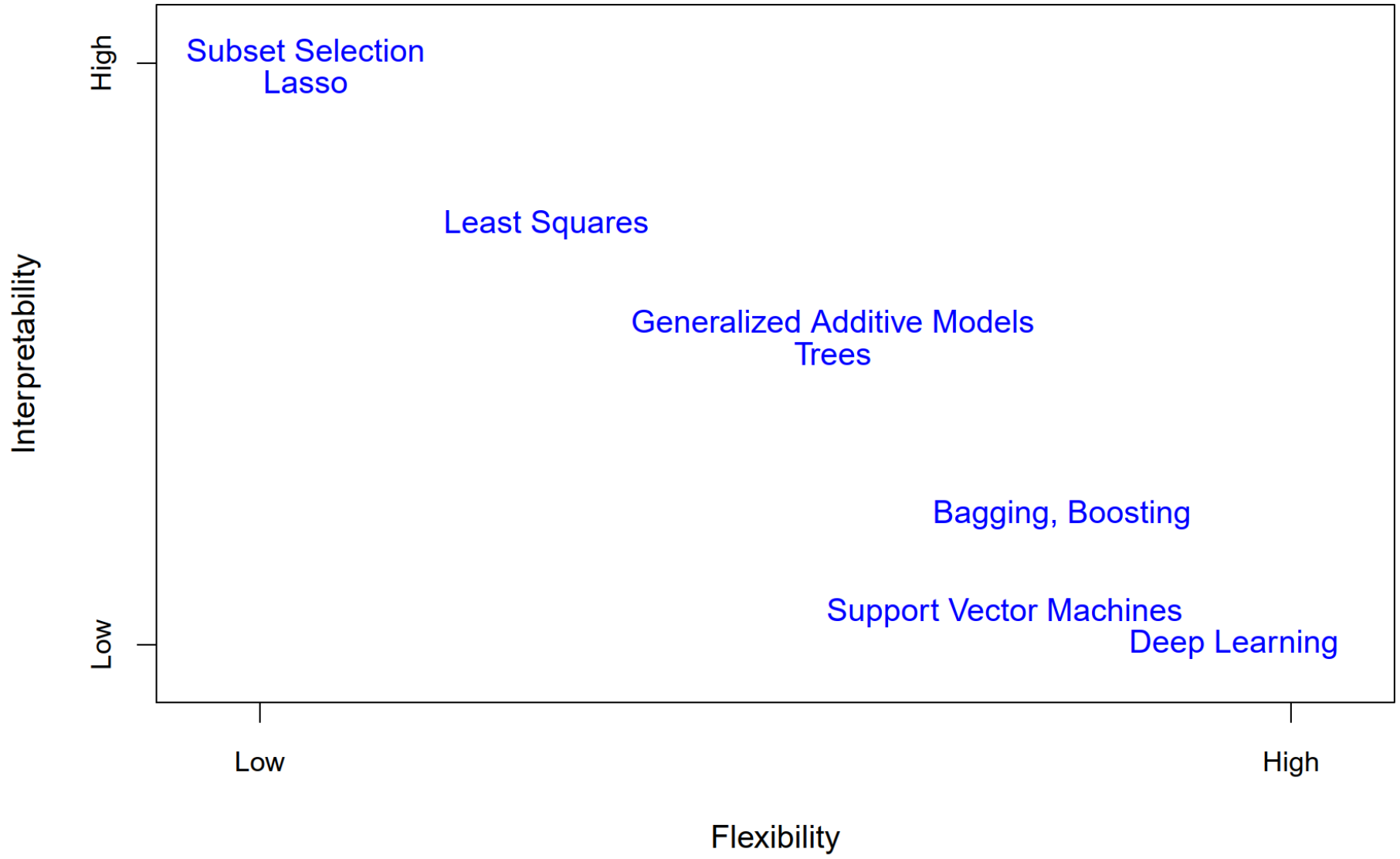
Moving Beyond Linearity

Szu-Chi Chung

Department of Applied Mathematics, National Sun Yat-sen University


Subset Selection
Lasso
Least Squares
Generalized Additive Models
Trees
Bagging, Boosting
Support Vector Machines
Deep Learning
High
Low
Interpretability
Low
High
Flexibility

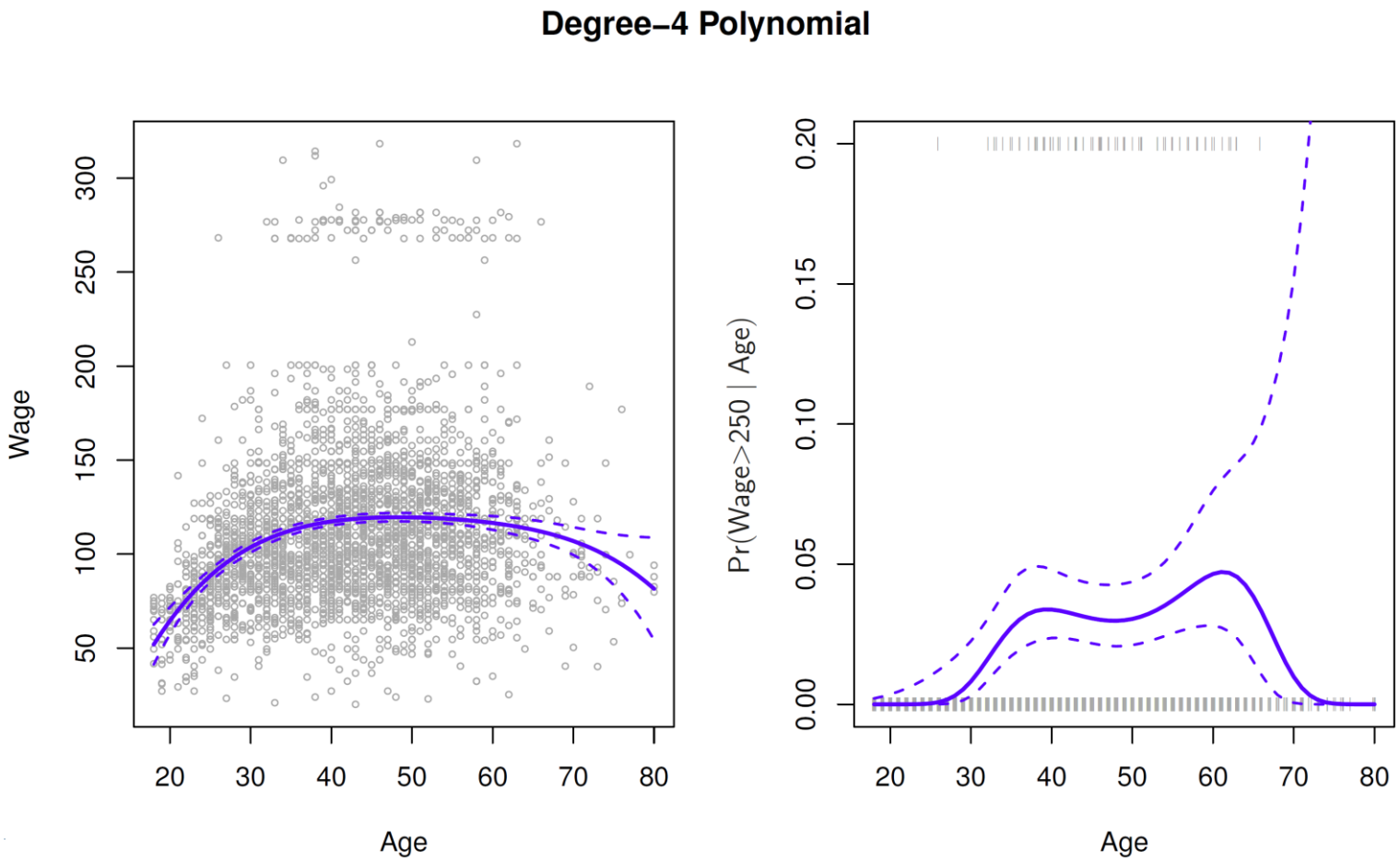
Moving Beyond Linearity

- 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

1. Polynomial Regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_p^2 + \epsilon_i$$



Details

- 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 (pointwise):

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

- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_l$, 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 $\hat{f}(x_0) \pm 2 \cdot se\left[\hat{f}(x_0)\right]$
- We either fix the degree p at some reasonably low value, else use cross-validation to choose p

Details continued

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

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_p^p)}{1 + \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_p^p}$$

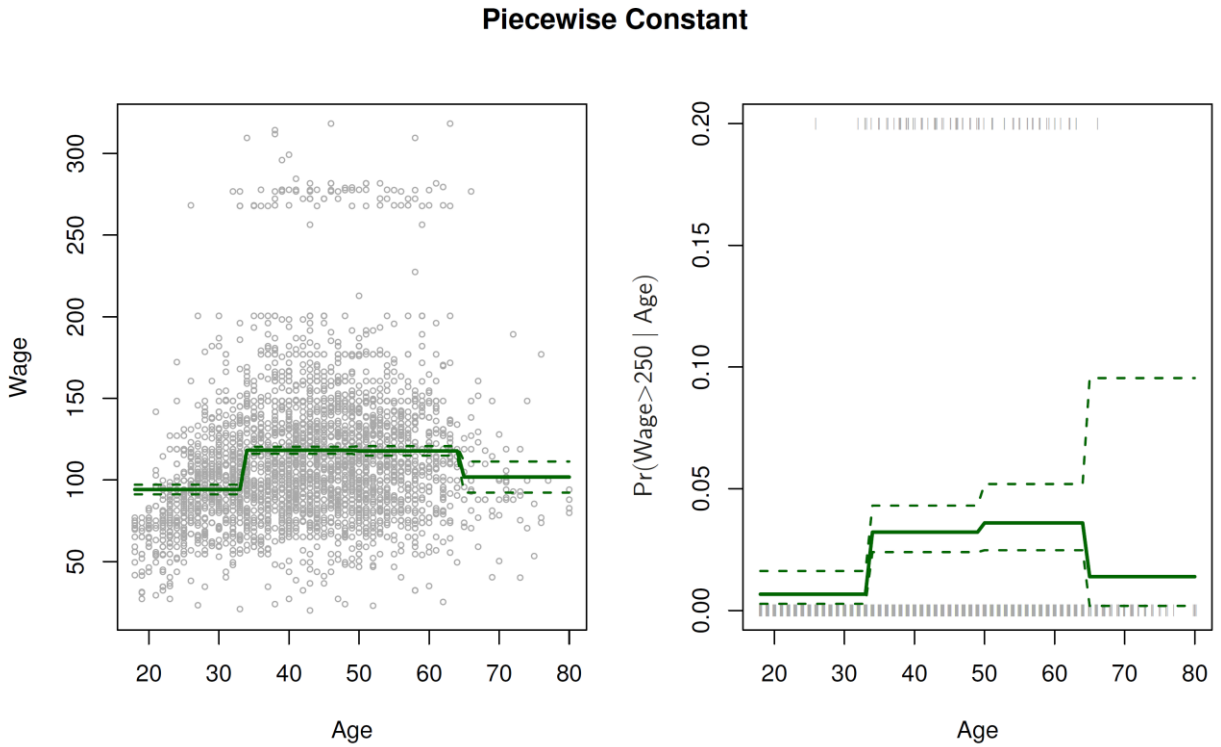
  - To get confidence intervals, compute upper and lower bounds on the logit scale, and then invert to get on probability scale
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later)
- Can fit using poly = PolynomialFeatures(4) in formula
- Caveat: polynomials have notorious tail behavior — very bad for extrapolation

2. Step Functions (Piecewise-constant)

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

$$C_0(X) = I(X < 35), C_1(X) = I(35 \leq X < 50), \ldots, C_3(X) = I(X \geq 65)$$

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon_i$$



Step functions continued

- 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:

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

would allow for different linear functions in each age category

- In Python: pd.cut(age,4) or pd.cut(age, [0, 25, 40, 65, 90])
- Choice of cutpoints or knots can be problematic. For creating nonlinearities, smoother alternatives such as splines are available

Basis function

- Instead of fitting the model in X, we fit

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_1 b_1(x_i) + \cdots + \beta_K b_K(x_i) + \epsilon_i$$

- For polynomial regression, $b_j(x_i) = x_i^j$
- For step function, $b_j(x_i) = I(c_j \leq x_i < c_{j+1})$
- We can think of the model as a standard linear model with predictors $b_1(x_i), b_2(x_i), \ldots, b_K(x_i)$
  - We can use least squares to estimate the unknown regression coefficients
  - All of the inference tools for linear models can be used, such as standard errors for the coefficient estimates and F-statistics for the model's overall significance…

3.1 Regression Splines - Linear Splines

- How can we fit a piecewise degree-d polynomial under the constraint that it (and possibly its first $d-1$ derivatives) be continuous?
- It turns out that we can use the basis function to represent the regression spline
- 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

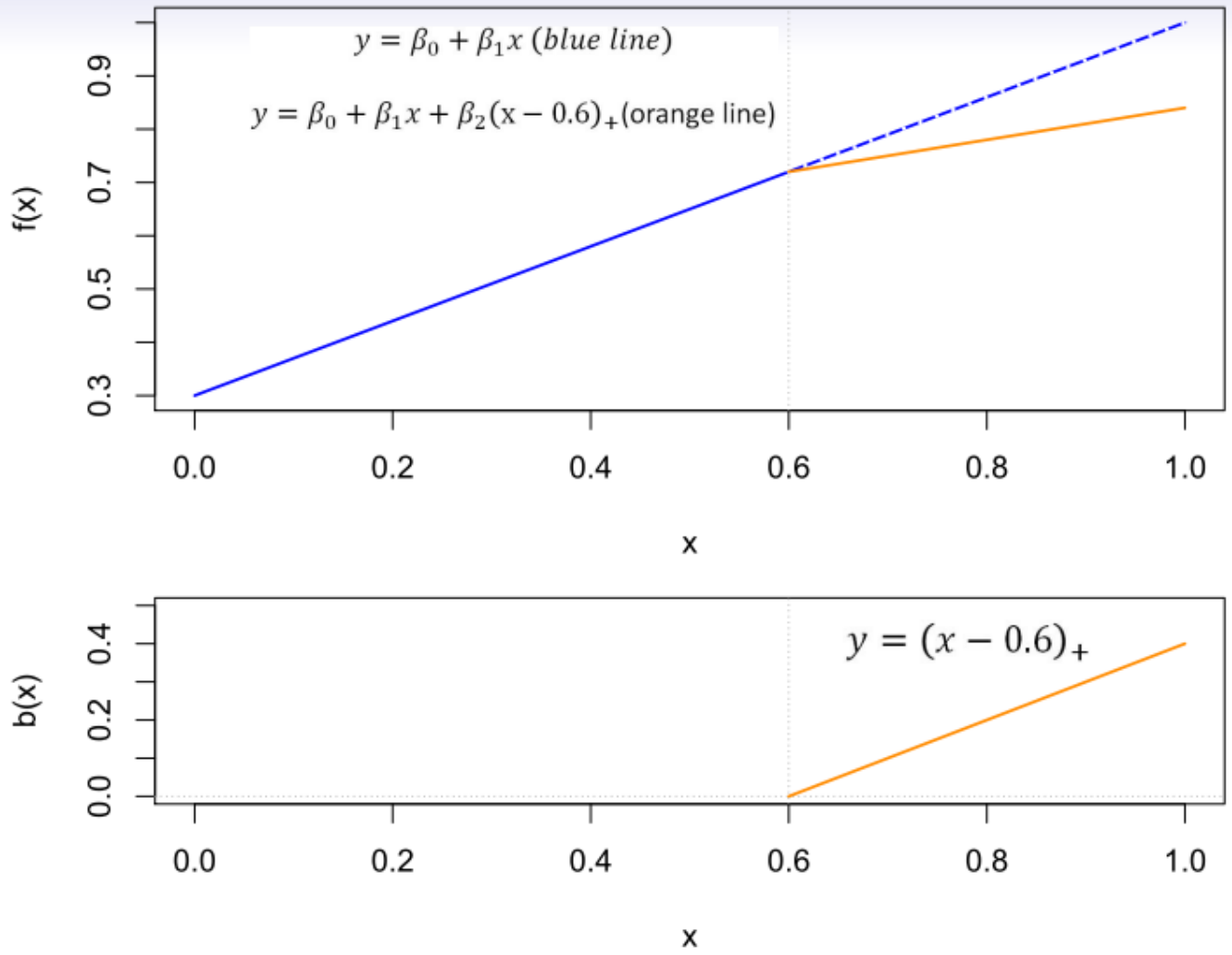
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+1} b_{K+1}(x_i) + \epsilon_i$$

Where the b_k are basis functions (with one more truncated basis per knot)

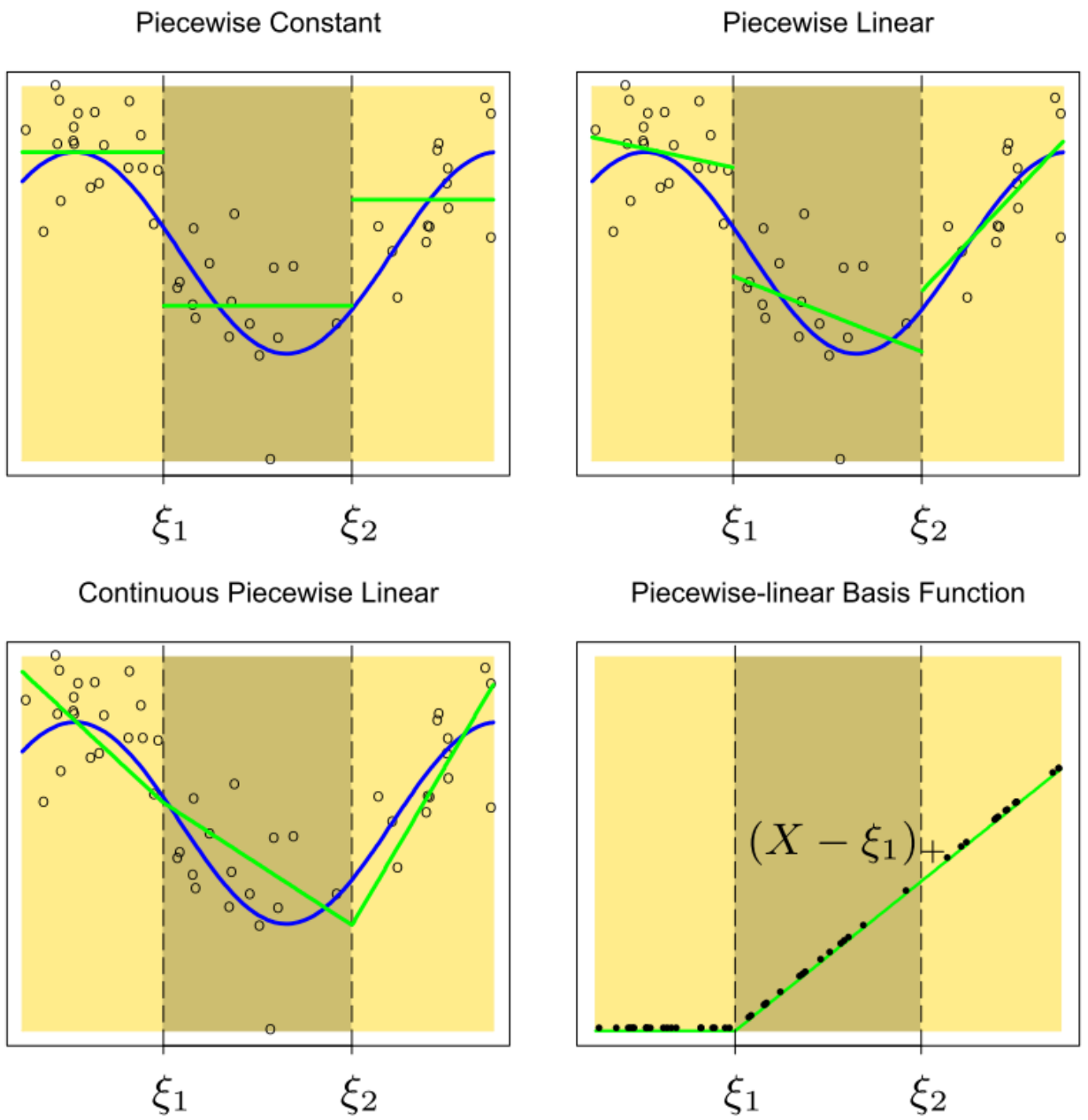
$$b_1(x_i) = x_i$$
$$b_{k+1}(x_i) = (x_i - \xi_k)_+, \qquad k = 1, \ldots, K$$

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

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


$y = \beta_0 + \beta_1 x$ (blue line)
$y = \beta_0 + \beta_1 x + \beta_2(x - 0.6)_+$(orange line)
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
$y = (x - 0.6)_+$
0.0
0.2
0.4
0.6
0.8
1.0
x

The blue curve represents the true function

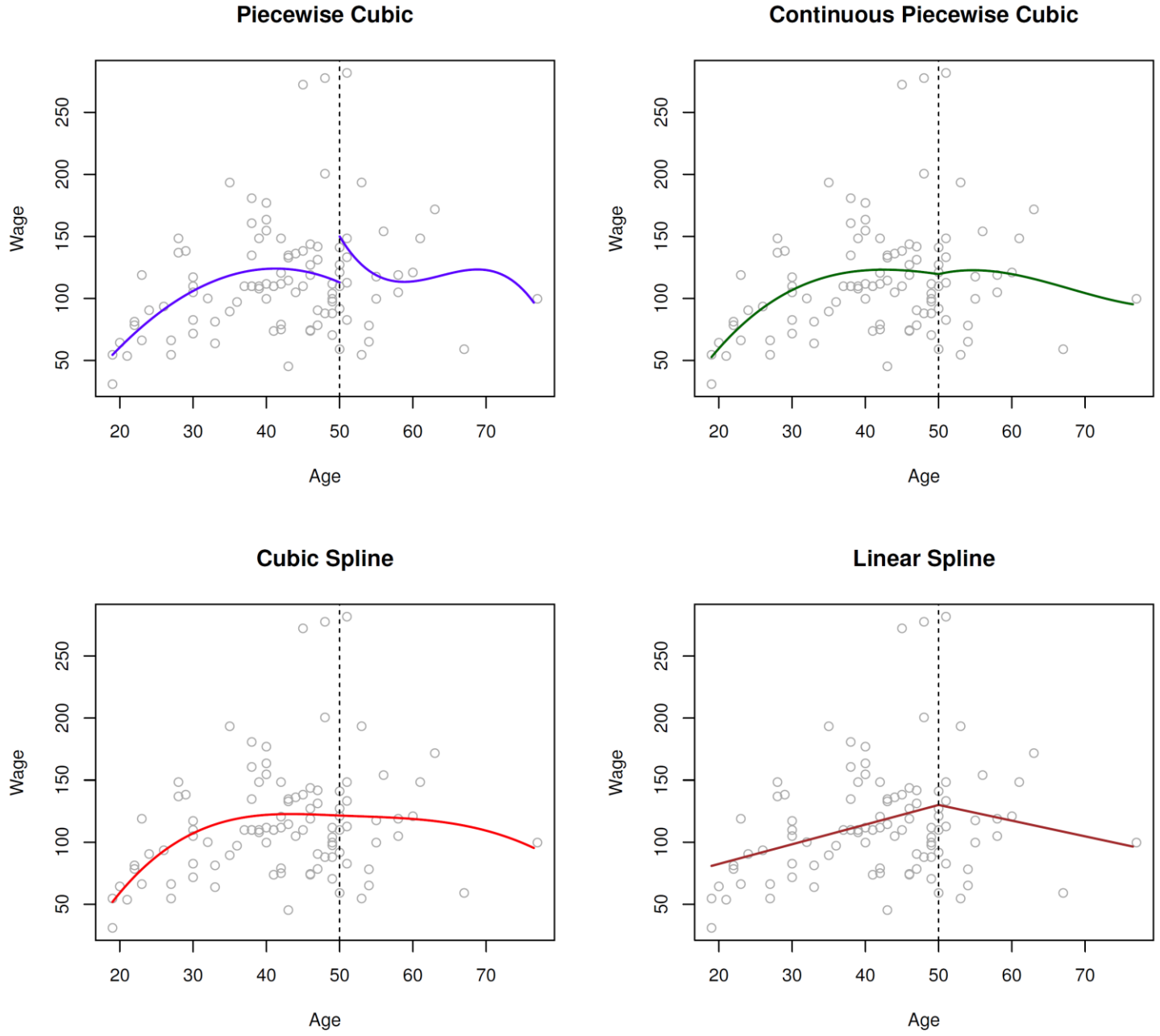


3.2 Regression Splines - Piecewise Polynomials

- Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by *knots*

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

- If we place K different knots throughout the range of X, then we will end up fitting $K + 1$ different cubic polynomials
- Better to add constraints to the polynomials, e.g. continuity
- The general definition of a degree-d spline is that it is a piecewise degree-d polynomial, with continuity in derivatives up to degree $d - 1$ at each knot


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

3.3 Regression Splines - Cubic Splines

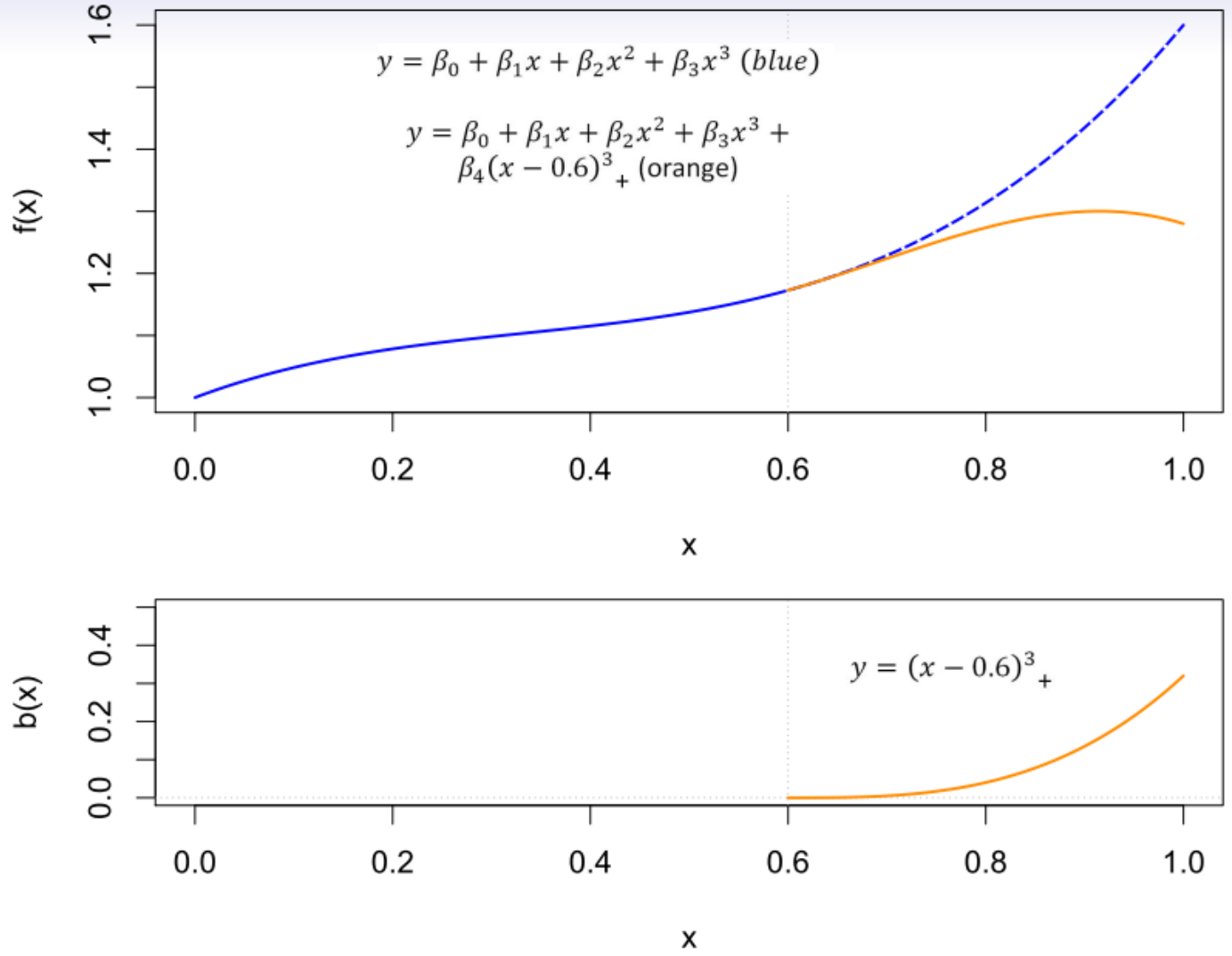
- A cubic spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot
- Again we can represent this model with truncated power basis functions ($K + 4$ degrees of freedom)

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

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

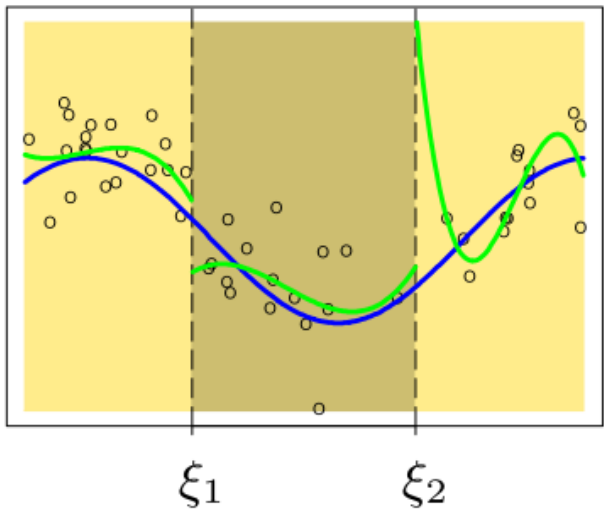
Where

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

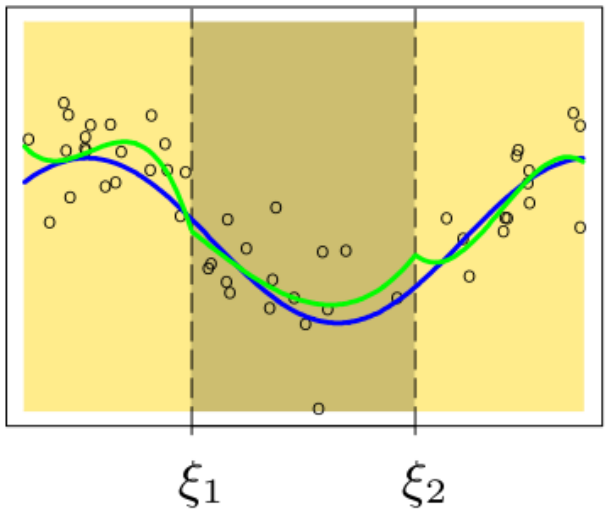

$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$ (blue)
$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - 0.6)^3_+$ (orange)
f(x)
1.0
1.2
1.4
1.6
0.0
0.2
0.4
0.6
0.8
1.0
x
$y = (x - 0.6)^3_+$
b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

Piecewise Cubic Polynomials

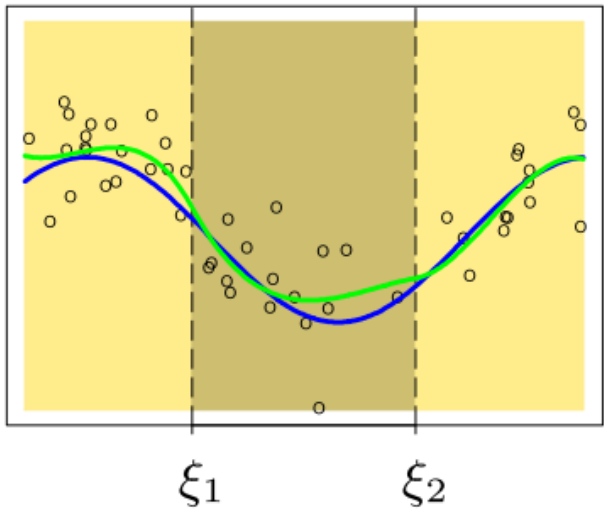
Discontinuous



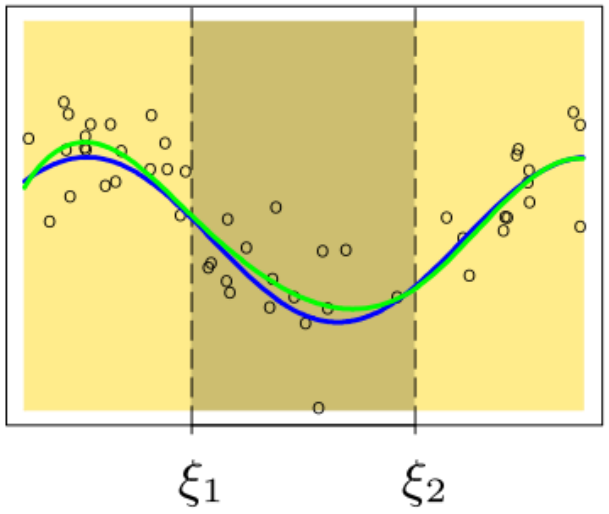
Continuous



Continuous First Derivative

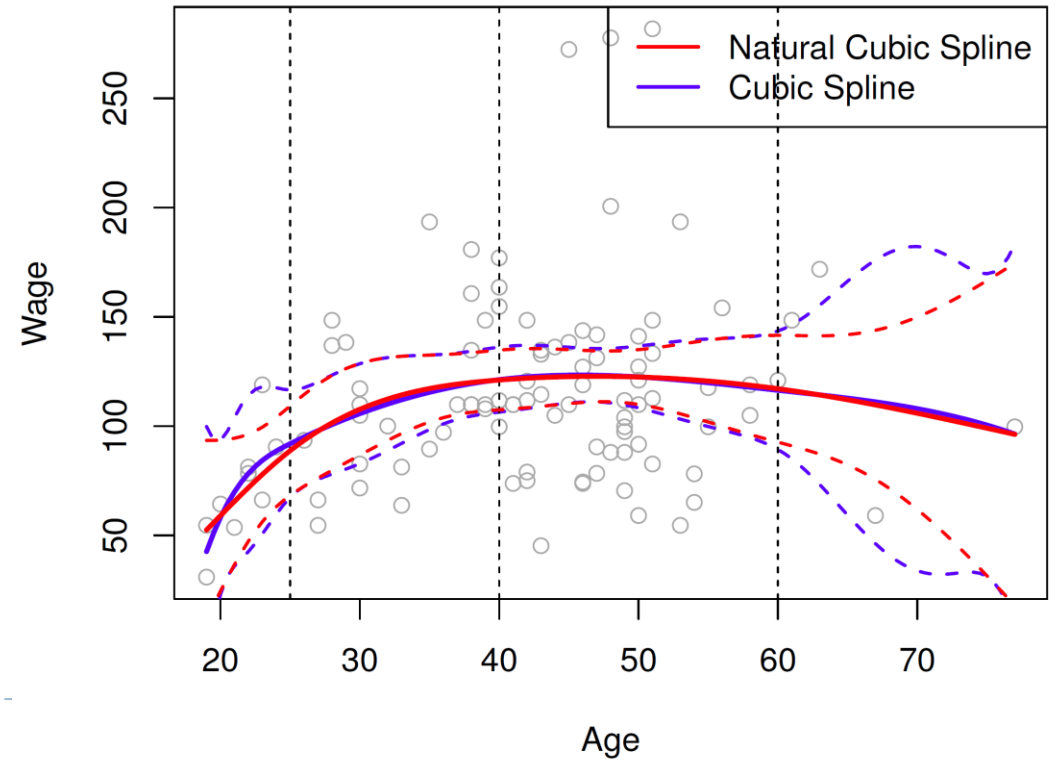


Continuous Second Derivative



3.4 Regression Splines - 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
  - It has zero 2nd and 3rd derivative outside the boundary knots



Natural Cubic Spline Basis

- A natural cubic spline model with K knots is represented by K basis functions:

$$f(x_i) = y_i = \beta_0 b_0(x_i) + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K-1} b_{K-1}(x_i) + \epsilon_i$$

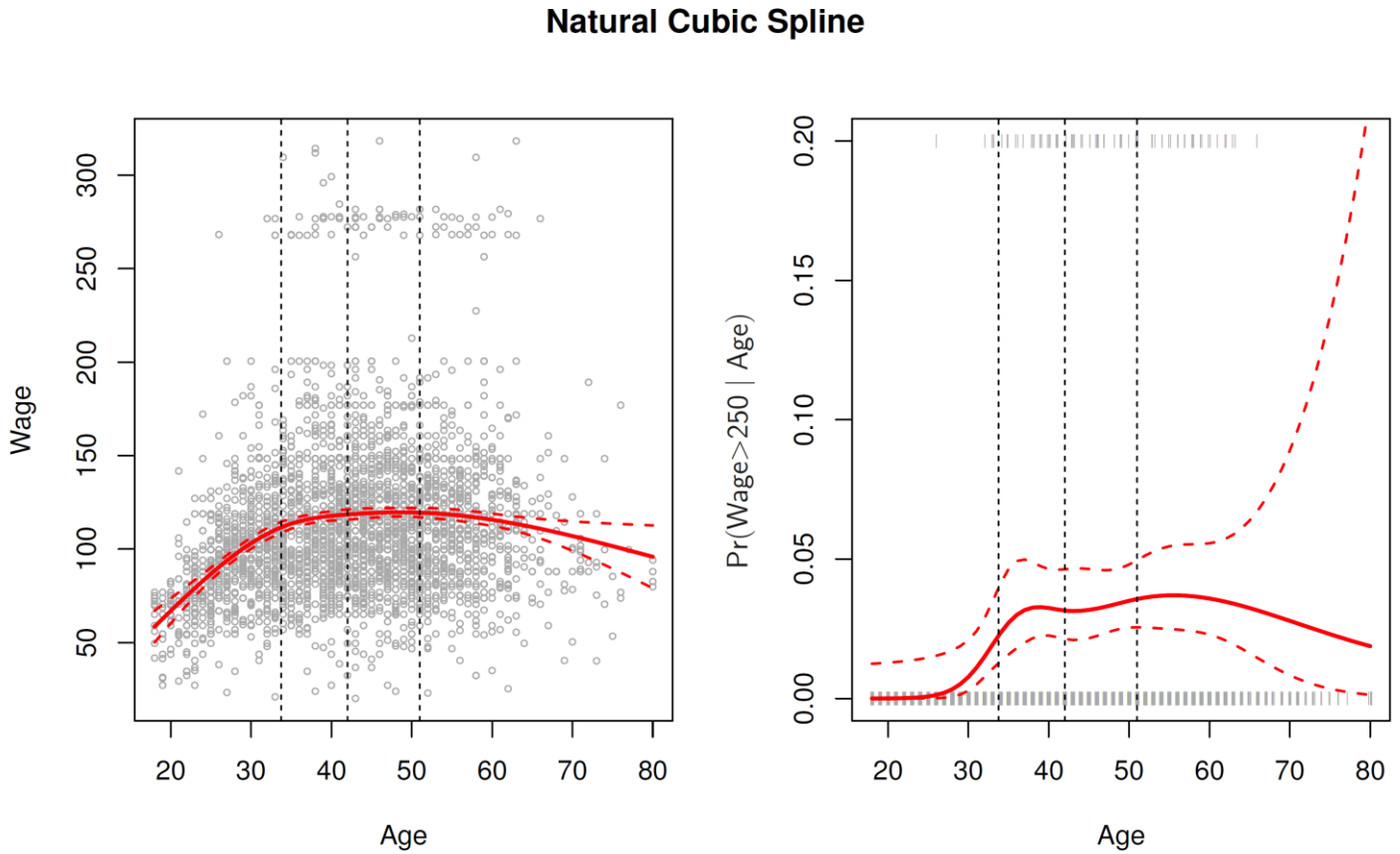
$$b_0(x_i) = 1$$
$$b_1(x_i) = x_i$$
$$b_{k+1}(x_i) = d_{k-1}(x_i) - d_{K-2}(x_i)$$

where

$$d_k(x_i) = \frac{(x_i - \xi_{k+1})^3 - (x_i - \xi_K)^3}{\xi_K - \xi_{k+1}}, \qquad k = 0, \ldots, K-2$$

Regression Splines

- Fitting splines in Python is easy: *bs*(x, ...) for any degree splines, and *cr*(x, ...) for natural cubic splines with any degree of freedom



Regression Splines

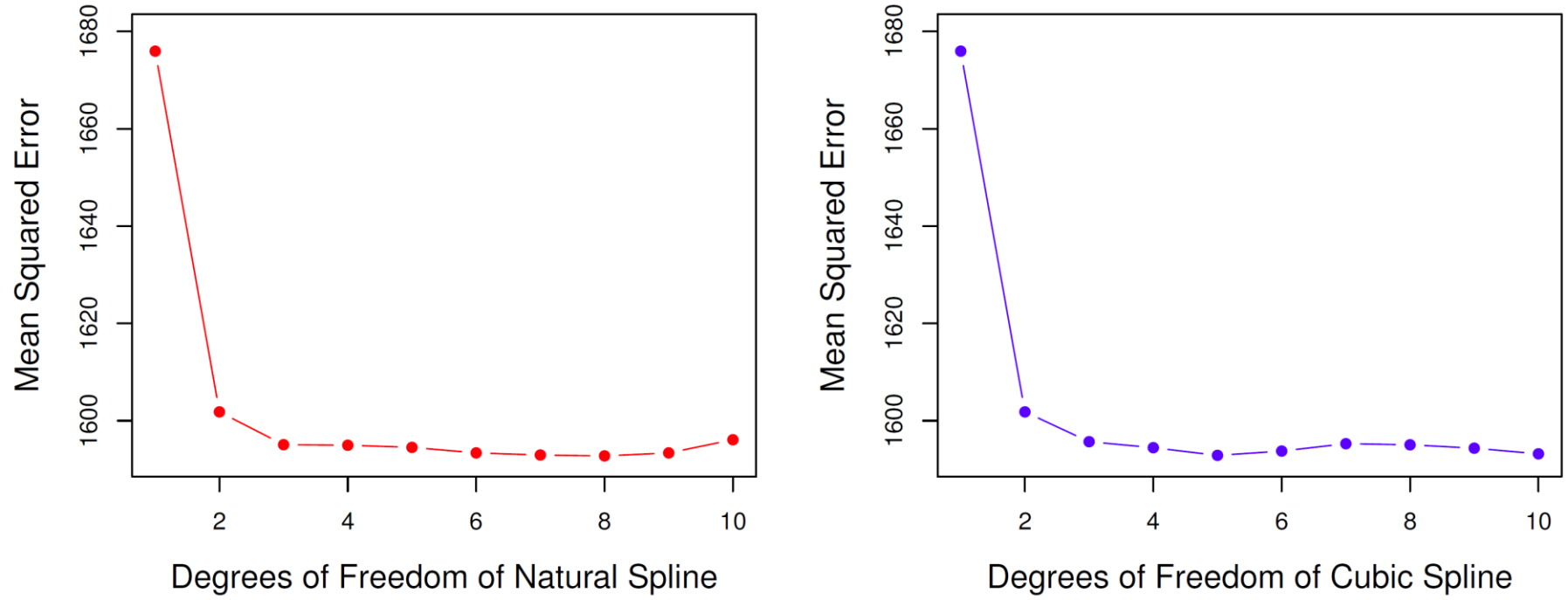
- The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly
  - Hence, one option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable
  - While this option can work well, in practice it is common to place knots in a uniform fashion
    - We have fit a natural cubic spline with three knots, the knot locations were chosen automatically as the 25th, 50th, and 75th percentiles of age
    - Another way to do this is to specify the desired degrees of freedom, and then have the software automatically place the corresponding number of knots at uniform quantiles of the data

Regression Splines

- How many knots should we use, or equivalently how many degrees of freedom should our spline contain?
  - One option is to try out different numbers of knots and see which produces the best looking curve
- A more objective approach is to use cross-validation
  - With this method, we remove a portion of the data (say 10%), fit a spline with a certain number of knots to the remaining data, and then use the spline to make predictions for the held-out portion
  - We repeat this process multiple times until each observation has been left out once, and then compute the overall cross-validated RSS
  - This procedure can be repeated for different numbers of knots K
  - Then the value of K giving the smallest test RSS is chosen

Regression Splines

- Ten-fold cross-validated mean squared errors for selecting the degrees of freedom when fitting splines to the Wage data. The response is wage and the predictor age. Left: A natural cubic spline. Right: A cubic spline

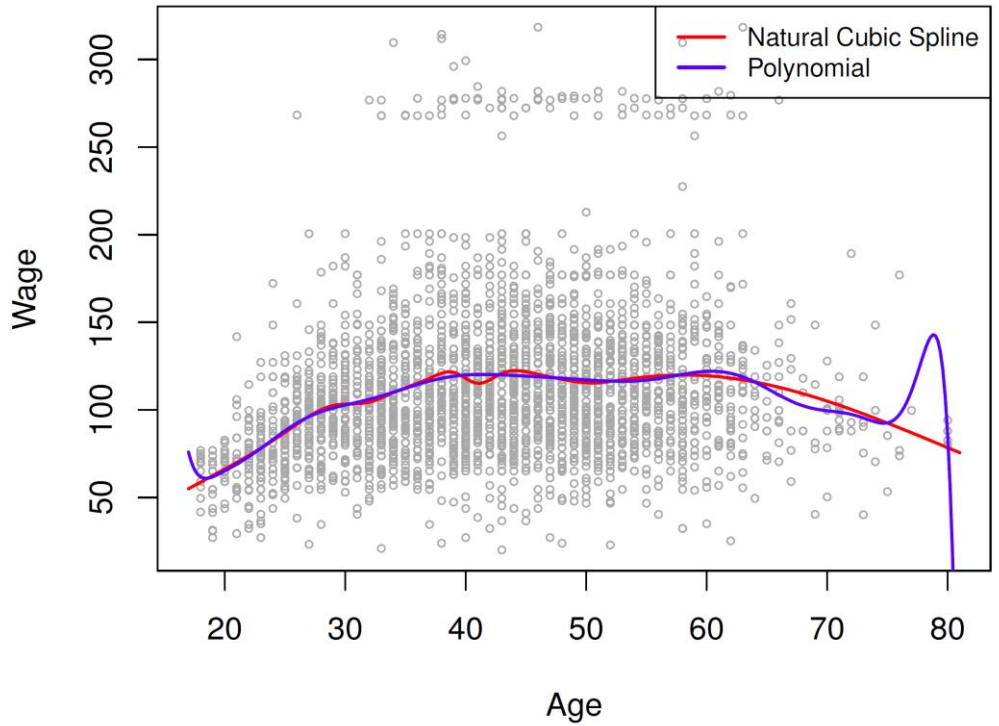


Regression Splines

- Figure 7.6 shows ten-fold cross-validated mean squared errors for splines with various degrees of freedom fit to the Wage data
- The left-hand panel corresponds to a natural spline and the right-hand panel to a cubic spline
- The two methods produce almost identical results, with clear evidence that a one-degree fit (a linear regression) is not adequate
- Both curves flatten out quickly, and it seems that three degrees of freedom for the natural spline and four degrees of freedom for the cubic spline are quite adequate

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 $K - 1$ degrees of freedom



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

4. Smoothing Splines

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

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

- 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

- The unique minimizer of this penalized RSS is a shrunken version of natural cubic spline with knots at the unique values of x_i, $i = 1, \ldots, n$
- Seems like there will be n features and presumably overfitting of the data. But, the smoothing term shrinks the model towards the linear fit
  - Effective degrees of freedom decrease from $n - 1$ to 2 as we increase λ
- Solution:

$$f(x) = \sum_{i=1}^{n} b_i(x)\beta_i$$

Where $b_1(x_i) = 1, b_2(x_i) = x_i, b_{i+1}(x_i) = d_{i-1}(x_i) - d_{n-2}(x_i), i = 2, \ldots, n-1$ are the Natural Cubic Spline Basis

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
  - The vector of n fitted values can be written as $\hat{g}_\lambda = S_\lambda y$, where S_λ is a $n \times n$ matrix (determined by the x_i and λ)
  - The effective degrees of freedom are given by

$$df_\lambda = \sum_{i=1}^{n} \{S_\lambda\}_{ii}$$

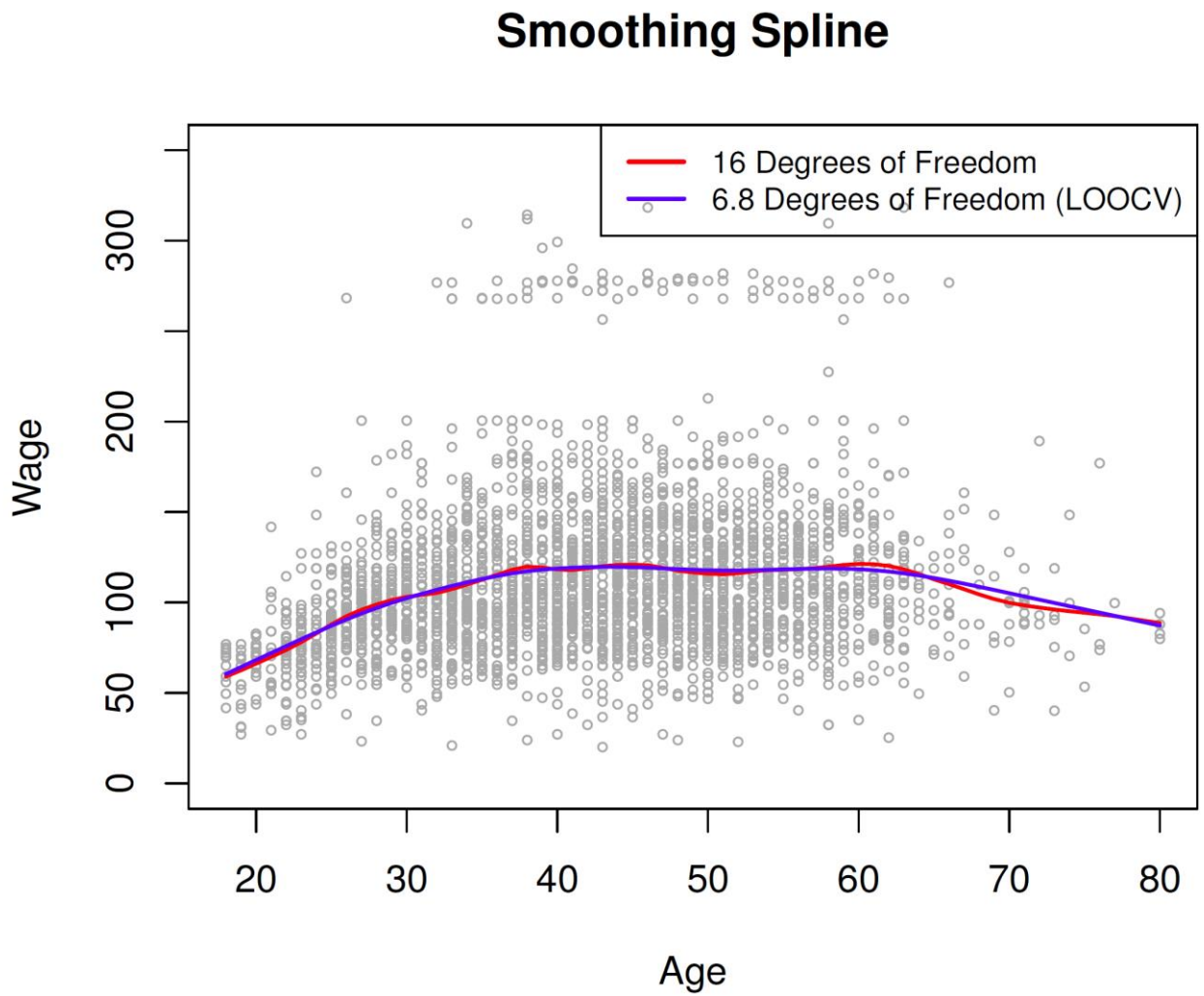
Smoothing Splines – choosing λ

- We can specify df rather than λ!
- The leave-one-out (LOO) cross-validated error is given by

$$RSS_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_{\lambda}^{(-i)})^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_{\lambda}(x_i)}{1 - \{S_{\lambda}\}_{ii}} \right]^2$$

- The notation $\hat{g}_{\lambda}^{(-i)}$ indicates the fitted value for this smoothing spline evaluated at x_i, where the fit uses all of the training observations except for the ith observation (x_i, y_i)

Smoothing Splines



5. Local Regression

- Local regression computes the fit at a target point x_0 using only the regression nearby training observations, which is described in Algorithm 7.1
- In Step 3 of Algorithm 7.1, the weights K_{i0} will differ for each value of x_0.
- At a new point, we need to fit a new weighted least squares regression model by minimizing (7.14) for a new set of weights
- Local regression is referred to as a *memory-based* procedure, because we need all the training data each time we wish to compute a prediction

Algorithm 7.1 *Local Regression At* $X = x_0$

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

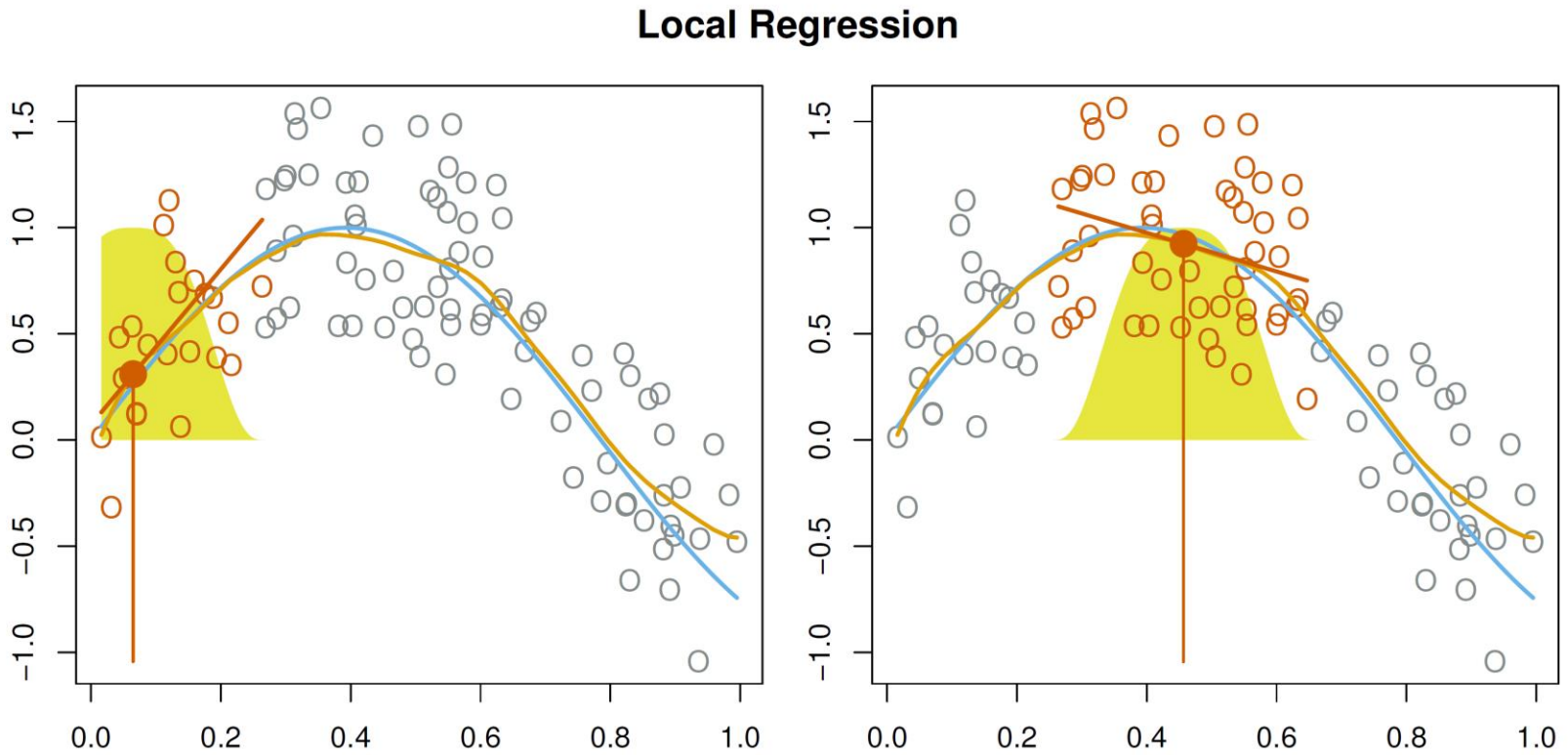
4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Local Regression

- Figure next page illustrates a simulated data, with one target point near 0.4, and another near the boundary at 0.05
- The blue line represents the function $f(x)$ from which the data were generated, and the light orange line corresponds to the local regression estimate $\hat{f}(x)$
- The yellow bell-shape indicates weights assigned to each point, decreasing to zero with distance from the target point. The fitted value $\hat{f}(x_0)$ at x_0 is obtained by fitting a weighted linear regression (orange line segment)

Local Regression

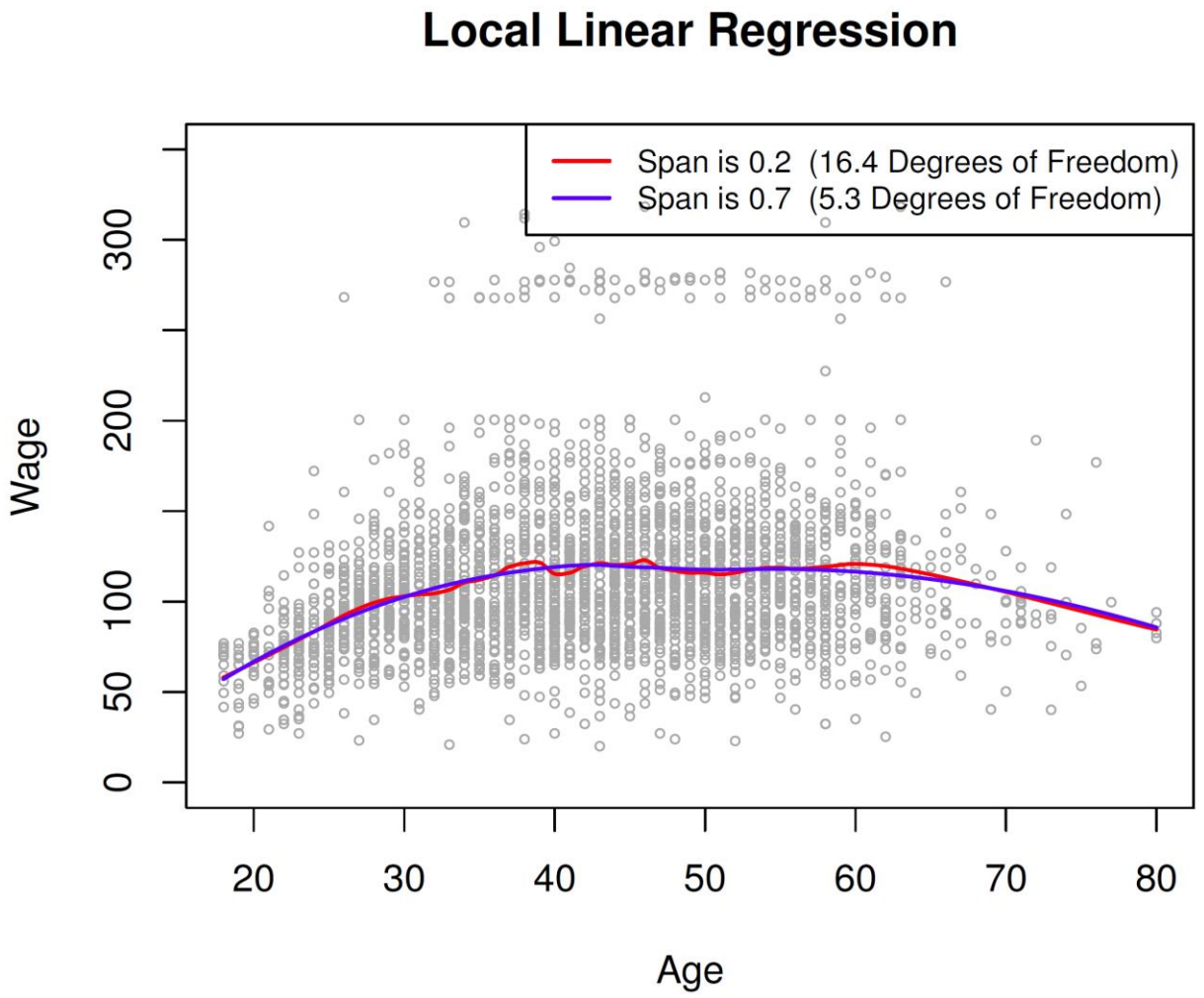
- With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares
- See text for more details, and *lowess()* function in Python



Local Regression

- Choices: span s (Step 1), the weighting function K (Step 2), and whether to fit a linear, constant, or quadratic regression (Step 3)
- The most important choice is the span s, similar to the tuning parameter λ in smoothing splines: it controls the flexibility of the non-linear fit
  - The smaller the value of s, the more local and wiggly will be our fit; alternatively, a very large value of s will lead to a global fit to the data using all of the training observations
  - We can use cross-validation to choose s, or we can specify it directly
- Figure 7.10 displays local linear regression fits on the Wage data, using two values of s: 0.7 and 0.2
  - The fit obtained using $s = 0.7$ is smoother than that obtained using $s = 0.2$

Local Regression



Local Regression

- In a setting with multiple features $X_1, X_2, \ldots, X_p$, one very useful generalization involves fitting a multiple linear regression model that is global in some variables, but local in another, such as time. Such varying coefficient models are a useful way of adapting a model to the most recently gathered data

- Local regression also generalizes very naturally when we want to fit models that are local in a pair of variables X_1 and X_2, rather than one. We can simply use two-dimensional neighborhoods, and fit bivariate linear regression models using the observations that are near each target point in two-dimensional space.

GAMs for Regression Problems

- A natural way to extend the multiple linear regression model

$$y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \epsilon_i$$

in order to allow for non-linear relationships between each feature and the response

- The generalized additive model (GAM) can be written as

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

- It is called an additive model because we calculate a separate f_j for each X_j, and then add together all of their contributions

GAMs for Regression Problems

- The beauty of GAMs is that we can use these methods as building blocks for fitting an additive model. Consider the task of fitting the model on the Wage data:

$$wage = \beta_0 + f_1(year) + f_2(age) + f_3(education) + \epsilon$$

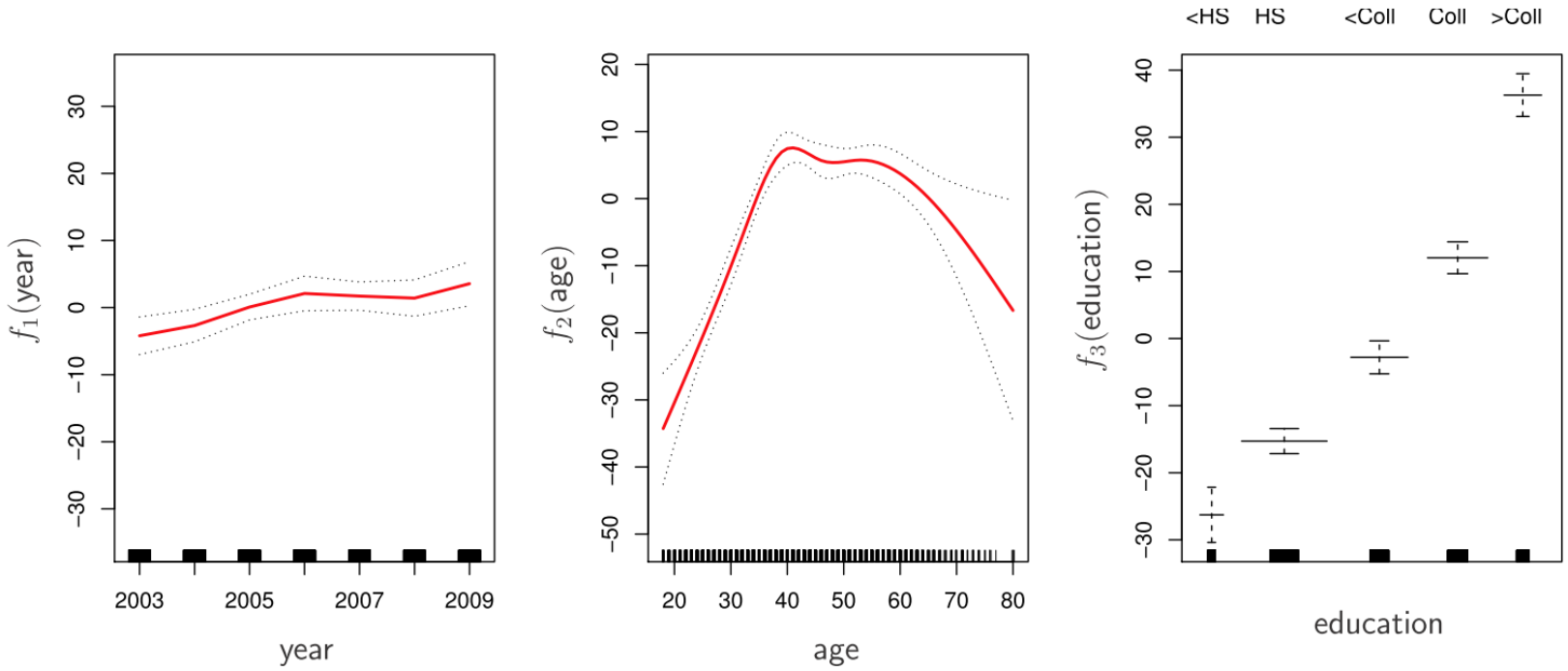
  - *Year* and *age* are quantitative variables
  - *Education* is a qualitative variable with five levels: <HS, HS, <Coll, Coll, >Coll, referring to the amount of high school or college education that an individual has completed
  - f_1 and f_2: natural splines
  - f_3: separate constant for each level, via the dummy variable approach

- Figure next page shows the fitted model using least squares

Generalized Additive Models

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

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

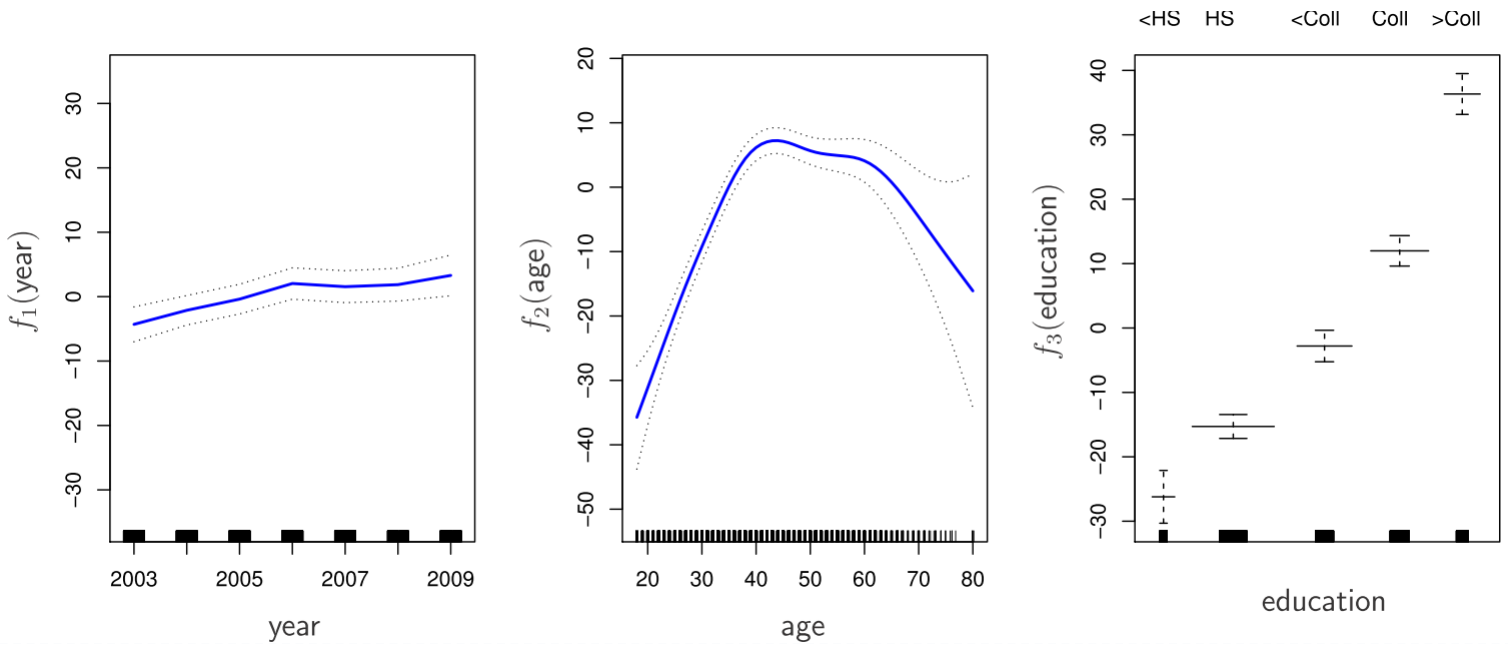


Generalized Additive Models

- The above figure can be easily interpreted
- The left-hand panel indicates that holding age and education fixed, wage tends to increase slightly with year; this may be due to inflation
- The center panel indicates that holding education and year fixed, wage tends to be highest for intermediate values of age, and lowest for the very young and very old
- The right-hand panel indicates that holding year and age fixed, wage tends to increase with education: the more educated a person is, the higher their salary, on average

Generalized Additive Models

- Figure 7.12 shows a similar triple of plots, but this time f_1, f_2: smoothing splines with 4 and 5 d.f.s, respectively
  - The fitted functions in Figure 7.12 and the previous figure look rather similar
  - In most situations, the differences in the GAMs obtained using smoothing splines versus natural splines are small



Pros and Cons of GAMs

- The advantages and limitations of a GAM
  - GAMs allow us to fit a non-linear f_j to each X_j , so that we can automatically model non-linear relationships
  - The non-linear fits can potentially make more accurate predictions for the response Y
  - Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed
  - The smoothness of the function f_j for the variable X_j can be summarized via degrees of freedom

Pros and Cons of GAMs

- The advantages and limitations of a GAM
  - The main limitation of GAMs is that the model is restricted to be additive: important interactions can be missed
  - We can manually add interaction terms to the GAM model by including additional predictors of the form $X_j \times X_k$
  - In addition, we can add low-dimensional interaction functions of the form $f_{jk}(X_j, X_k)$ into the model; such terms can be fit using two-dimensional smoothers such as local regression, or two-dimensional splines
- For fully general models, we have to look for even more flexible approaches such as random forests and boosting, described in Chapter 8. GAMs provide a useful compromise between linear and fully nonparametric models

GAMs for Classification Problems

- GAMs can also be used when Y is qualitative

- Assume Y takes on values zero or one, and let

$$p(X) = \Pr(Y = 1|X)$$

- Extend the logistic regression model to the logistic regression GAM:

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

GAMs for Classification Problems

- We fit a GAM to the Wage data to predict the probability that an individual's income exceeds \$250,000 per year:

$$p(X) = \Pr(wage > 250 | year, age, education)$$

- Consider the GAM:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 \times year + f_2(age) + f_3(education)$$

- f_2: smoothing spline with five d.f.
- f_3: a step function, with dummy variables for each level of education
- The resulting fit is shown in Figure 7.13

GAMs for Classification Problems

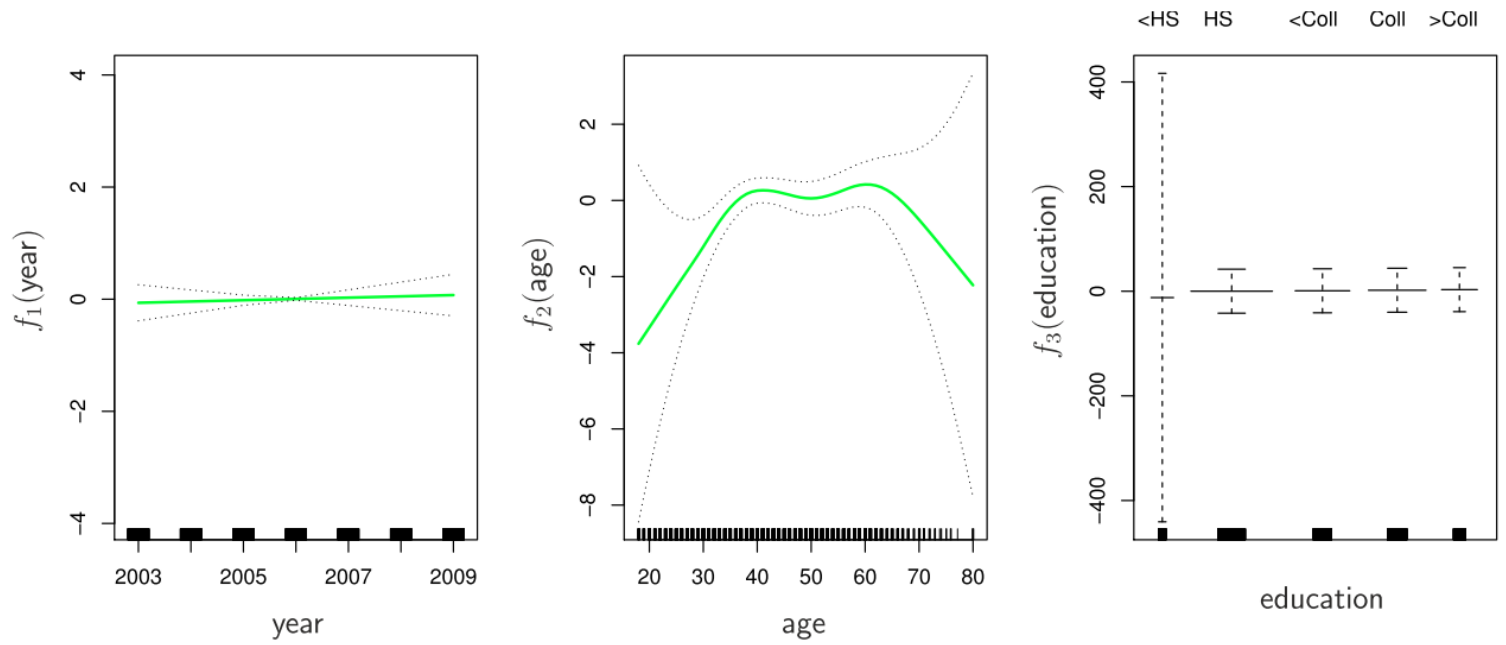


FIGURE 7.13. *For the* `Wage` *data, the logistic regression GAM given in (7.19) is fit to the binary response* `I(wage>250)`*. Each plot displays the fitted function and pointwise standard errors. The first function is linear in* `year`*, the second function a smoothing spline with five degrees of freedom in* `age`*, and the third a step function for* `education`*. There are very wide standard errors for the first level* `<HS` *of* `education`*.*

GAMs for Classification Problems

- The last panel looks suspicious, with very wide confidence intervals for level <HS. In fact, there are no ones for that category: no individuals with less than a high school education make more than $250000 per year
- Hence we refit the GAM, excluding the individuals with less than a high school education
- The resulting model is shown in Figure 7.14

GAMs for Classification Problems

We observe that age and education have a much larger effect than year on the probability of being a high earner

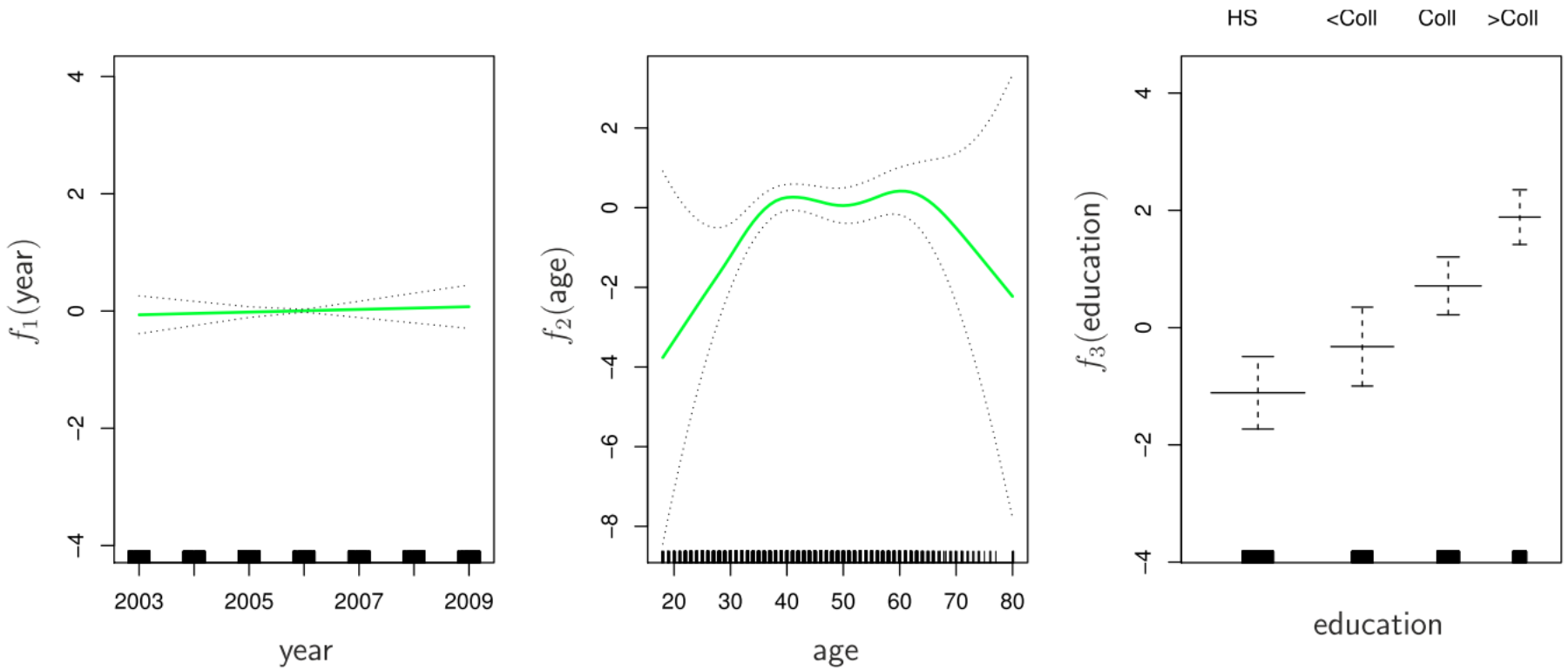


FIGURE 7.14. *The same model is fit as in Figure 7.13, this time excluding the observations for which* `education` *is* `<HS`*. Now we see that increased education tends to be associated with higher salaries.*

Appendix

Natural Cubic Spline Basis

- Since each of these basis functions has zero 2nd and 3rd derivative outside the boundary knots, so does $f(x_i)$ defined below
- When $x_i \leq \xi_k$

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k b_k(x_i) = \beta_0 + \beta_1 x_i \rightarrow f^{(2)}(x_i) = f^{(3)}(x_i) = 0$$

- When $x_i > \xi_k$

$$b_{k+1}(x_i) = \frac{(x_i - \xi_k)^3 - (x_i - \xi_K)^3}{\xi_K - \xi_k} - \frac{(x_i - \xi_{K-1})^3 - (x_i - \xi_K)^3}{\xi_K - \xi_{K-1}}$$

Since $b_{k+1}^{(2)}(x_i) = b_{k+1}^{(3)}(x_i) = 0, \forall\, k = 1, \ldots, K-1 \rightarrow f^{(2)}(x_i) = f^{(3)}(x_i) = 0$

Smoothing Splines (ESL 5.4)

- The objective function is the penalized RSS

$$RSS(\beta, \lambda) = (y - H\beta)^T (y - H\beta) + \lambda \beta^T \Omega_H \beta$$

Where $y = (y_1, \ldots, y_n)', \beta = (\beta_1, \ldots, \beta_n)'$ and

$(H)_{ij} = h_j(x_i)$, and $\Omega_{jk} = \int h_j''(t) h_k''(t)\, dt$

- The estimate

$$\hat{\beta} = (H^T H + \lambda \Omega_H)^{-1} H^T y = S_\lambda y$$

- This is a generalized ridge regression (when $\Omega_H = I$, it's a ridge regression)
- Can show that $S_\lambda = (I + \lambda M)^{-1}$ where the matrix M does not depend on λ

Nonparametric Logistic Regression (ESL 5.6)

- Consider logistic regression with a single x:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = g(x)$$

And a penalized log-likelihood criterion

$$l(g,\lambda) = \sum_{i=1}^{N}\{y_i \log(p(x_i)) + (1-y_i)\log(1-p(x_i))\} - \frac{1}{2}\lambda\int g''(t)^2 dt$$

$$= \sum_{i=1}^{N}\{y_i g(x_i) - \log(1+e^{g(x_i)})\} - \frac{1}{2}\lambda\int g''(t)^2 dt$$

- Again can show that the optimal g is a natural spline with knots at the datapoint
- Can use Newton-Raphson to do the fitting

Thin-Plate Splines (Multidimensional Splines) (ESL 5.7)

- The discussion up to this point has been one-dimensional. The higher-dimensional analogue of smoothing splines are "thin-plate splines." In 2-D, instead of minimizing

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

Minimized

$$\min_{g \in S} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda J(f)$$

Where

$$J(f) = \iint \left(\frac{\partial^2 g(x)}{\partial x_1^2}\right)^2 + \left(\frac{\partial^2 g(x)}{\partial x_1 x_2}\right)^2 + \left(\frac{\partial^2 g(x)}{\partial x_2^2}\right)^2 dx_1 dx_2$$

Thin-Plate Splines

- The solution has the form:

$$g(x) = \beta_0 + \beta^T x + \sum_{j=1}^{N} \alpha_j h_j(x)$$

Where

$$h_j(x) = \eta(|x - x_j|) \text{ and } \eta(z) = z^2 \log z^2$$