GSERM 2017
Regression III
Flexible Nonlinear Models

June 20, 2017 (afternoon session)

Nonlinearity is just $\frac{\partial Y}{\partial X} \neq c$.

- Implies that, at some point over the range of values of X, the shape of the relationship between Y and X changes
- How do we deal with this when it happens?

One option:

1. Transform X and/or Y to make the relationship linear
2. Fit a linear model

Another alternative: **fit a nonlinear model** of Y on X.

Common Nonlinear Models and Methods

- Polynomials of X
- Nonlinear Least Squares
- "Kernel-Regularized" Least Squares
- **Spline Functions**
- **Smoothing Splines / Additive Models**
- Tree-Based Methods
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)

A Simple Example: Piecewise Regression

Idea: If $\frac{\partial Y}{\partial X}$ varies across X, then simply fit different regressions for different "regions" defined by values of X.

Define:

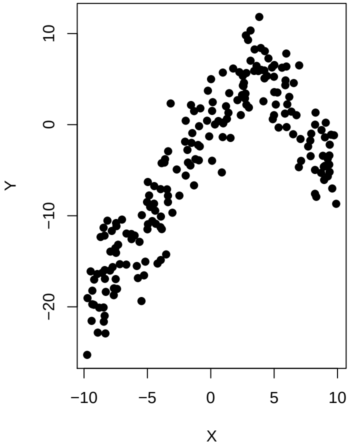
$$X_{Li} = \begin{cases} X \text{ if } X < c \\ 0 \text{ otherwise} \end{cases}$$

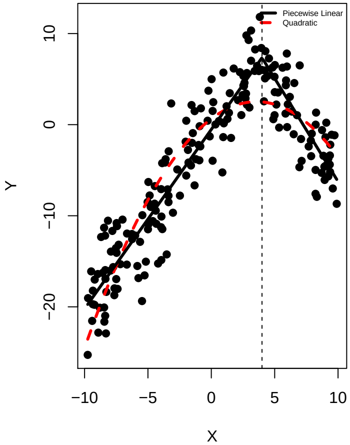
$$X_{Hi} = \begin{cases} X \text{ if } X > c \\ 0 \text{ otherwise} \end{cases}$$

for some chosen value of c. Then fit:

$$Y_i = \beta_0 + \beta_1 X_{Li} + \beta_2 X_{Hi} + u_i$$

A Simple Example: Piecewise Linear Regression





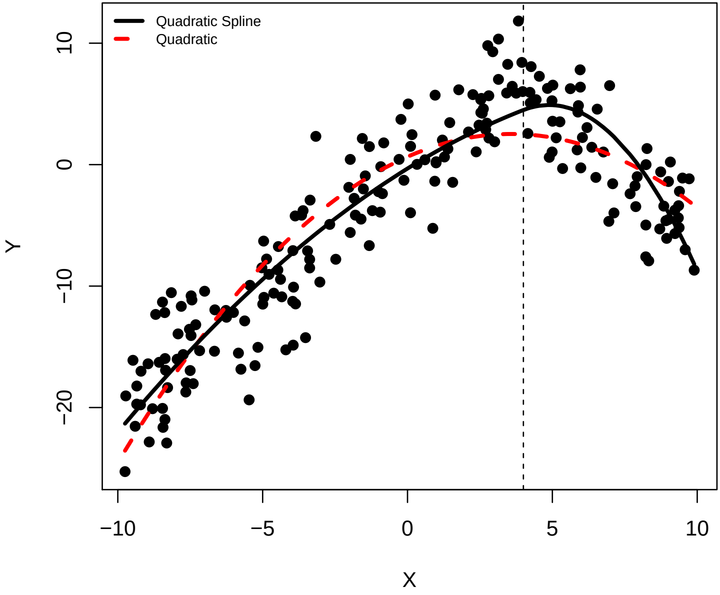
Splines are:

- Piecewise regressions, with
- "Knots" (points where the shape of the function changes) defined by the researcher, and
- some more flexible (nonlinear) functional form for the relationship between X and Y in between each pair of adjacent knots.

So, for a single "knot" (break point) like we had above, a quadratic spline ($p = 2$) would be:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_{Li}^2 + \beta_3 X_{Hi}^2 + u_i$$

Quadratic Spline



More Splines

Probably the most commonly used basis splines are *cubic splines*, which are based on fitting cubic (third-order) polynomials ($p = 3$) between knots:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_{Li}^3 + \beta_4 X_{Hi}^3 + u_i.$$

Note a few things about splines:

- Typically fit using least squares
- Choice of number and location of "knots" is researcher-driven
- Interpretation is via graphical methods (parameter estimates don't have natural interpretations)
- There are other splines too: "natural" splines, B-splines, others (see Keele's 2008 book)
- **Prone to overfitting**

Smoothing Splines

Smoothing splines[1] are a means of avoiding overfitting in spline-based models. Basis (and other) splines are fit via OLS; if we generically call the spline function $f(X)$, that means we are minimizing:

$$SS = \sum_{i=1}^{N}[Y_i - f(X_i)]^2$$

Smoothing splines add a penalty term of the form:

$$SS = \sum_{i=1}^{N}[Y_i - f(X_i)]^2 + \lambda \int_{X_1}^{X_N} [f''(X)]^2 dx$$

where

- f'' denotes the second derivative of the spline function $f()$ and
- λ is the "smoothing parameter." This second term is sometimes called a "roughness penalty."

[1] Hat tip to David Armstrong for this exposition of smoothing splines; his is longer and almost surely better than mine.

Smoothing Splines: λ

The parameter λ controls the degree of "penalty" assigned for overfitting/roughness. $\lambda = 0$ corresponds to no such penalty, while higher values lead to "smoother" functions.

How do we choose λ?

- The "best" λ to use in any given instance depends on the actual relationship between X and Y, something we don't know
- Alternatives:
  - Trial and error
  - Cross-validation

Smoothing Splines: *df*

Of course, one can also control the degree of "smoothness" (overfitting) in smoothing splines by varying the order of the basis of the polynomial used for the spline fits.

In addition, the *degrees of freedom* (*df*) of the smoother is related to λ, and can also be used to control the amount of smoothing.

- *df* is technically something like the number of degrees of freedom (effective parameters) used by the smoother
- Think of *df* as something like the level of complexity of the function that the smoother is willing to tolerate
- Higher values of *df* correspond to more complex forms (that is, greater overfitting)
- See Keele (2008, pp. 64-69) for the mathematical details.

Smoothing Splines in R

There are at least two R packages that will fit smoothing splines:

- the `smoothing.spline` function in the `splines` package
- the `sm.spline` function in the `pspline` package

For the latter (illustrated here):

- `spar` controls smoothing via λ
- `df` controls smoothing via df.

As with splines generally, **interpretation is graphical**.

A Simulated Example

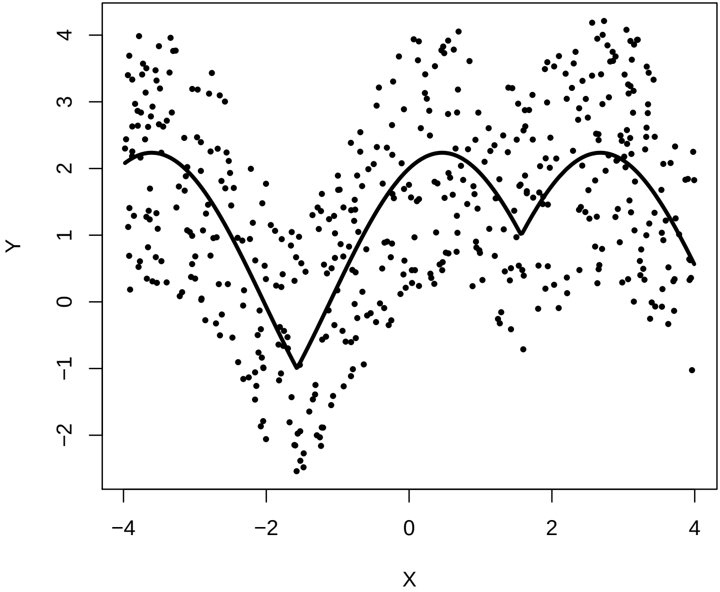
Generate data according to:

$$Y_i = \sin(X_i) + [2 \times |\cos(-X_i)|] + u_i$$

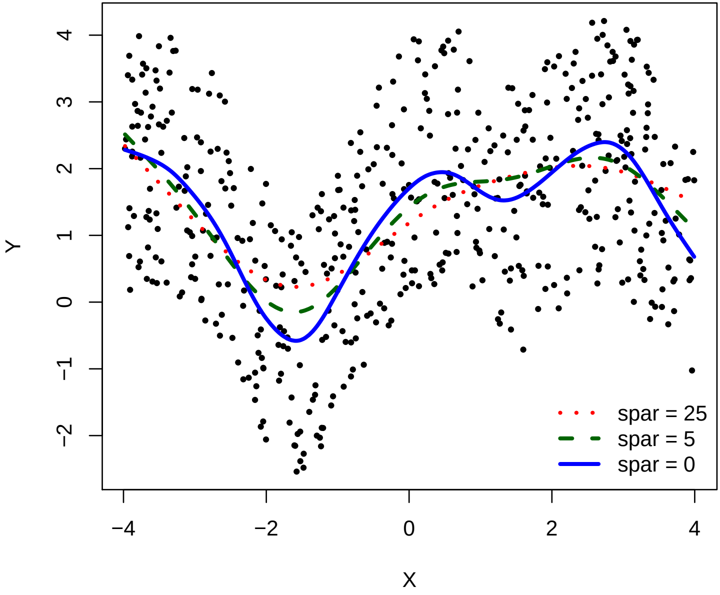
where

$$\begin{aligned} X_i &\in U(-4, 4) \text{ and} \\ u_i &\in U(-1, 1) \text{ and} \\ N &= 500 \end{aligned}$$

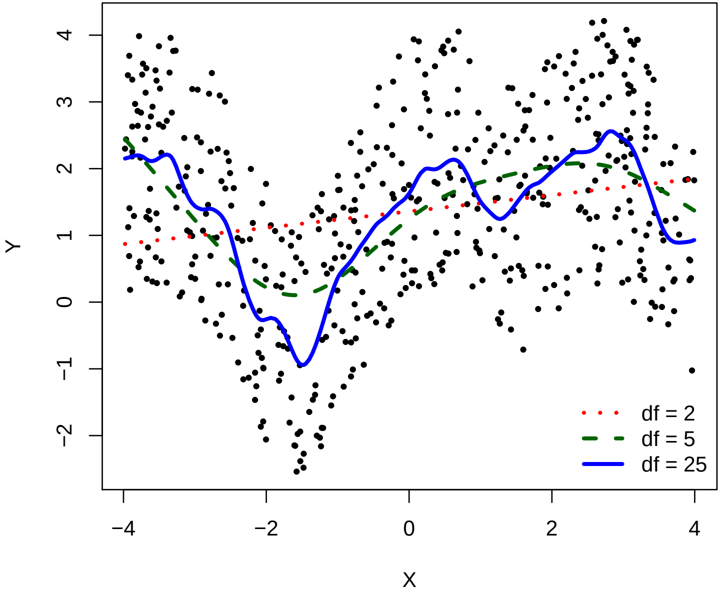
Our Simulated Data



Varying λ (via `spar()`)



Varying λ (via `df()`)



Multivariate Smoothing: Additive Models

Additive models[2] generalize the models we just discussed to the case of more than one predictor. The general form is:

$$Y_i = \beta_0 + f_1(X_{1i}) + f_2(X_{2i}) + ... + f_k(X_{ki}) + u_i$$

- The $f(\cdot)$s are analogous to the βs in linear regression
- As in linear regression, we want each estimate $\widehat{f(\cdot)}$ to be "holding everything else constant"
- If the Xs were independent, we could estimate them separately.
- Since they aren't we need to remove the effects of other predictors (which are unknown) before we begin...

[2] Thanks again to Dave Armstrong and Luke Keele for this exposition.

Intuition: Suppose we had a two-variable model:

$$Y_i = \beta_0 + f_1(X_{1i}) + f_2(X_{2i}) + u_i$$

If we knew $f_2(\cdot)$, but not $f_1(\cdot)$, we could write:

$$Y_i - f_2(X_{2i}) = \beta_0 + f_1(X_{1i}) + u_i$$

and then get $f_1(\cdot)$ via smoothing splines or the like.

Instead, we can iteratively act as if we know $f_1(\cdot)$ and $f_2(\cdot)$:

- Fit $f_1(X_{1i})$ assuming we know $f_2(X_{2i})$
- Generate partial residuals from $\widehat{f_1(X_{1i})}$
- Use the partial residuals to fit a model for $f_2(\cdot)$
- Generate partial residuals from $\widehat{f_1(X_{2i})}$
- Iterate until convergence.

Once again, Keele (2008, Chapter 6) has details.

Additive models can be fitted using the `gam` routine in the `mgcv` package. For an example, we'll add a variable to our earlier simulation:

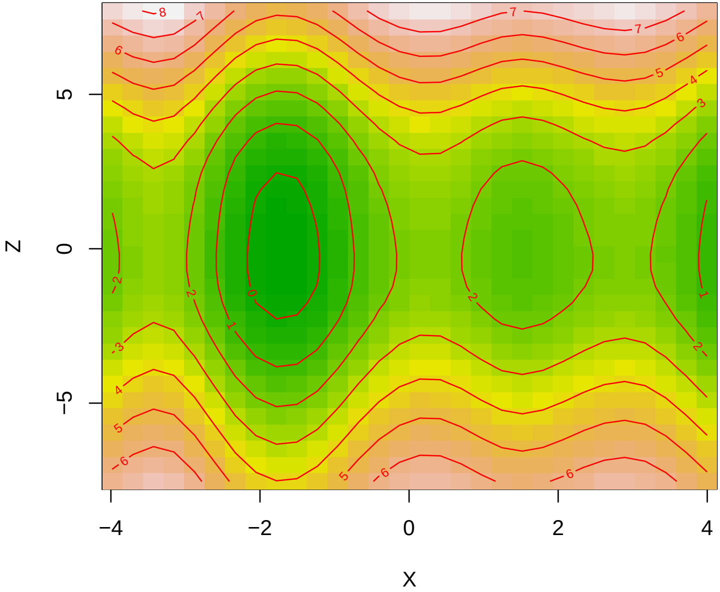
$$Y_i = \sin(X_i) + [2 \times |\cos(-X_i)|] + 0.1 \times Z_i^2 + u_i$$

where X and u are as before, $N = 500$, and $Z_i = -X_i + \epsilon_i$ and $\epsilon \sim U(-4, 4)$.

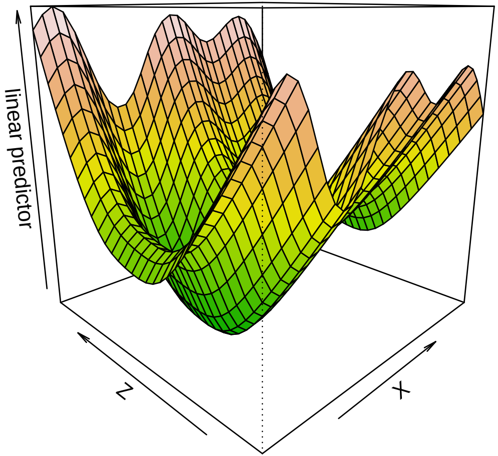
The code is surprisingly simple:

```
simfit <- gam(Y2 ~ s(X,bs="cr")
                 + s(Z,bs="cr"),data=df)
vis.gam(simfit, color="terrain", plot.type="contour",
        main=" ")
```

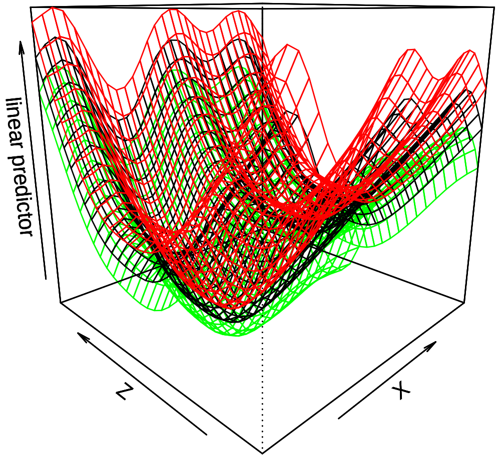
Additive Model: Contour Plot



Additive Model: Perspective Plot



Additive Model: Perspective Plot with 99% c.i.s



Summary

- Nonlinear *models* are different from nonlinear *transformations*...
- This is just a sample....
- Keele's book (and its website) is excellent on this subject
- Challenges:
  - Often require (or perform better with) large amounts of data
  - Requirement that one interpret via graphical means
  - "Atheoretical"... (vs. inductive)