this approach can be also be generalised to the case with many and different covariates. The details of GAMs are beyond this course.

7.3.1 Lab: Generalized Additive Model

The data set `pisasci2006` we consider for this analysis has been constructed using average Science scores by country from the Programme for International Student Assessment (PISA) 2006, along with GNI per capita (Purchasing Power Parity, 2005 dollars), Educational Index, Health Index, and Human Development Index from UN data. The key variables are as follows:

- Overall Science Score (average score for 15 year olds)
- Interest in science
- Support for scientific inquiry
- Income Index
- Health Index
- Education Index
- Human Development Index (composed of the Income index, Health Index, and Education Index)

We start with the simple situation of a single predictor. Let's begin by using a simple linear regression to predict science scores by the Income index. We could use the standard `lm()` function, but I'll leave that as an exercise for comparison. We can still do straightforward linear models with the `gam()` function, and again it is important to note that the standard linear model can be seen as a special case of a GAM.

```
library(mgcv)
pisa <- read.csv("pisasci2006.csv")
mod_lm <- gam(Overall ~ Income, data = pisa)
summary(mod_lm)
```

We are getting here the same thing you get from a regular linear model, because you just ran one. However, there are a couple things to look at. The coefficient is statistically significant. We also see the deviance explained, which serves as a generalization of R-squared, and in this case, it actually is equivalent to the unadjusted R-squared. Likewise, there is the familiar adjusted version of it to account for small sample size and model complexity. The scale estimate is the scaled deviance, which here is equivalent to the residual sums of squares. The GCV score is the generalized cross validation score. The smaller this value the better the model.

Let's now fit an actual generalized additive model using TPRS as our basis. We again use the `gam()` function as before for basic model fitting, but now we are using a function `s()` within the formula to denote the smooth terms.

```
mod_gam1 <- gam(Overall ~ s(Income), data = pisa)
summary(mod_gam1)
```

Fist thing to note is that, aside from the smooth part, our model code is similar to what we're used to with core R functions such as `lm()` and `glm()`. In the summary, we first see the distribution assumed, as well as the link function used, in this case normal and identity, respectively, which to iterate, had we had no smoothing, would result in a simple linear regression. After that we see that the output is separated into parametric and smooth, or nonparametric parts. In this case, the only parametric component is the intercept, but it's good to remember that you are not bound to smooth every effect of interest, and indeed, as we will discuss in more detail later, part of the process may involve refitting the model with terms that were found to be linear for the most part anyway. The smooth component of our model regarding a country's income and its relationship with overall science score suggests it is statistically significant.

Now, Let's see what we can do with a more realistic case where we have added model complexity. We will start with a multiple linear regression, this time adding the Health and Education indices.

```
mod_lm2 <- gam(Overall ~ Income + Edu + Health, data=pisa)
summary(mod_lm2)
```

I appears we have statistical effects for Income and Education, but not for Health, and the adjusted R-squared suggests a notable amount of the variance is accounted for. Lets see about nonlinear effects.

As far as the generalized additive model goes, we can approach things in a similar manner as before. We will ignore the results of the linear model for now and look for nonlinear effects for each covariate.

```
mod_gam2 <- gam(Overall ~ s(Income) + s(Edu) + s(Health), data = pisa)
summary(mod_gam2)
```

There are a couple of things to take note of. First, statistically speaking, we come to the same conclusion as the linear model regarding the individual effects. One should take particular note of the effect of Health index. The effective degrees of freedom with value 1 suggests that it has essentially been reduced to a simple linear effect. The following will update the model to explicitly model the effect as such, but as one can see based on GCV, the results are identical.

```
mod_gam2B = update(mod_gam2, .~.-s(Health) + Health, data = pisa)
summary(mod_gam2B)
```

We also note that this model accounts for much of the variance in Overall science scores, with an adjusted R-squared of .86. In short, it looks like the living standards and educational resources of a country are associated with overall science scores, even if we don't really need the Health index in the model.

Now we examine the effects of interest visually via the function `plot()`.

```
plot(mod_gam2B, pages = 1, seWithMean = TRUE, shade = TRUE, scale = 0)
```

We can see the effects of interest. We see the tapering off of Income's effect at its highest levels, and in addition, a kind of sweet spot for a positive effect of Education in the mid-range values, with a slight positive effect overall.

One can utilize smooths of more than one variable, in effect, a smooth of the smooths of the variables that go into it. This is akin to an interaction in typical model settings. Let's create a new model to play around with this feature. After fitting the model, we provide a visualization in 3D.

```
mod_gam3 <- gam(Overall ~ te(Income, Edu), data  =pisa)
summary(mod_gam3)

vis.gam(mod_gam3, type = 'response', plot.type = 'persp',
        phi = 30, theta = 30, n.grid = 500, border = NA)
```

In the above we are using a type of smooth called a tensor product smooth, and by smoothing the marginal smooths of Income and Education, we see a bit clearer story. As we might suspect, wealthy countries with more of an apparent educational infrastructure are going to score higher on the Overall science score. However, wealth alone does not necessarily guarantee higher science scores (note the dark bottom right corner on the contour plot), though without at least moderate wealth hopes are fairly dim for a decent score.