An Introduction to Splines

James H. Steiger

Department of Psychology and Human Development
Vanderbilt University

An Introduction to Splines

Introduction

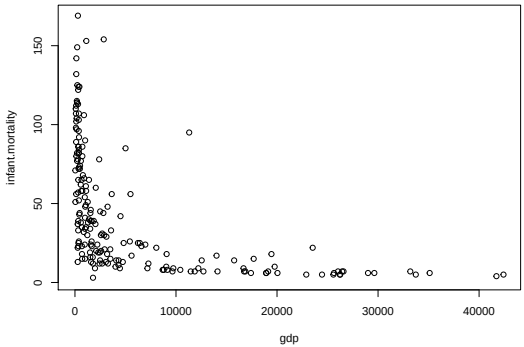
- When transformation won't linearize your model, the function is complicated, and you don't have deep theoretical predictions about the nature of the X-Y regression relationship, but you do want to be able to characterize it, at least to the extent of predicting new values, you may want to consider a generalized additive model (GAM).
- A generalized additive model represents $E(Y|X = x)$ as a weight sum of smooth functions of x.
- We'll briefly discuss two examples, polynomial regression and spline regression.

Piecewise Regression

- Nonlinear relationships between a predictor and response can sometimes be difficult to fit with a single parameter function or a polynomial of "reasonable" degree, say, between 2 and 5.
- For example, you are already familiar with the UN data relating per capita GDP with infant mortality rates per 1000. We've seen before that these data are difficult to analyze in their original form, but can be linearized by log-transforming both the predictor and response.
- Here are the original data from `car`.

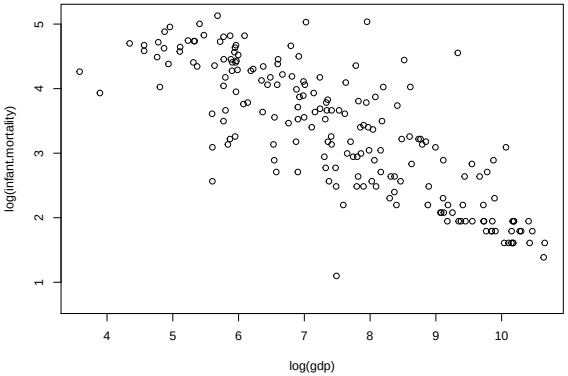
Piecewise Regression

```
> data(UN)
> attach(UN)
> plot(gdp,infant.mortality)
```



Piecewise Regression

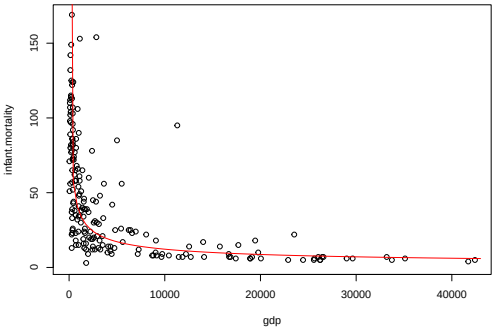
```
> plot(log(gdp),log(infant.mortality))
```



Piecewise Regression

Here we fit the log-log model, then back-transform it to the original metric and plot the curve.

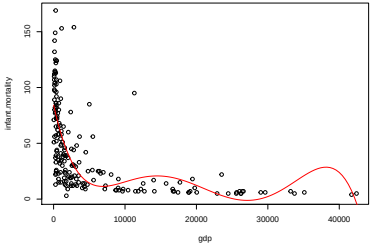
```
> loglog.fit <- lm(I(log(infant.mortality)) ~ I(log(gdp)))
> plot(gdp,infant.mortality)
> curve(exp(coef(loglog.fit)[1] + coef(loglog.fit)[2]*log(x)),5,43000,add=T,col="red")
```



Piecewise Regression

This works quite a bit better than, say, fitting a polynomial of order 5, because polynomials can be very unstable at their boundaries!

```
> poly5.fit <- lm(infant.mortality ~ gdp + I(gdp^2)
+     + I(gdp^3) + I(gdp^4) + I(gdp^5))
> plot(gdp,infant.mortality)
> b0 <- coef(poly5.fit)[1]
> b1 <- coef(poly5.fit)[2]
> b2 <- coef(poly5.fit)[3]
> b3 <- coef(poly5.fit)[4]
> b4 <- coef(poly5.fit)[5]
> b5 <- coef(poly5.fit)[6]
> curve(b0+b1*x + b2*x^2 + b3*x^3 + b4*x^4 +
+   b5 * x^5, 4,43000,add=T,col="red")
```



Piecewise Regression

- Another approach is to fit more than one straight line.
- Our motivation to do this with the present data is substantive. We can see that there are many countries jammed up against the left of the plot with *gdp* values below 2000, and there is a steep decline of infant mortality as a function of *gdp* within that area of the plot. Once gdp exceeds around 2000, the decline is much less steep.
- So, for example, we could fit one straight line to the data where gdp is less than or equal to 2000, and another for the data points where gdp exceeds 2000.
- We already know how to do this!

Piecewise Regression

- Define an indicator variable, and then use it as a predictor, but also allow an interaction between this dummy predictor and gdp
- We can express the model as

$$\begin{aligned} E(child.mortality|gdp) &= \beta_0 + \beta_1 gdp + \beta_2 (gdp > 2000)_+ \\ &\quad + \beta_3 gdp (gdp > 2000)_+ \end{aligned}$$

- The dummy variable $(gdp > 2000)_+$ takes on the value 1 when $gdp > 2000$, zero otherwise. You can see that for observations where gdp exceeds 2000, the model becomes

$$E(child.mortality|gdp) = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) gdp \quad (1)$$

- What is the model when $gdp \leq 2000$? (C.P.)

Piecewise Regression

- The point of separation in the piecewise regression system is called a *knot*.
- We can have more than one knot.
- We can select the knot *a priori* (say, at the median value of the predictor), or, as in this case, we can allow the data to dictate.

Linear Spline Regression

- This system is straightforward to implement in R.
- However, the lines need not join at the knots.
- To force the lines to join, eliminate several intercept-difference parameters and define the system with k knots $a_1 \ldots a_k$ as follows:

$$\begin{aligned} E(Y|X) &= \beta_0 + \beta_1 X + \beta_2 (X - a_1)_+ + \beta_3 (X - a_2)_+ \\ &\quad + \ldots + \beta_{k-1} (X - a_k)_+ \end{aligned} \tag{2}$$

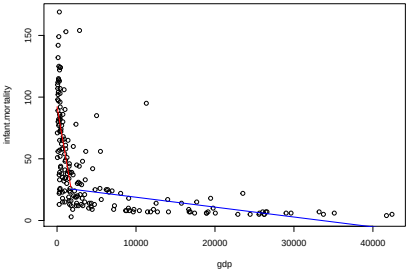
- We call this *linear spline regression*.
- The terms of the form $(u)_+$ have the value u if u is positive, and 0 otherwise.
- Let's see how this is done in R with a knot at 1750. Notice that the second line segment starts at a height equal to that of the first line at $X = 1750$.

Linear Spline Regression

```
> fit.jpw <- lm(infant.mortality ~1 + gdp + I((gdp-1750)*(gdp>1750)))
> summary(fit.jpw)

Call:
lm(formula = infant.mortality ~ 1 + gdp + I((gdp - 1750) * (gdp >
    1750)))

Residuals:
    Min      1Q  Median      3Q     Max
-69.045 -11.923  -2.760   8.761 127.998

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  92.152745   4.061900   22.69   <2e-16 ***
gdp                          -0.037298   0.003347  -11.14   <2e-16 ***
I((gdp - 1750) * (gdp > 1750))  0.036496   0.003474   10.51   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26.5 on 190 degrees of freedom
  (14 observations deleted due to missingness)
Multiple R-squared:  0.5325,        Adjusted R-squared:  0.5276
F-statistic: 108.2 on 2 and 190 DF,  p-value: < 2.2e-16
```

Linear Spline Regression

```
> b.0 <- coef(fit.jpw)[1]
> b.1 <- coef(fit.jpw)[2]
> b.2 <- coef(fit.jpw)[3]
> x.0 <- seq(0,1750,1)
> x.1 <- seq(1750,42000,1)
> y.0 <- b.0 + b.1 * x.0
> y.1 <- (b.0 + b.1 * 1750 + (b.1 + b.2)* x.1)
> plot(gdp,infant.mortality)
> lines(x.0,y.0, col="red")
> lines(x.1,y.1, col="blue")
```



Linear Spline Regression

- We didn't do that well with only two knots.
- We could probably do much better with 3 or 4.
- Another alternative is to fit different cubic functions that are connected at the knots.
- We discuss *cubic spline regression* in the next section.

Cubic Spline Regression

- Cubic spline regression fits cubic functions that are joined at a series of k knots.
- These functions will look really smooth if they have the same first and second derivatives at the knots.
- Such a system follows the form

$$\begin{aligned} E(Y|X) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \\ &\quad \beta_4 (X - a_1)_+^3 + \beta_5 (X - a_2)_+^3 + \ldots + \\ &\quad \beta_{k+3} (X - a_k)_+^3 \end{aligned} \tag{3}$$

Restricted Cubic Spline Regression

- With enough knots, cubic spline regression can work very well.
- However, like with polynomial regression, the system sometimes works very poorly at the outer ranges of X.
- A solution to this problem is to restrict the outer line segments at the lower and upper range of X to be straight lines.

Restricted Cubic Spline Regression

- To force linearity when $X < a_1$, the X^2 and X^3 terms must be eliminated.
- To force linearity when $X > a_k$, the last two βs are redundant, i.e., are just combinations of the other βs.
- Such a system with k knots $a_1 \ldots a_k$ follows the form

$$E(Y|X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_{k-1} X_{k-1} \tag{4}$$

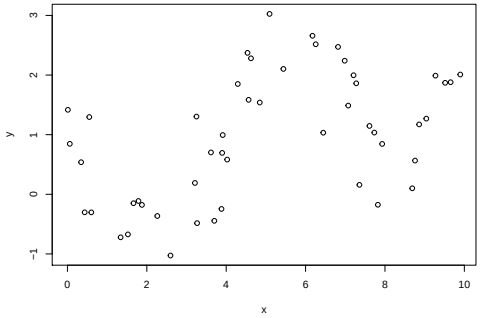
where $X_1 = X$, and, for $j = 1, \ldots, k-2$,

$$\begin{aligned} X_{j+1} &= (X - a_j)_+^3 (X - a_{k-1})_+^3 (a_k - a_j)/(a_k - a_{k-1}) \\ &\quad + (X - a_k)_+^3 (a_{k-1} - a_j)/(a_k - a_{k-1}) \end{aligned} \tag{5}$$

Restricted Cubic Spline Regression

- Here are some artificial data:

```
> set.seed(12345)
> x <- runif(50, 0, 10)
> y <- cos(x + 1) + x/5  + 0.5*rnorm(50)
> plot(x,y)
```



- In the following figures from Fox's Applied Regression text, we see a progression of fits to these data.

Restricted Cubic Spline Regression

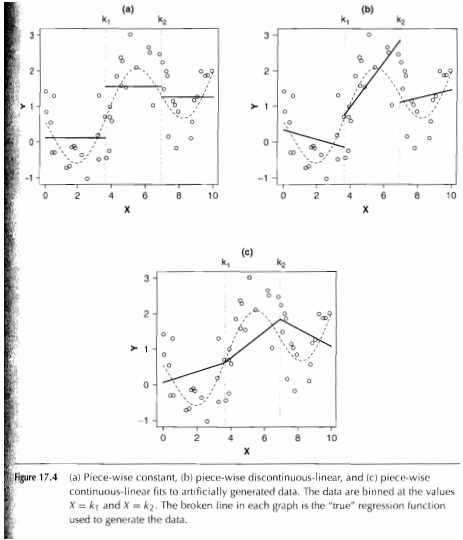


Figure 17.4 (a) Piece-wise constant, (b) piece-wise discontinuous-linear, and (c) piece-wise continuous-linear fits to artificially generated data. The data are binned at the values $X = k_1$ and $X = k_2$. The broken line in each graph is the "true" regression function used to generate the data.

Restricted Cubic Spline Regression

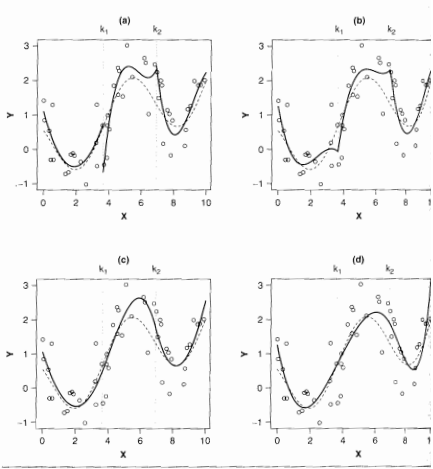


Figure 17.5 Piece-wise cubic fits to artificially generated data: (a) discontinuous; (b) continuous; (c) continuous with continuous slopes; (d) continuous with continuous slopes and curvature. The broken line in each graph is the "true" regression function used to generate the data.

Restricted Cubic Spline Regression

- The spline-fitting process can be automated by R to a large extent.
- In the code below, we select an optimal smooth and apply it to some artificial data.
- On the next slide, we show the true function in red, the data (perturbed by noise), and the result of the spline fit.
- In this case, in which we have 100 equally spaced data points, the results are excellent.

```
> library(pspline)
> n <- 100
> x <- (1:n)/n
> true <- ((exp(1.2*x)+1.5*sin(7*x))-1)/3
> noise <- rnorm(n, 0, 0.15)
> y <- true + noise
> library(pspline)
> n <- 100
> x <- (1:n)/n
> true <- ((exp(1.2*x)+1.5*sin(7*x))-1)/3
> noise <- rnorm(n, 0, 0.15)
> y <- true + noise
> fit <- smooth.Pspline(x, y, method=3)
> plot(x,y)
> lines(x,fit$ysmth,type='l',col="red")
> fit <- smooth.Pspline(x, y, method=3)
> plot(x,y)
> lines(x,fit$ysmth,type='l',add=TRUE)
> curve(((exp(1.2*x)+1.5*sin(7*x))-1)/3,0,
+       1,add=TRUE,col="red")
```

Restricted Cubic Spline Regression