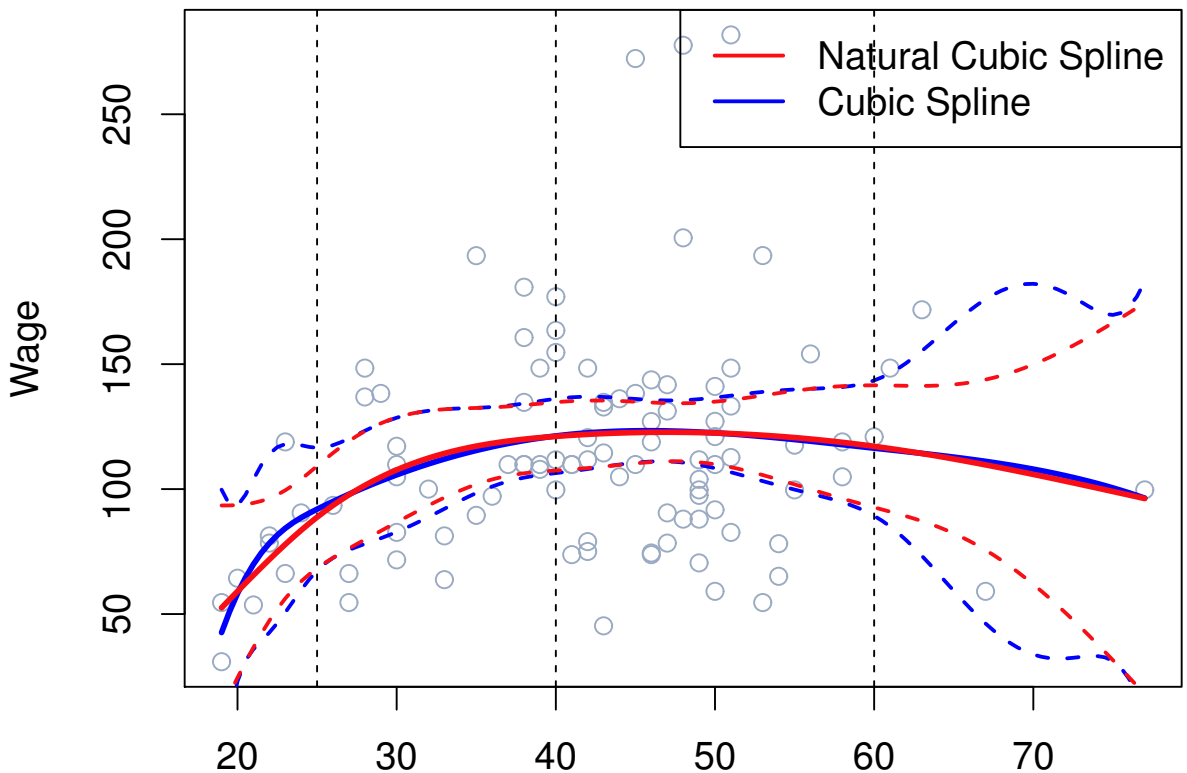
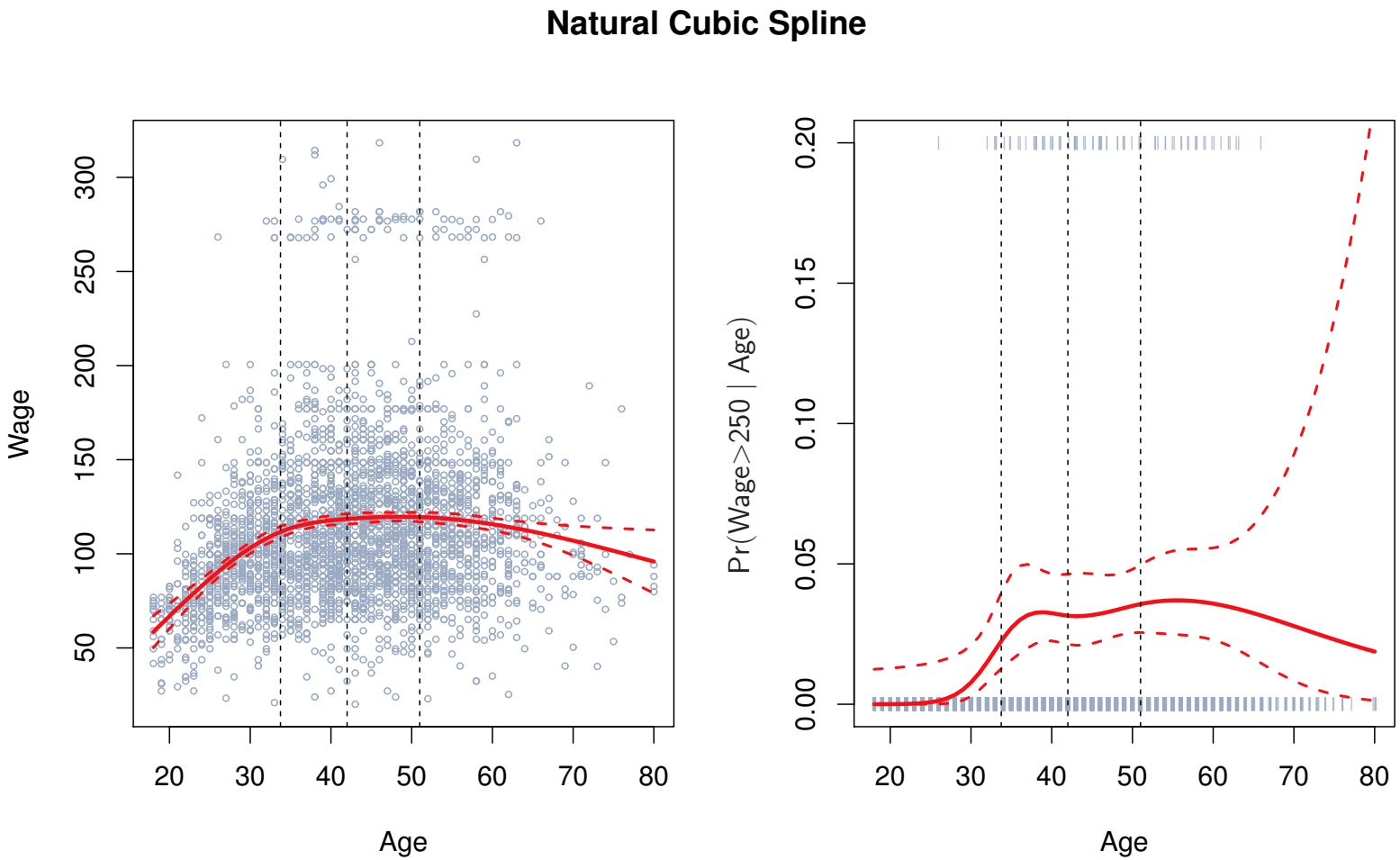
Natural Cubic Splines

A natural cubic spline extrapolates linearly beyond the boundary knots. This adds $4 = 2 \times 2$ extra constraints, and allows us to put more internal knots for the same degrees of freedom as a regular cubic spline.

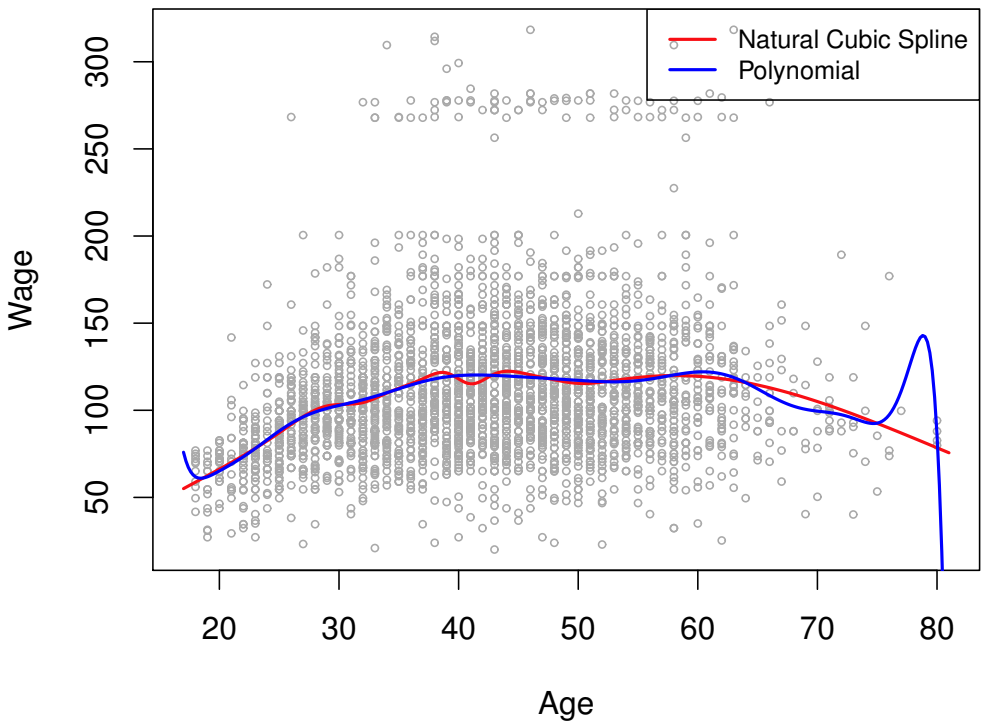


Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package `splines`.



Knot placement

- 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

This section is a little bit mathematical 

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.
  - The smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$.
  - As $\lambda \to \infty$, the function $g(x)$ becomes linear.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.
- The vector of n fitted values can be written as $\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y}$, where $\mathbf{S}_\lambda$ is a $n \times n$ matrix (determined by the x_i and λ).
- The *effective degrees of freedom* are given by

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}.$$

Smoothing Splines continued — choosing λ

- We can specify *df* rather than λ!

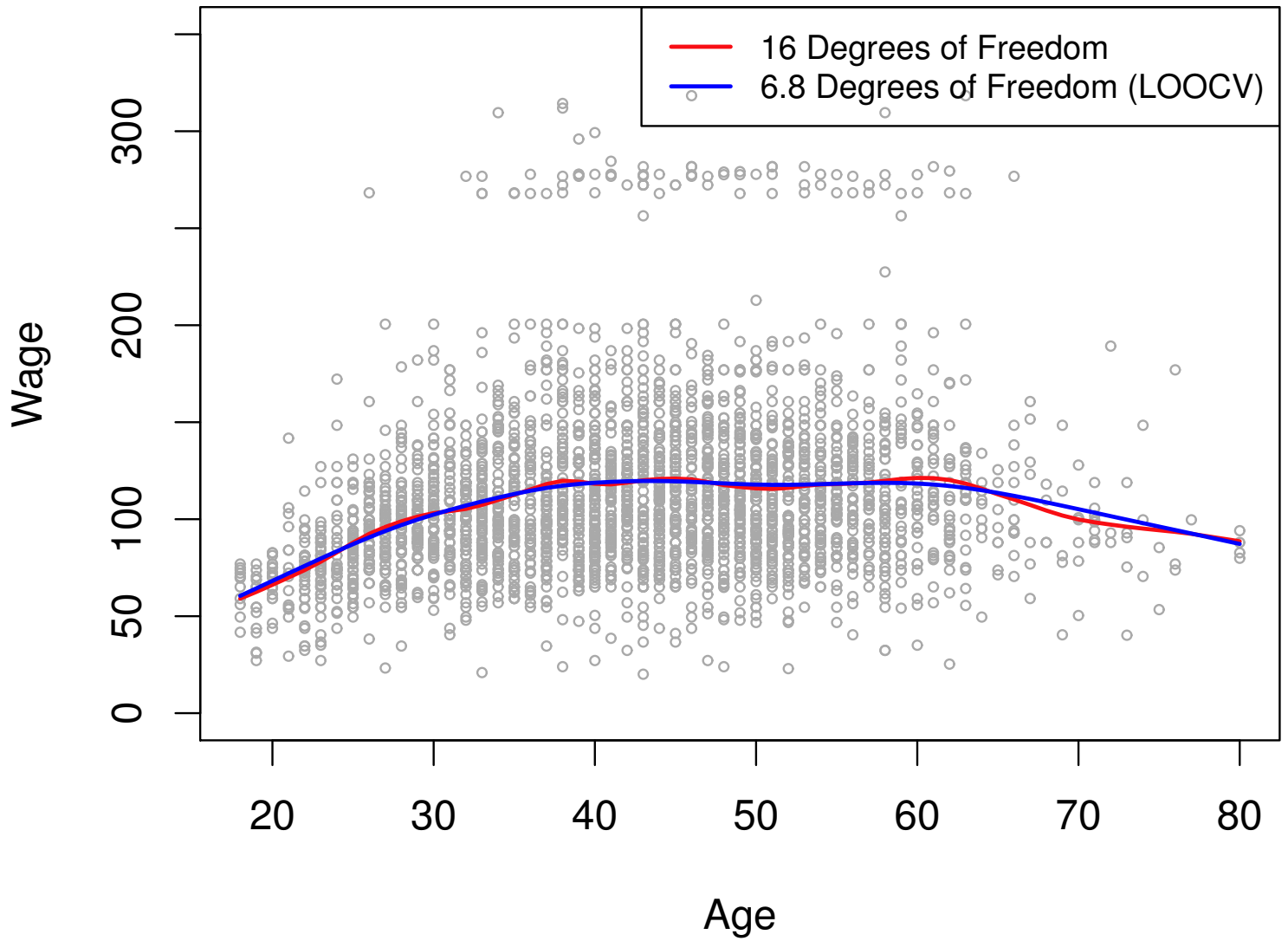
  In R: `smooth.spline(age, wage, df = 10)`

- The leave-one-out (LOO) cross-validated error is given by

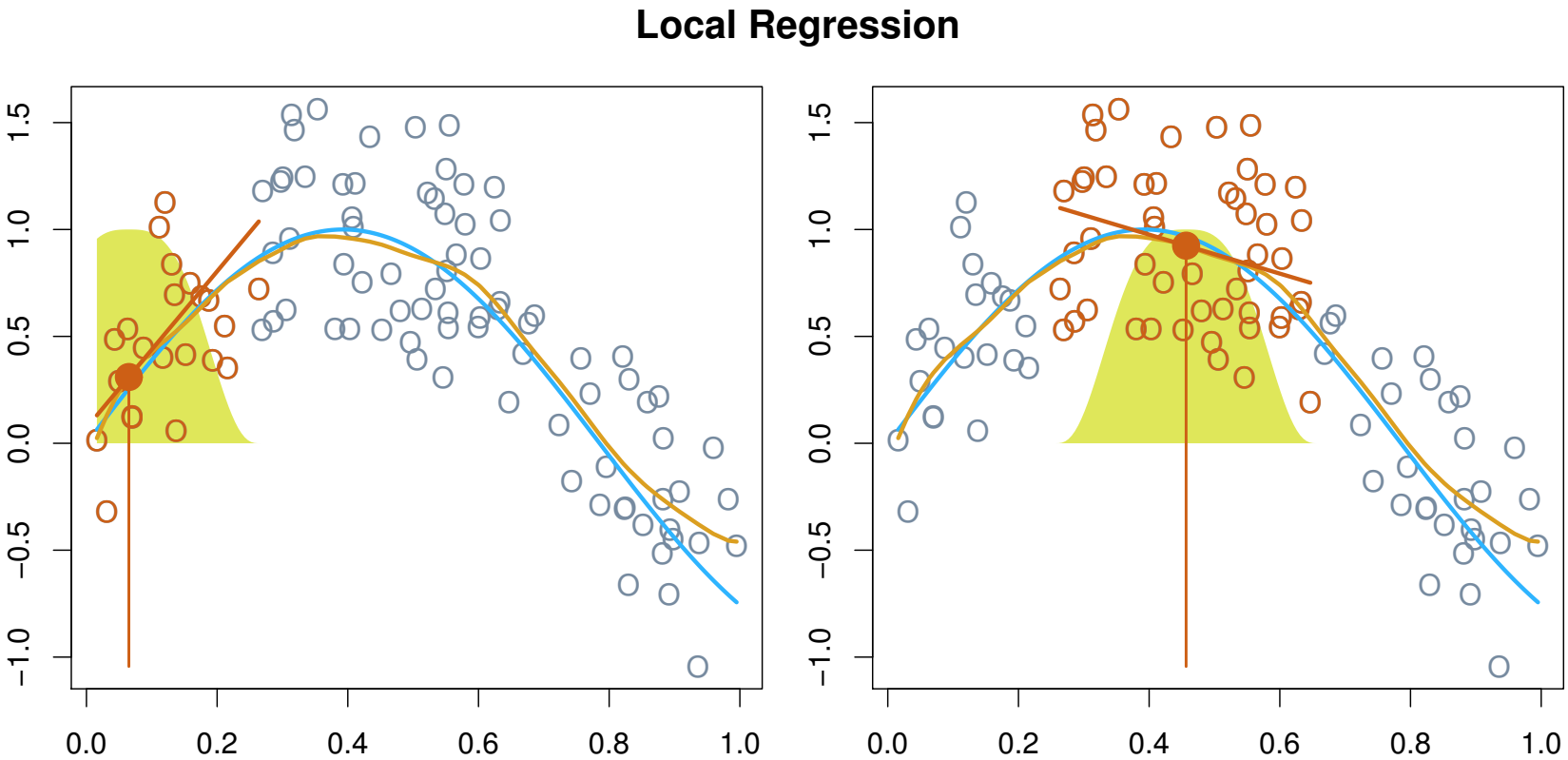
$$\text{RSS}_{cv}(\lambda) = \sum_{i=1}^{n}(y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n}\left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}}\right]^2.$$

  In R: `smooth.spline(age, wage)`

Smoothing Spline



Local Regression

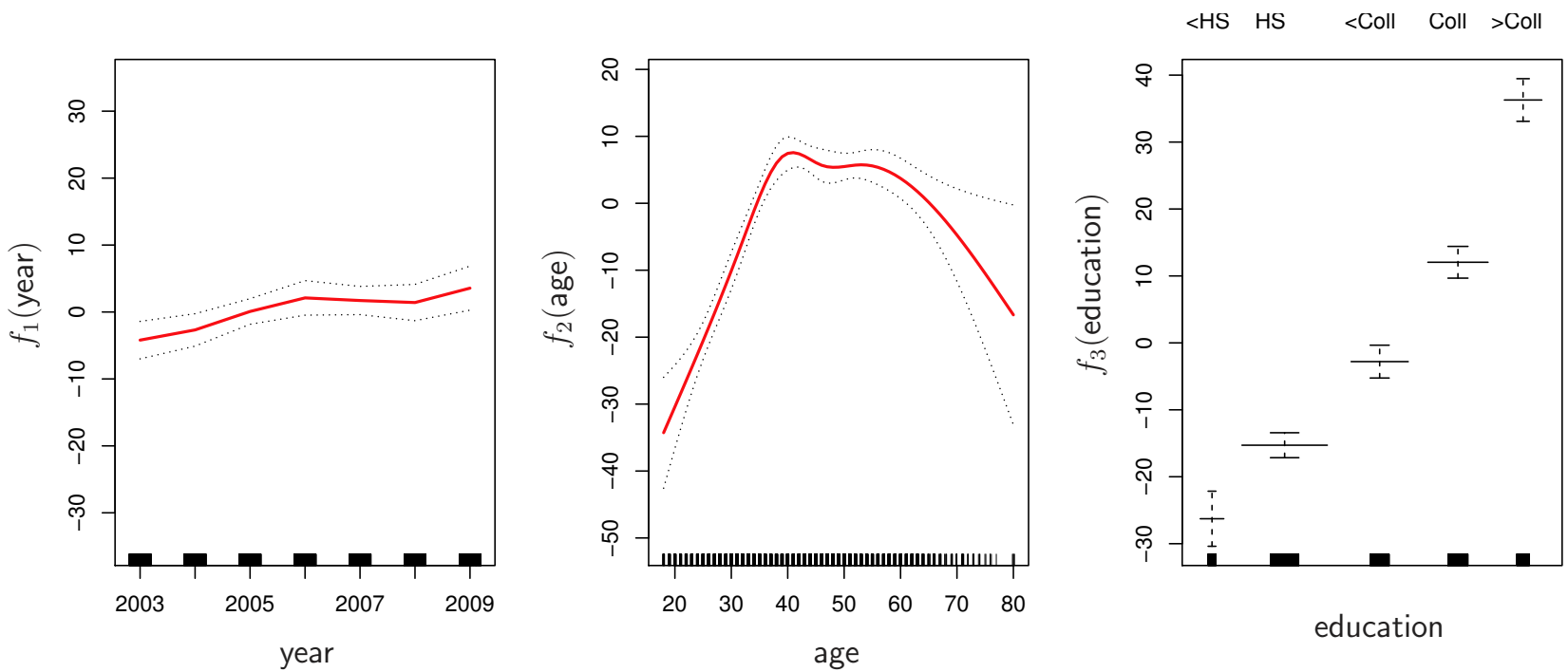


With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares.
See text for more details, and `loess()` function in R.

Generalized Additive Models

Allows for flexible nonlinearities in several variables, but retains the additive structure of linear models.

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i.$$



GAM details

- Can fit a GAM simply using, e.g. natural splines:

  $$\texttt{lm}(\texttt{wage} \sim \texttt{ns}(\texttt{year}, \texttt{df} = 5) + \texttt{ns}(\texttt{age}, \texttt{df} = 5) + \texttt{education})$$

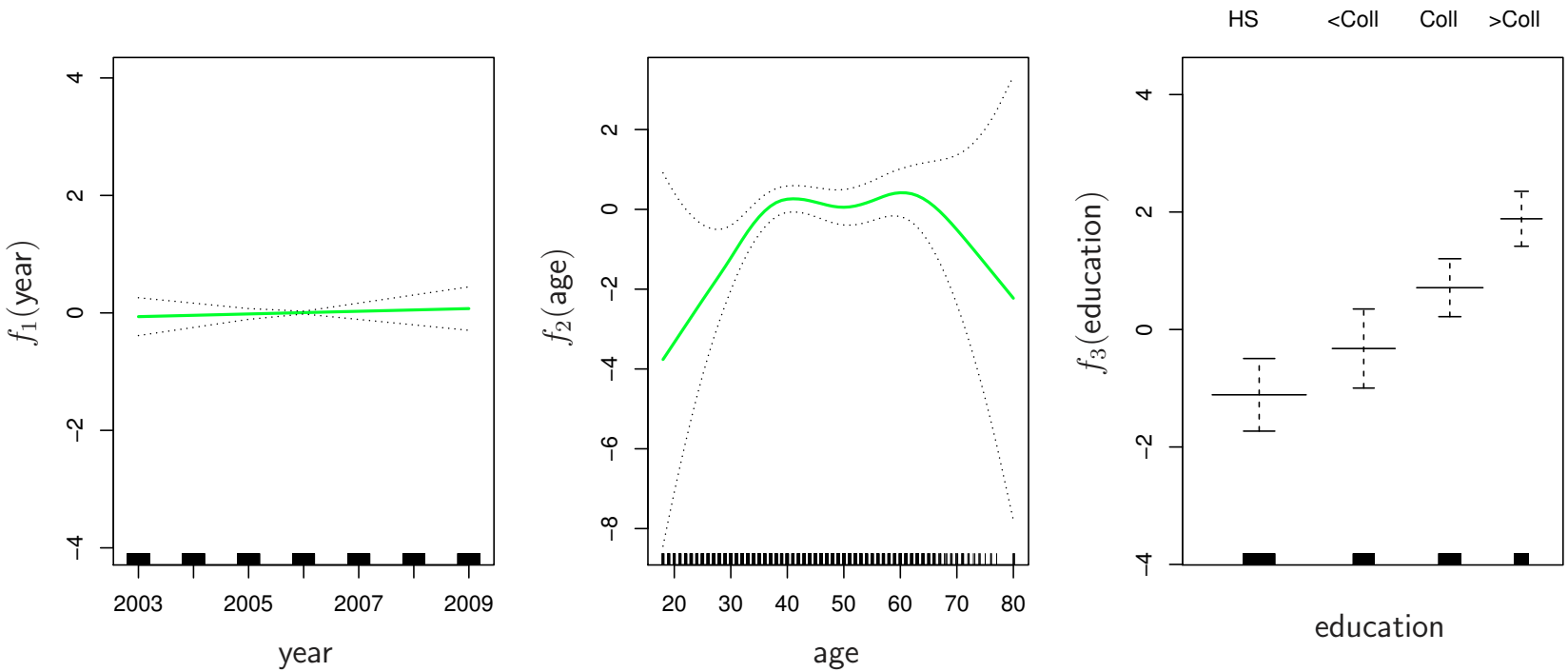
- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.
- Can use smoothing splines or local regression as well:

  $$\texttt{gam}(\texttt{wage} \sim \texttt{s}(\texttt{year}, \texttt{df} = 5) + \texttt{lo}(\texttt{age}, \texttt{span} = .5) + \texttt{education})$$

- GAMs are additive, although low-order interactions can be included in a natural way using, e.g. bivariate smoothers or interactions of the form `ns(age,df=5):ns(year,df=5)`.

GAMs for classification

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



`gam(I(wage > 250) ~ year + s(age, df = 5) + education, family = binomial)`