Lecture 8: Nonlinearity (Polynomial Regression, Spline & GAM)

BIOS635

02/04/2020

Some slides taken/modified from other people’s lectures online.

Moving beyond linearity

The truth is never linear!
Or almost never!

But often the linearity assumption is good enough.

When its not . . .

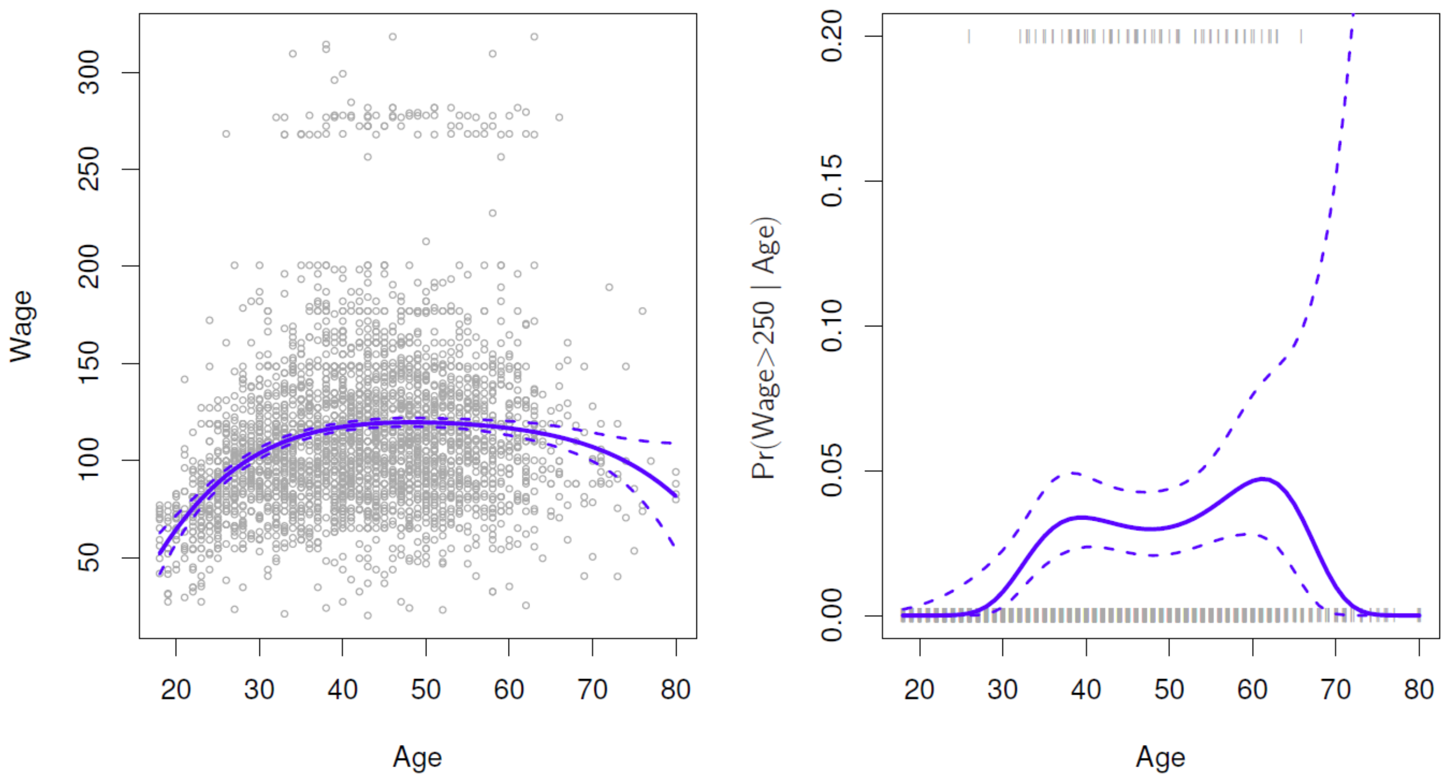
- polynomials,
- step functions,
- splines,
- local regression, and
- generalized additive models

offer a lot of flexibility, without losing the ease and interpretability of linear models.

Polynomial regression

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

Degree−4 Polynomial



Polynomial regression: linear

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

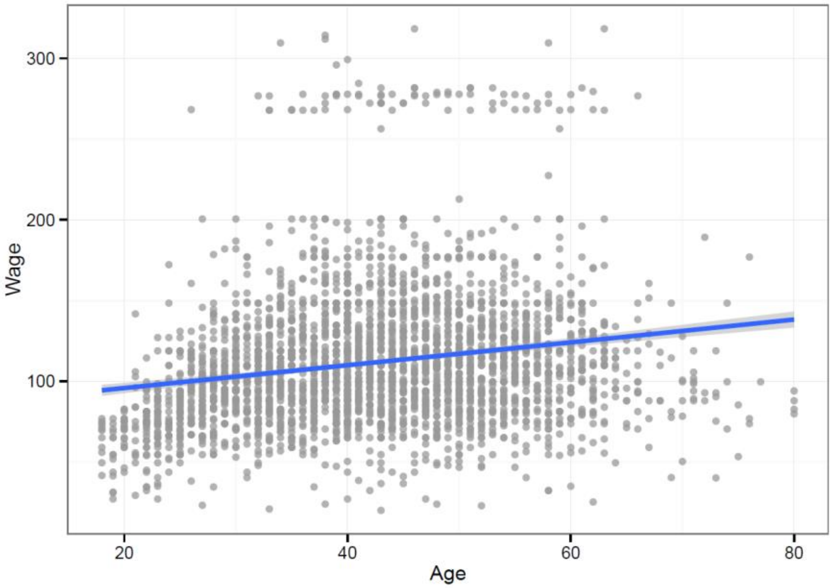
- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_\ell$, can get a simple expression for *pointwise-variances* $\mathrm{Var}[\hat{f}(x_0)]$ at any value x_0. In the figure we have computed the fit and pointwise standard errors on a grid of values for x_0. We show $\hat{f}(x_0) \pm 2 \cdot \mathrm{se}[\hat{f}(x_0)]$.
- We either fix the degree d at some reasonably low value, else use cross-validation to choose d.

Polynomial regression: logistic

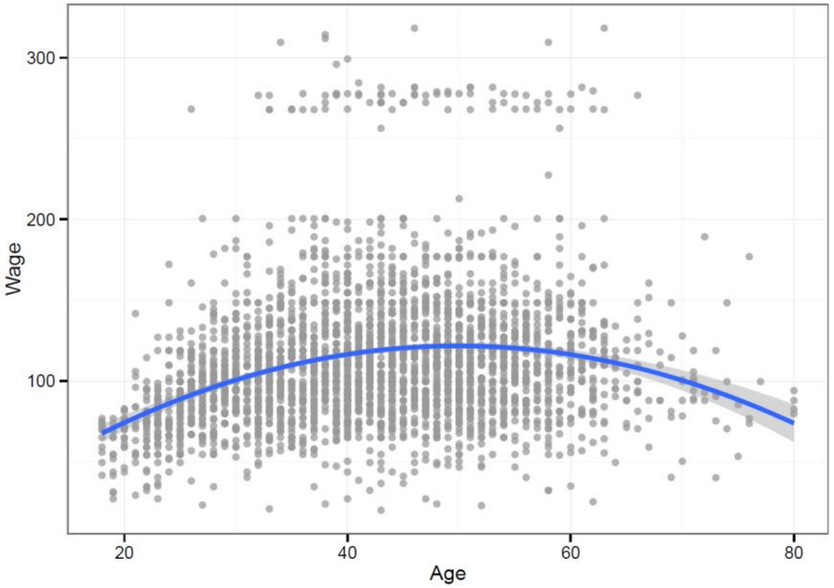
- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

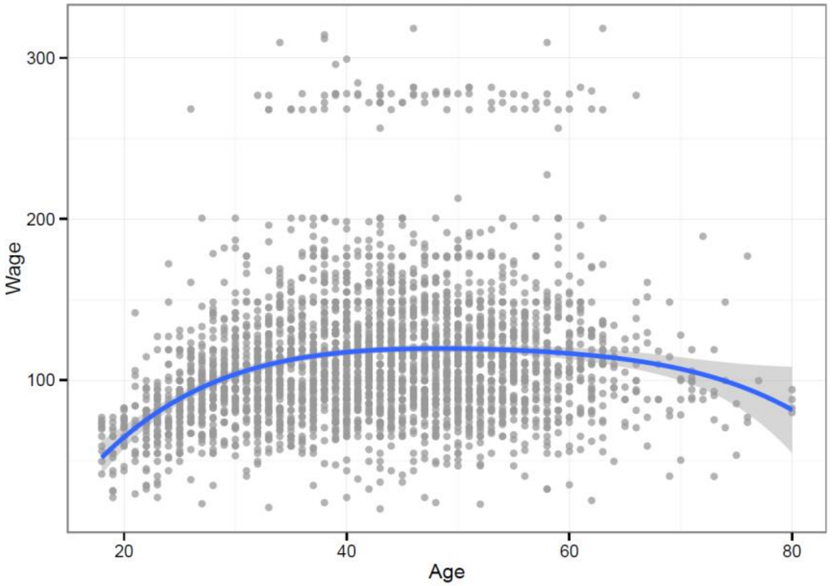
- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).
- Caveat: polynomials have notorious tail behavior — very bad for extrapolation.
- Can fit using $\mathtt{y} \sim \mathtt{poly(x, degree = 3)}$ in formula.



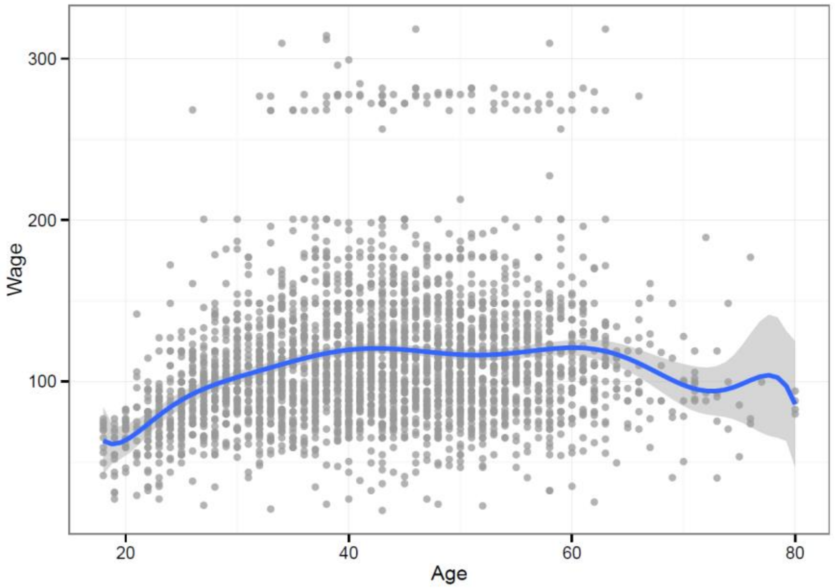
`lm(wage ~ age, data = Wage)`



`lm(wage ~ poly(age, 2), data = Wage)`



`lm(wage ~ poly(age, 4), data = Wage)`



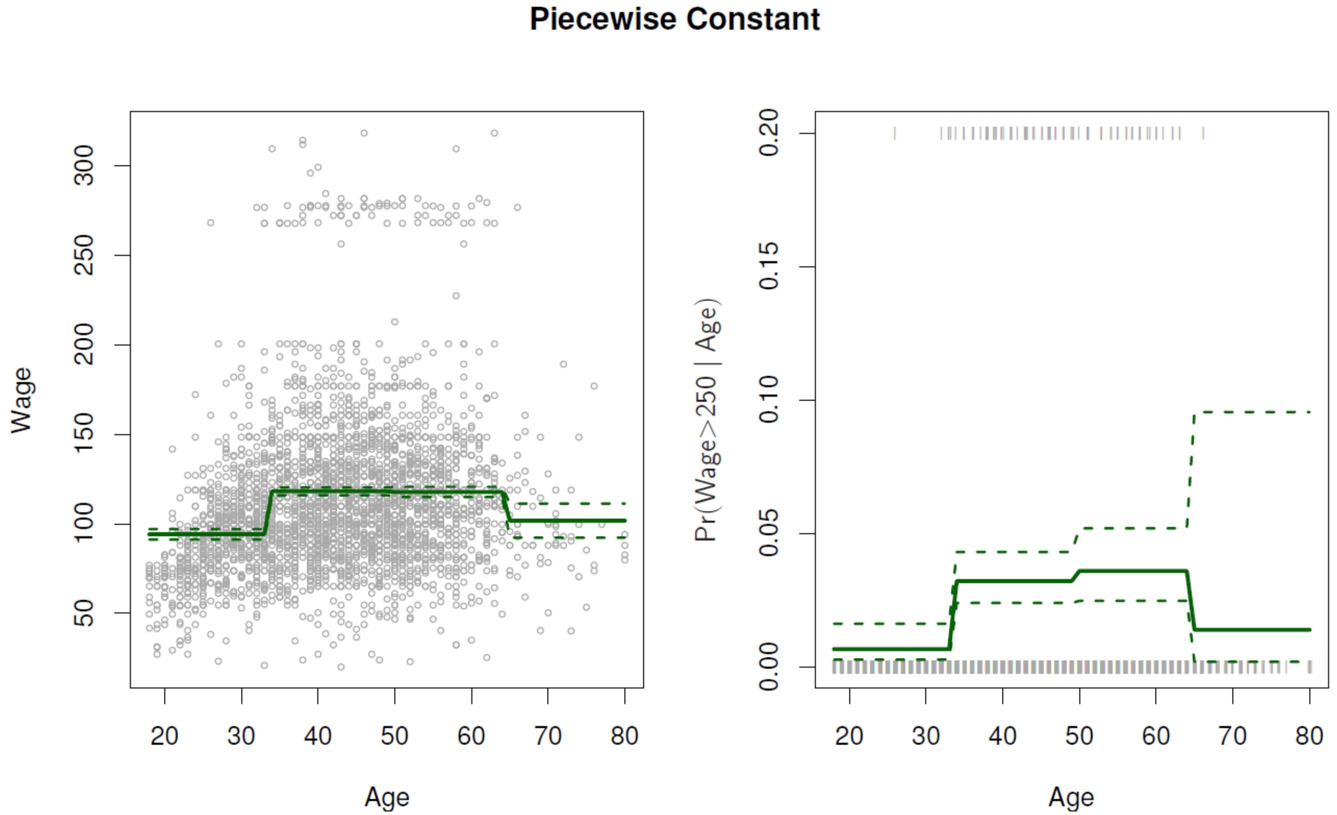
`lm(wage ~ poly(age, 10), data = Wage)`

Step functions

Another way of creating transformations of a variable — cut the variable into distinct regions.

$$C_1(X) = I(X < 35), \quad C_2(X) = I(35 \leq X < 50), \ldots, C_3(X) = I(X \geq 65)$$

Age	C_1	C_2	C_3
18	1	0	0
24	1	0	0
45	0	1	0
67	0	0	1
54	0	1	0
⋮	⋮	⋮	⋮



Step functions, cont'd

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

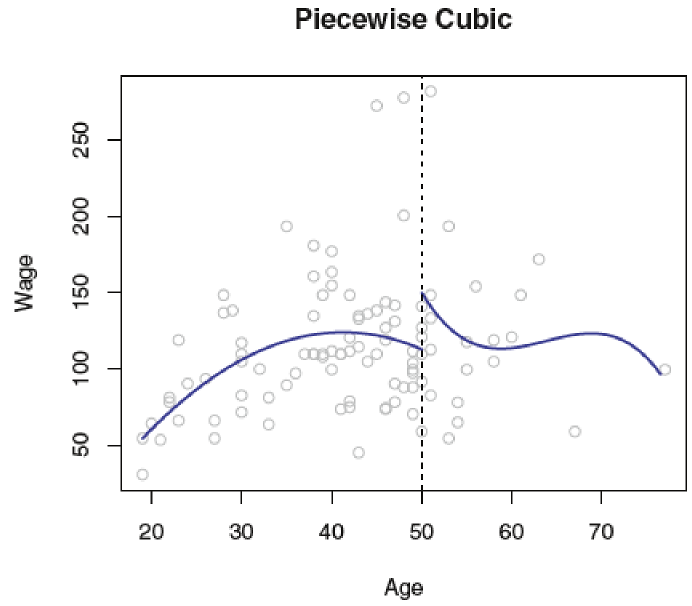
  would allow for different linear functions in each age category.
- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.
- Choice of cutpoints or *knots* can be problematic. For creating nonlinearities, smoother alternatives such as *splines* are available.

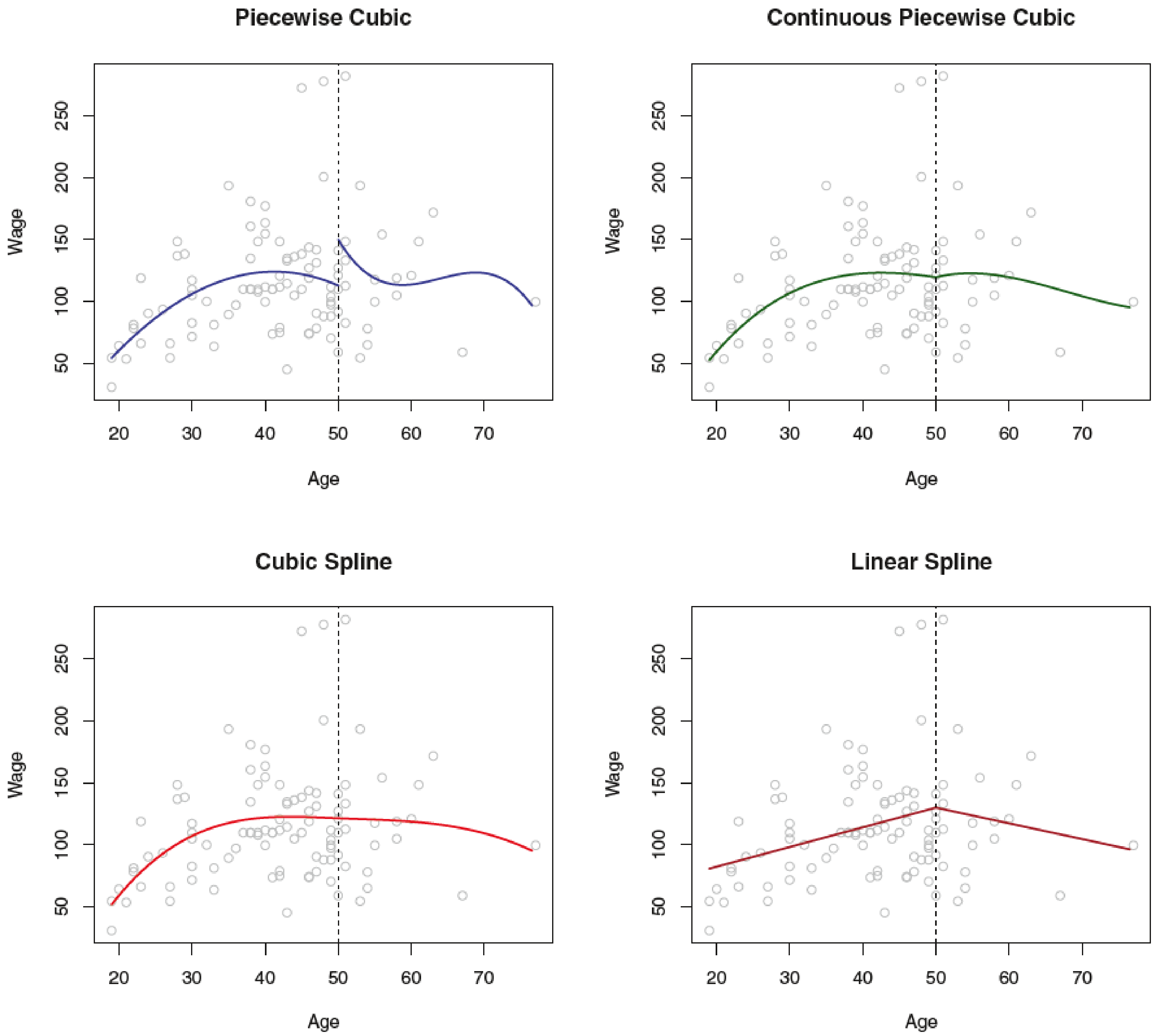
Piecewise polynomials

- Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by knots. E.g. (see figure)

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

- Better to add constraints to the polynomials, e.g. continuity.
- *Splines* have the "maximum" amount of continuity.




Piecewise Cubic
Continuous Piecewise Cubic
Cubic Spline
Linear Spline
Wage
Age
50
100
150
200
250
20
30
40
50
60
70

Linear splines

A linear spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise linear polynomial continuous at each knot.

We can represent this model as

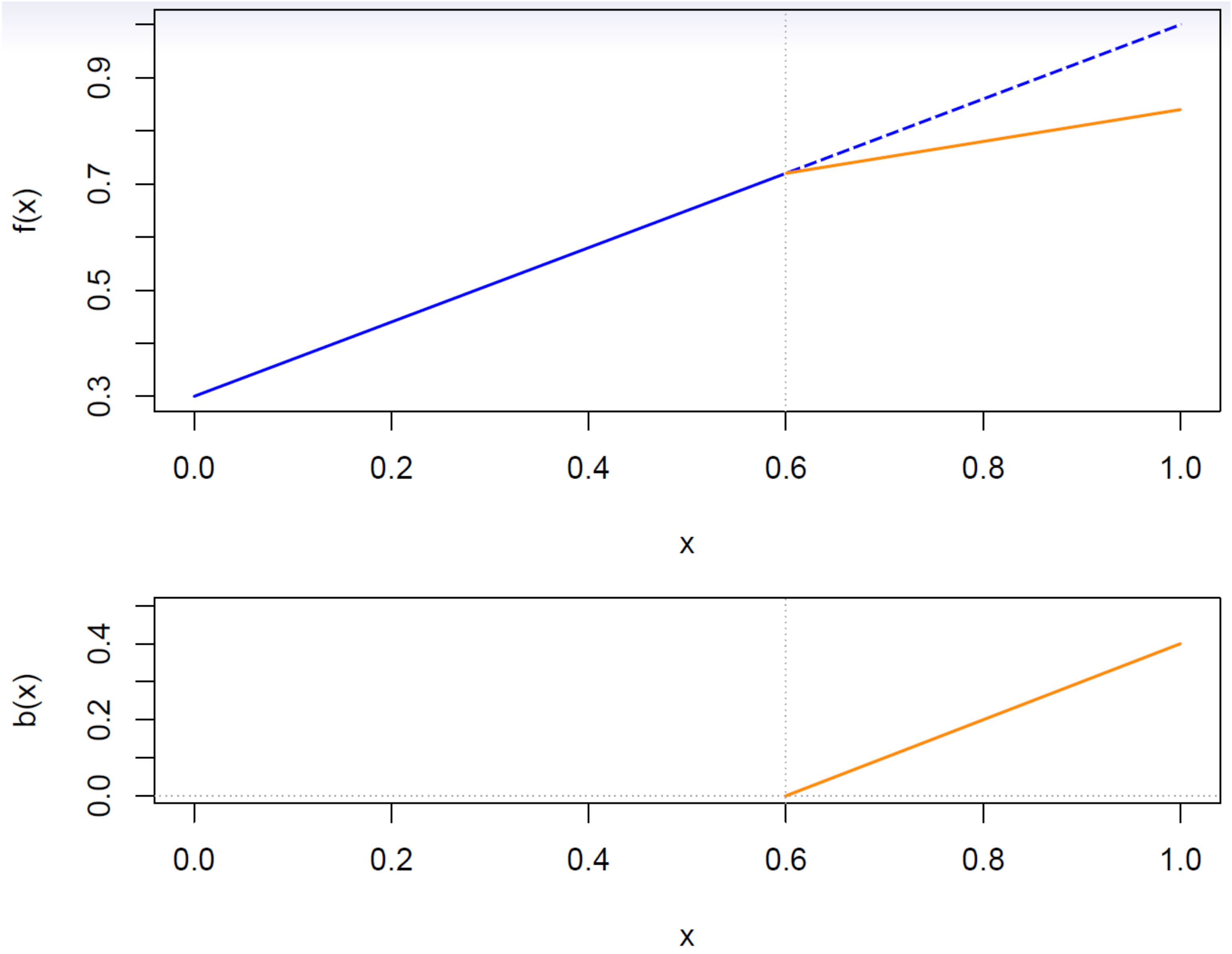
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

where the b_k are *basis functions*.

$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \ldots, K \end{aligned}$$

Here the $()_+$ means *positive part*; i.e.

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$



f(x)
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
x
b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

Cubic splines

A cubic spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.

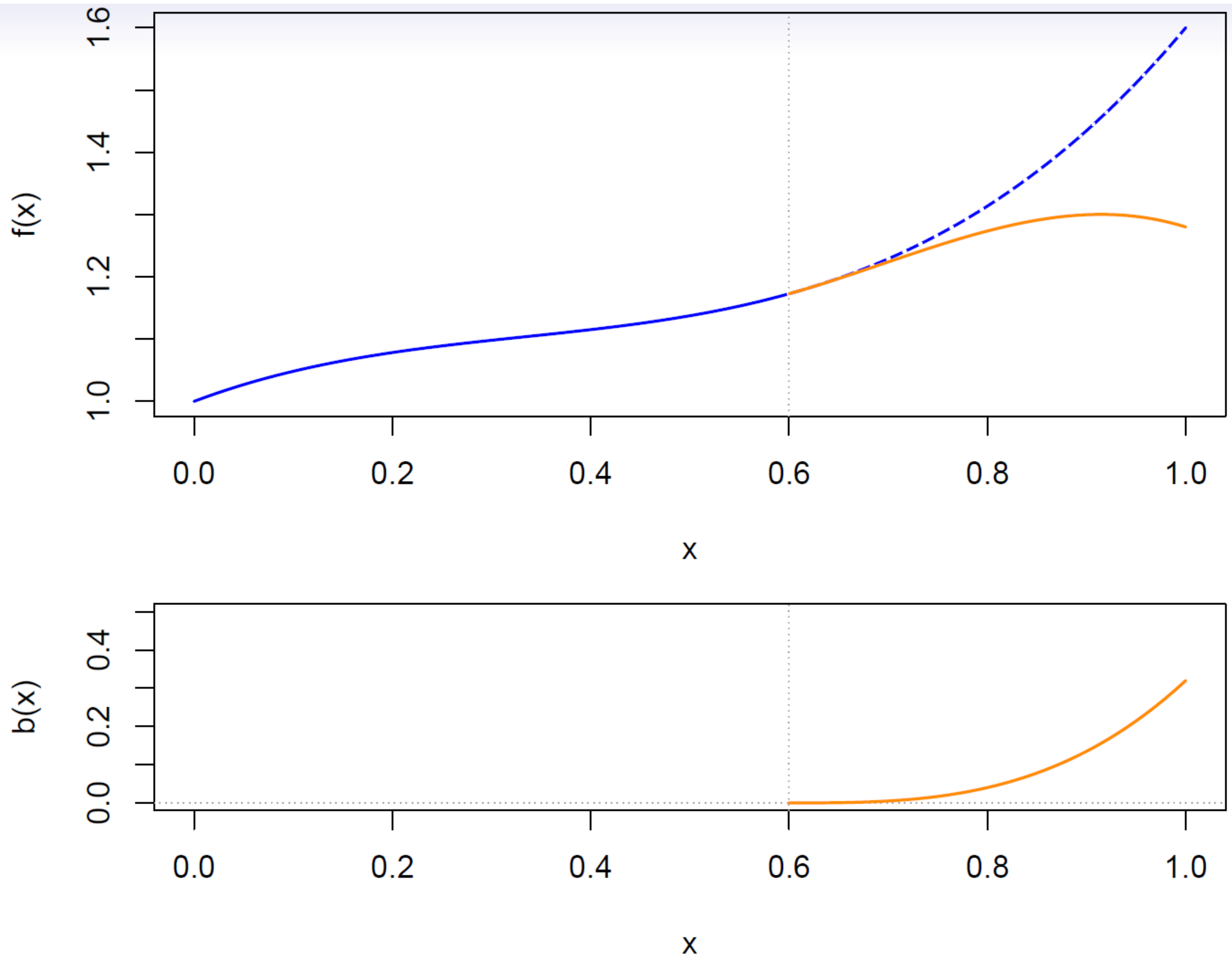
Again we can represent this model with truncated power basis functions

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \ldots, K \end{aligned}$$

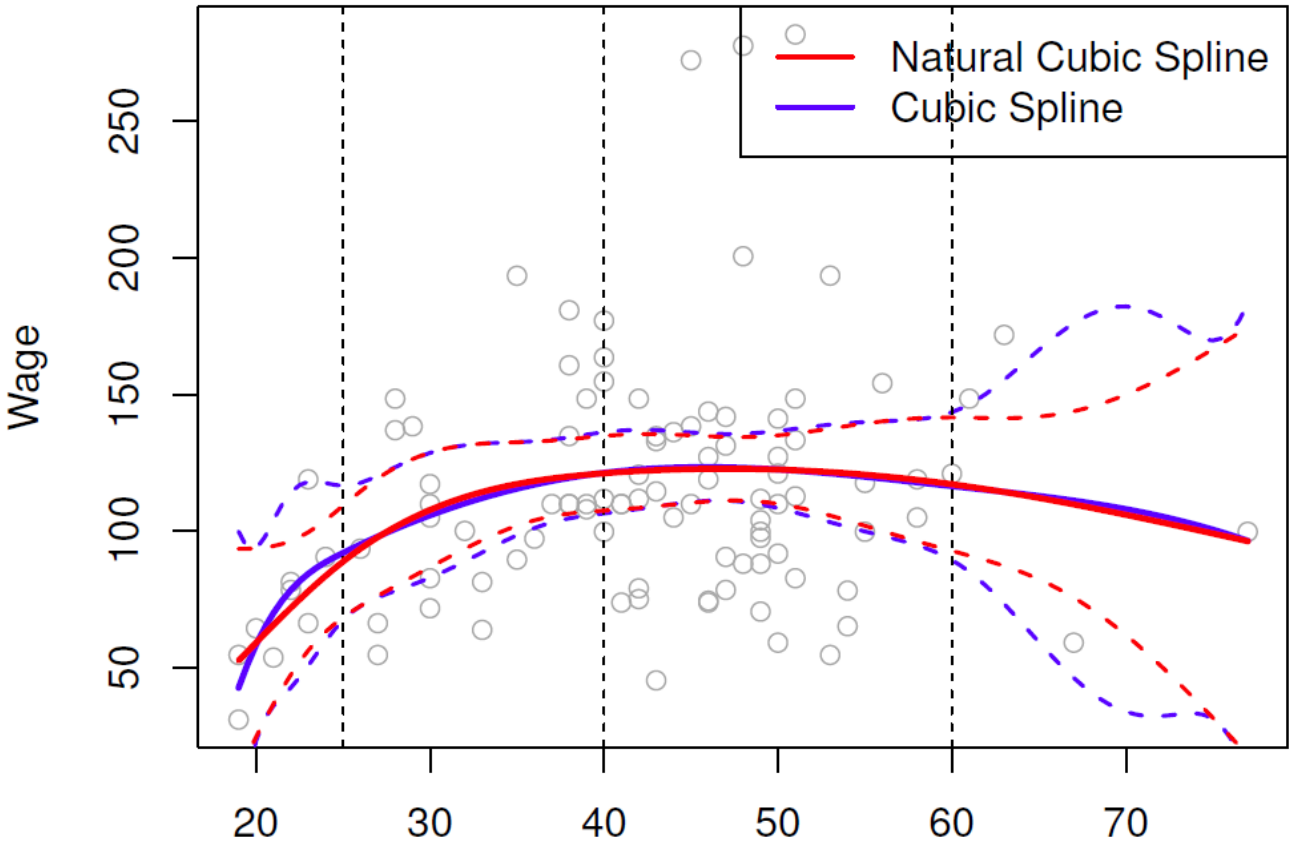
where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$


f(x)
1.0
1.2
1.4
1.6
0.0
0.2
0.4
0.6
0.8
1.0
x
b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

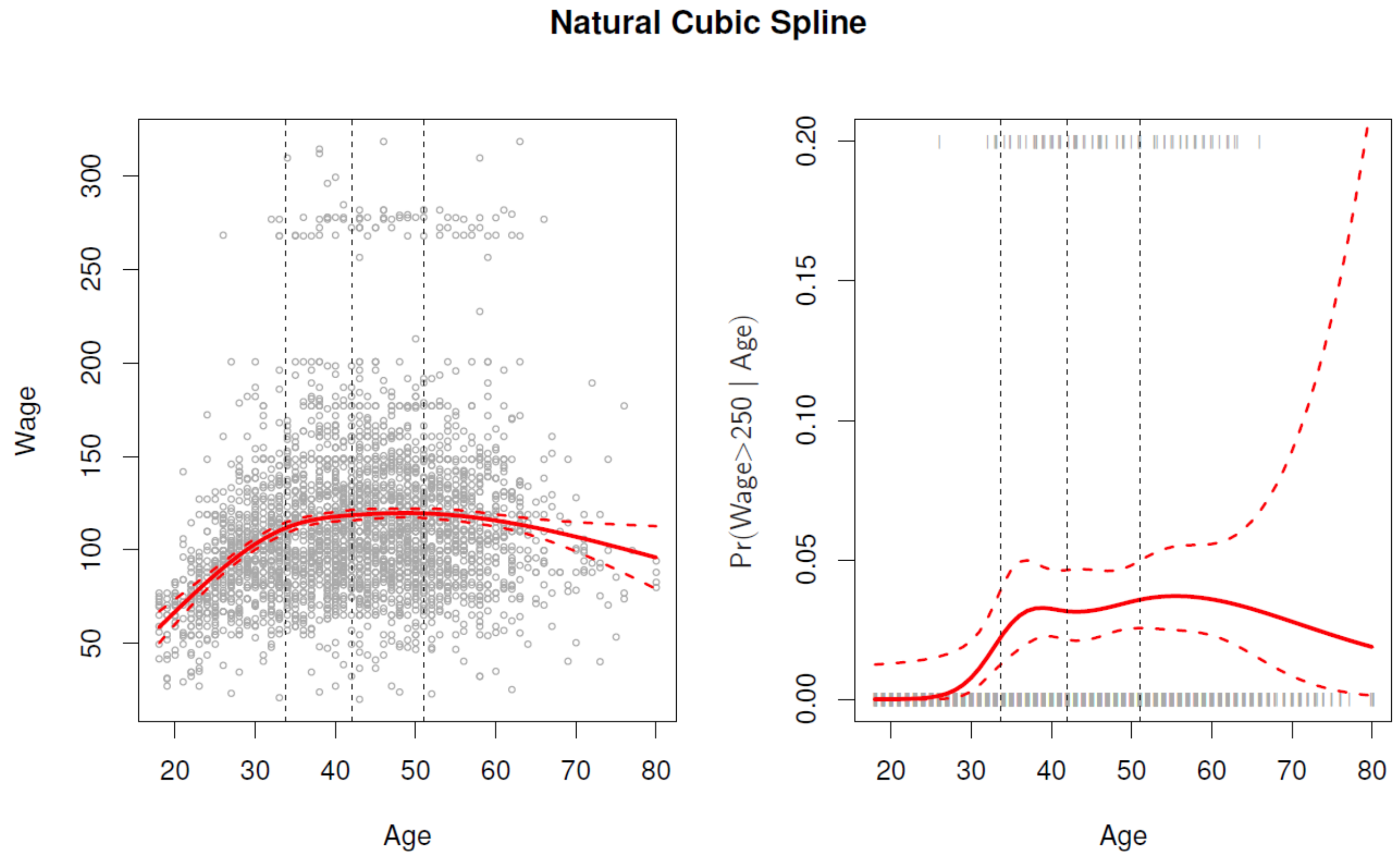
Natural cubic splines

Second derivatives of the spline polynomials are set equal to zero at the end points of interpolation. This forces the spline to be a straight line outside the interval.



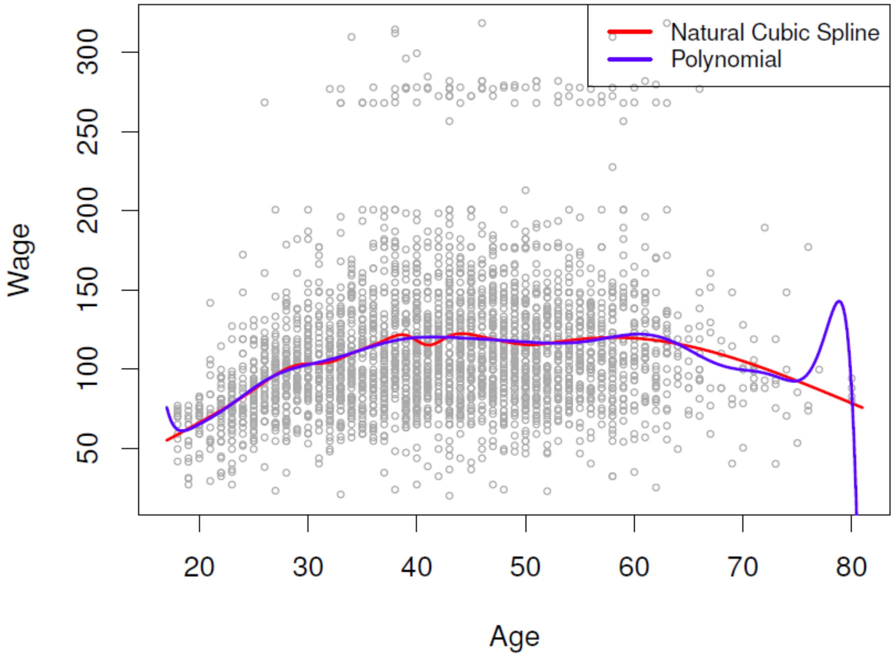
Fitting splines in R

Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package `splines`.



Knot placement & degree of freedom

- One strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X.
- A cubic spline with K knots has $K+4$ parameters or degrees of freedom.
- A natural spline with K knots has K degrees of freedom.



Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

```
ns(age, df=14)
poly(age, deg=14)
```

Smoothing splines

- The smoothing spline estimator is the solution $\hat{g}$ to the problem

$$\text{minimize} \quad \underbrace{\sum_{i=1}^{n} (y_i - g(x_i))^2}_{\text{RSS}} + \lambda \underbrace{\int g''(t)^2 dt}_{\text{Roughness penalty}}$$

- This is a penalized regression problem
- We're saying we want a function that:
  1. Fits the data well; and
  2. isn't too *wiggly*
- Large $\lambda \implies \hat{g}$ will have low variability (& higher bias)
- Small $\lambda \implies \hat{g}$ will have high variability (& lower bias)

How is this *at all* related to splines?

Smoothing splines, cont'd

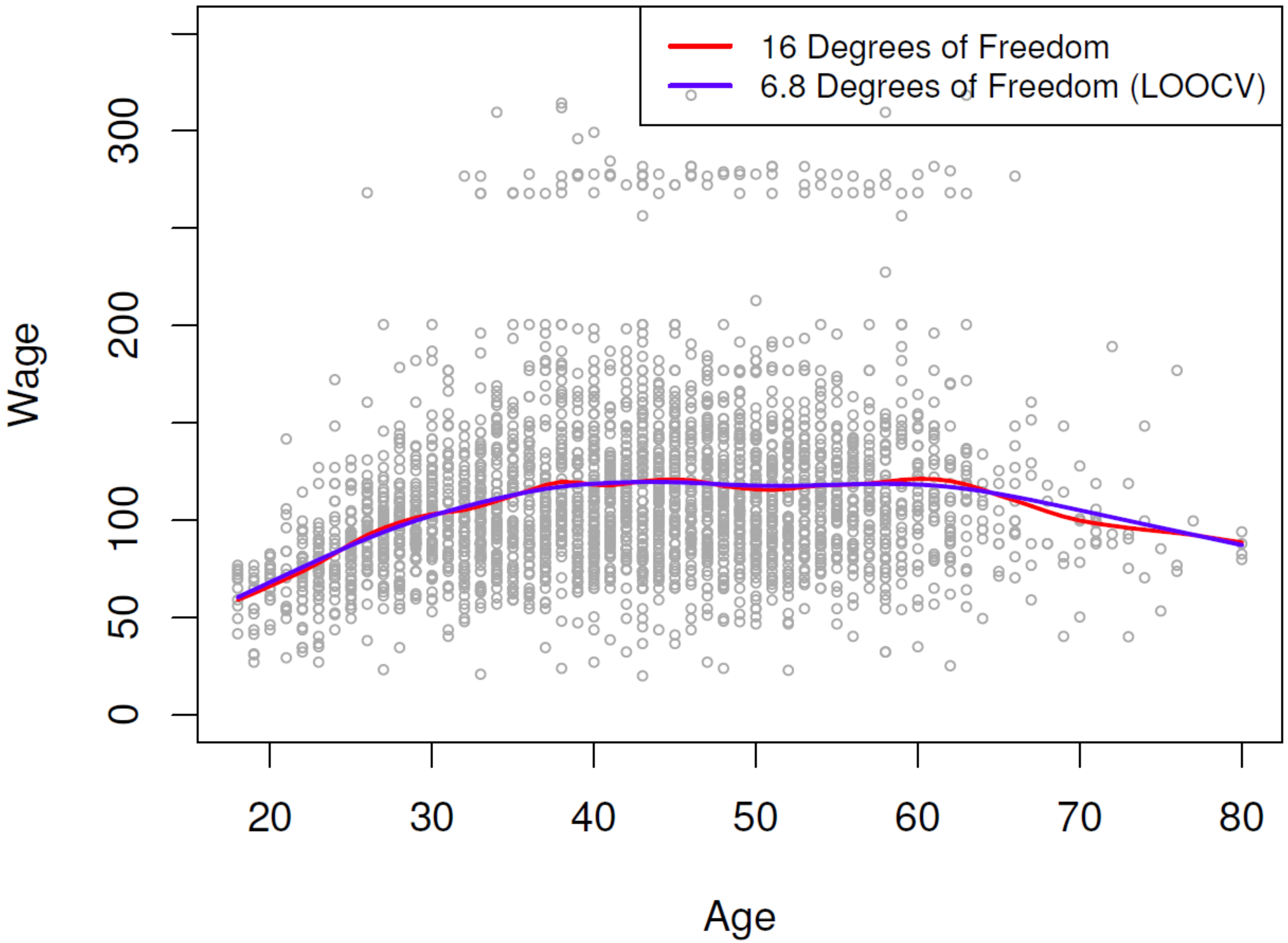
$$\text{minimize} \quad \underbrace{\sum_{i=1}^{n}(y_i - g(x_i))^2}_{\text{RSS}} + \underbrace{\lambda \int g''(t)^2 dt}_{\text{Roughness penalty}} \qquad (*)$$

It turns out…

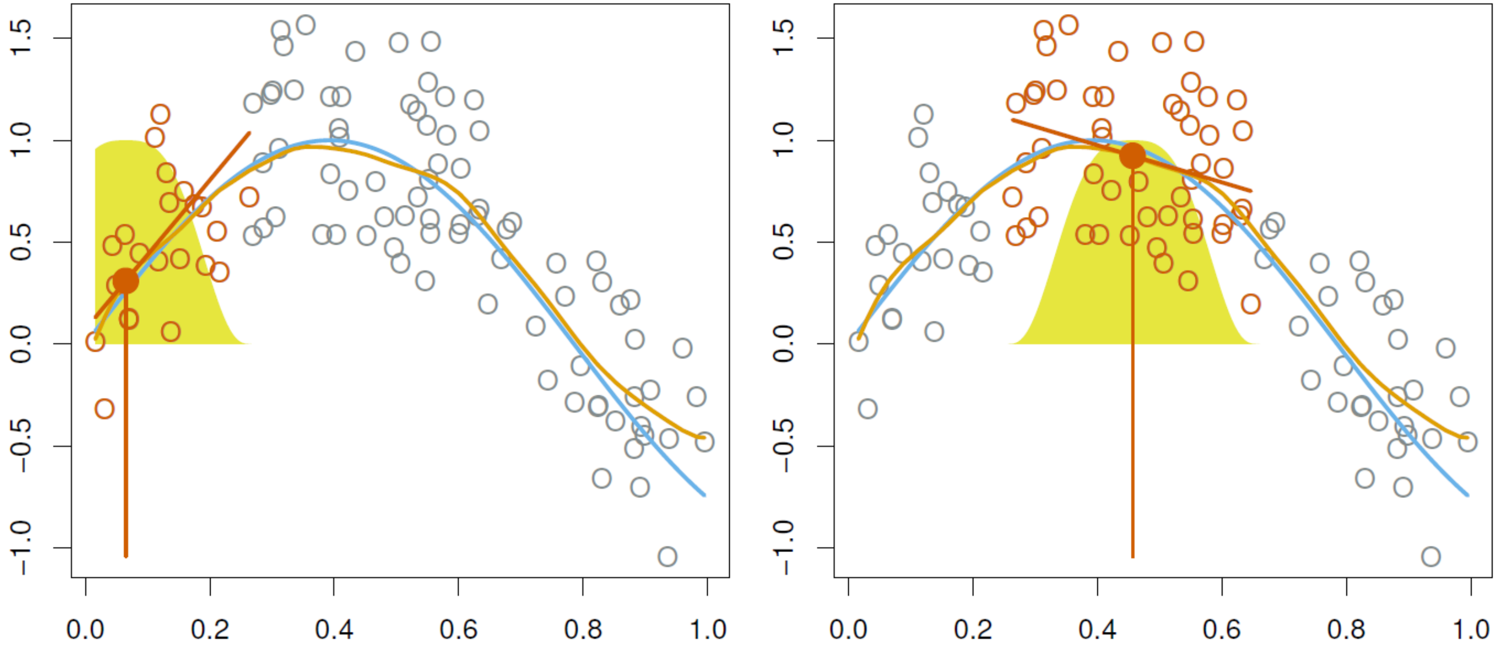
- The solution to $(*)$ is a natural cubic spline
- The solution has knots at every unique value of x
- The effective degrees of freedom of the solution is calculable
- $\lambda \longleftrightarrow df$

Specify df in R: smooth.spline(age, wage, df = 10)

Choose lambda via leave-one-out cross validation (LOOCV) in R: smooth.spline(age, wage)



Local regression



With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares.
See text for more details, and `loess()` function in R.

Nonlinearity coding in R

Model	R command
Degree-d polynomial regression	~ poly(x, degree = d)
Step functions with knots c1, c2, c3	~ cut(x, breaks = c(c1, c2, c3))
Cubic spline	~ bs(x, df)
Natural cubic spline	~ ns(x, df)
Degree-d spline	~ bs(x, df, degree = d)
Smoothing spline	~ s(x, df)
Local linear regression	~ lo(x)

Generalized additive models (GAMs)

- Recall the Linear Regression Model

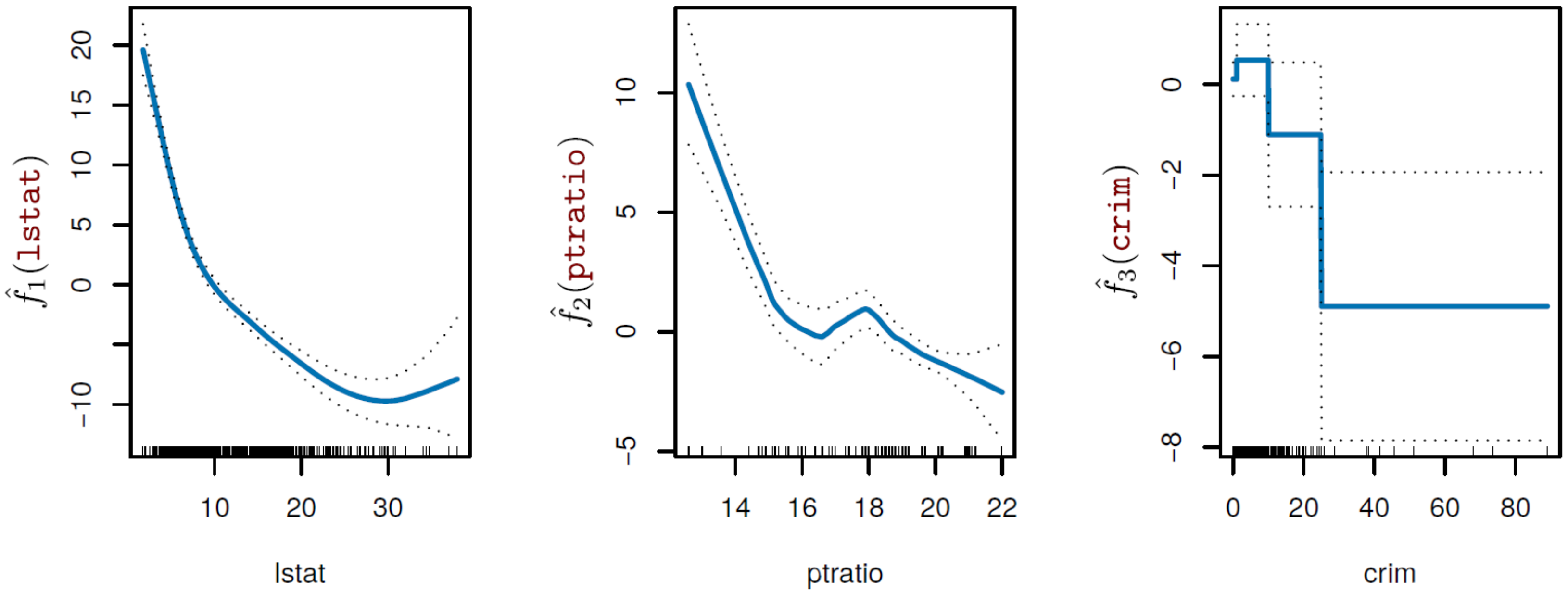
$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \epsilon$$

- We can now extend this to the far more flexible Additive Model

$$Y = \beta_0 + \sum_{j=1}^{p} f_j(X_j) + \epsilon$$

- Each f_j can be any of the different methods we just talked about: Linear term $(\beta_j X_j)$, Polynomial, Step Function, Piecewise Polynomial, Degree-k spline, Natural cubic spline, Smoothing spline, Local linear regression fit, …
- You can mix-and-match different kinds of terms
- The `gam` and `mgcv` packages enable Additive Models in **R**

GAMs: Boston housing data



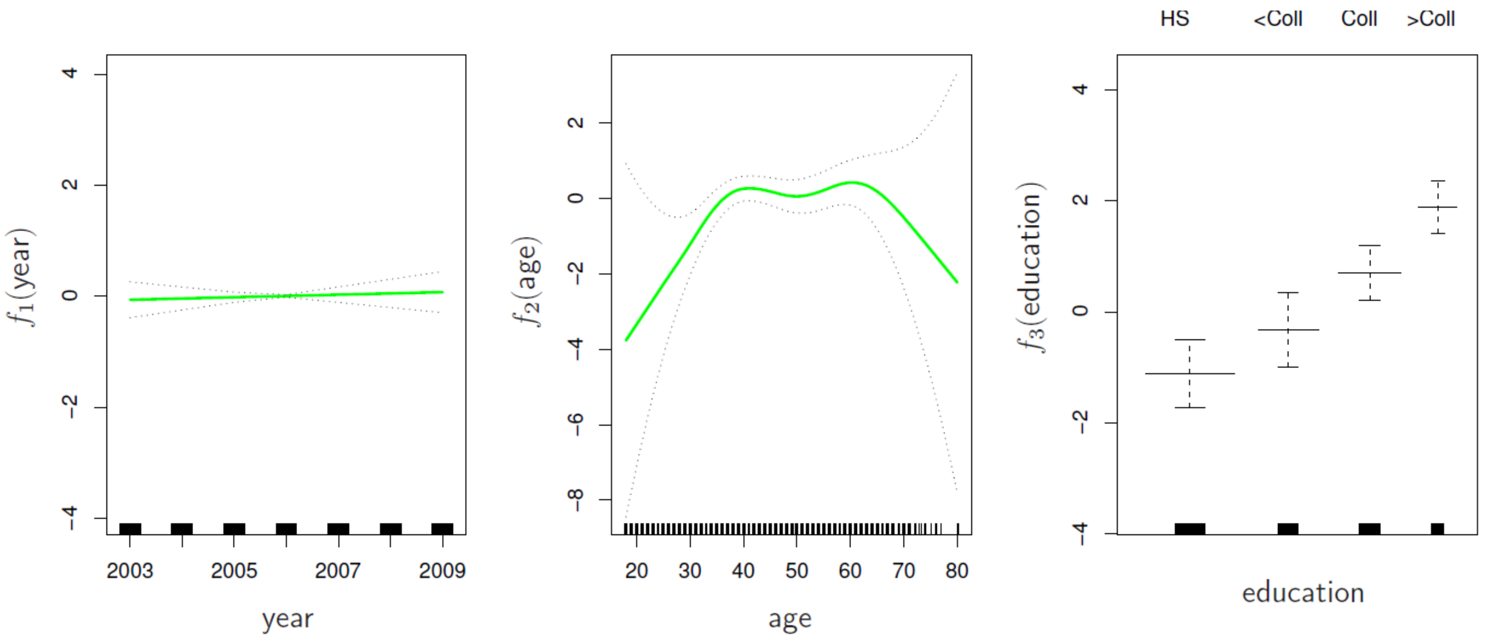
$$\texttt{medv} = f_1(\texttt{lstat}) + f_2(\texttt{ptratio}) + f_3(\texttt{crim}) + \epsilon$$

- $f_1(\texttt{lstat})$ smoothing spline with 5 df
- $f_2(\texttt{ptratio})$ local linear regression
- $f_3(\texttt{crim})$ step function with breaks at $\texttt{crim} = 1, 10, 25$

```
gam(medv ~ s(lstat, 5) + lo(ptratio) +
           cut(crim, breaks = c(-Inf, 1, 10, 25, Inf)),
           data = Boston)
```

GAMs for classification

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p).$$



$$\texttt{gam(I(wage} > \texttt{250)} \sim \texttt{year} + \texttt{s(age, df} = \texttt{5)} + \texttt{education, family} = \texttt{binomial)}$$