Multiple non-linear regression

Jiří Kléma

Department of Computer Science,
Czech Technical University in Prague

Lecture based on **ISLR book** and its accompanying slides

http://cw.felk.cvut.cz/wiki/courses/b4m36san/start

Agenda

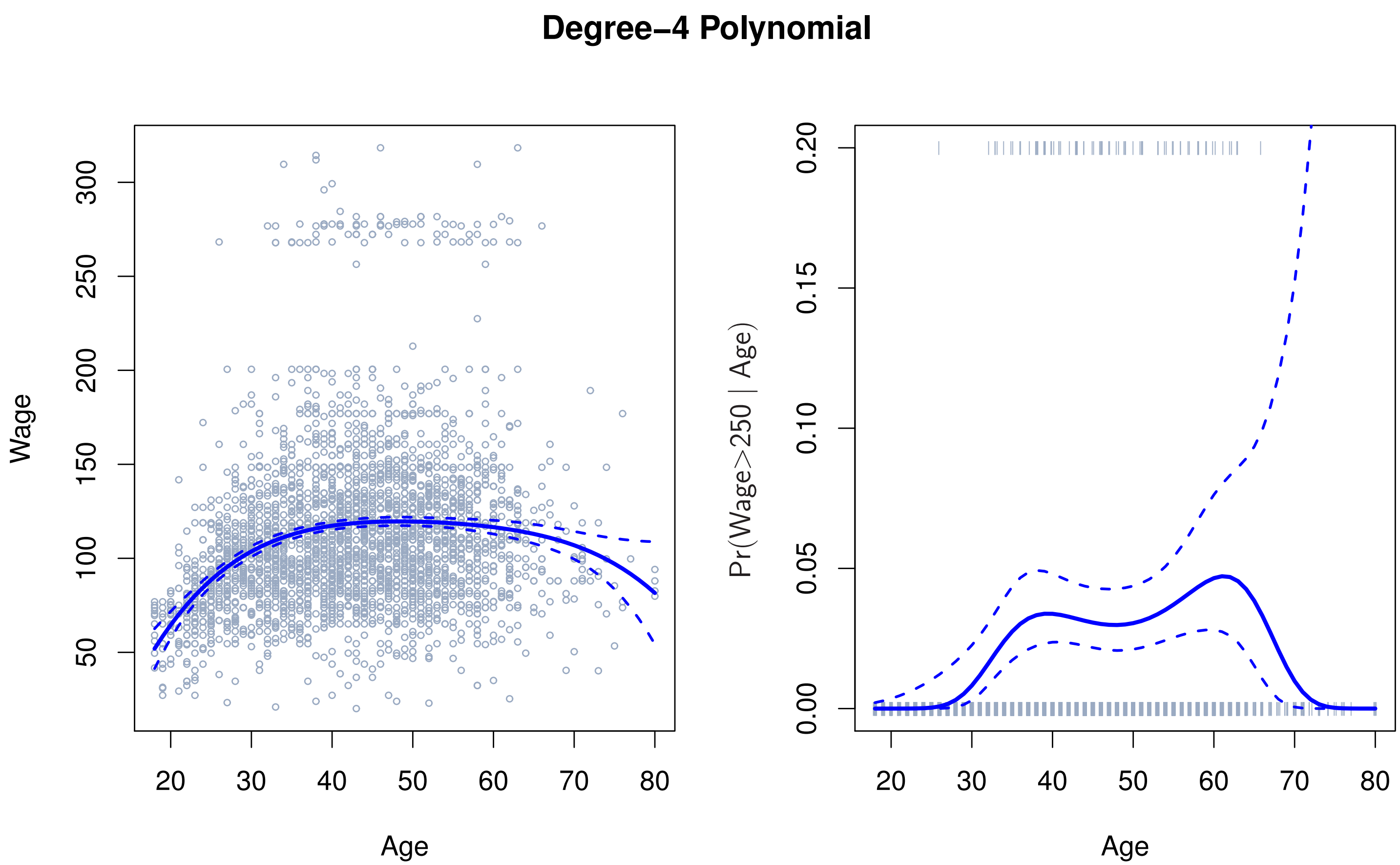
- Linear regression
  - a model with single predictor and its extension toward multiple linear regression,
  - parameters, interpretation, hypotheses testing,
  - special issues: qualitative predictors, outliers, collinearity,
- linear model selection and regularization
  - subset selection,
  - regularization = shrinkage, lasso, ridge regression,
  - choosing the optimal model, estimating test error,
- **moving beyond linearity**
  - basically via basis expansion,
  - polynomial regression, step functions,
  - splines, local regression,
  - generalized additive models.

Moving Beyond Linearity

- The truth is never linear! Or almost never!
- When the linearity assumption is not good enough . . .
  - polynomials
    - ∗ expansion up to the n-th degree polynomial,
  - step functions
    - ∗ cut the predictor into distinct regions, construct stepwise models,
  - splines
    - ∗ piecewise polynomials with constraints,
  - local regression
    - ∗ fit many local (typically linear) models along the range of the predictor.
  - generalized additive models
    - ∗ extension of **multiple** linear regression to non-linear elements.
- offer a lot of flexibility,
- without losing the ease and interpretability of linear models.

Polynomial regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \epsilon_i$$



Polynomial regression

- 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

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \cdots + + \hat{\beta}_d x_0^d$$

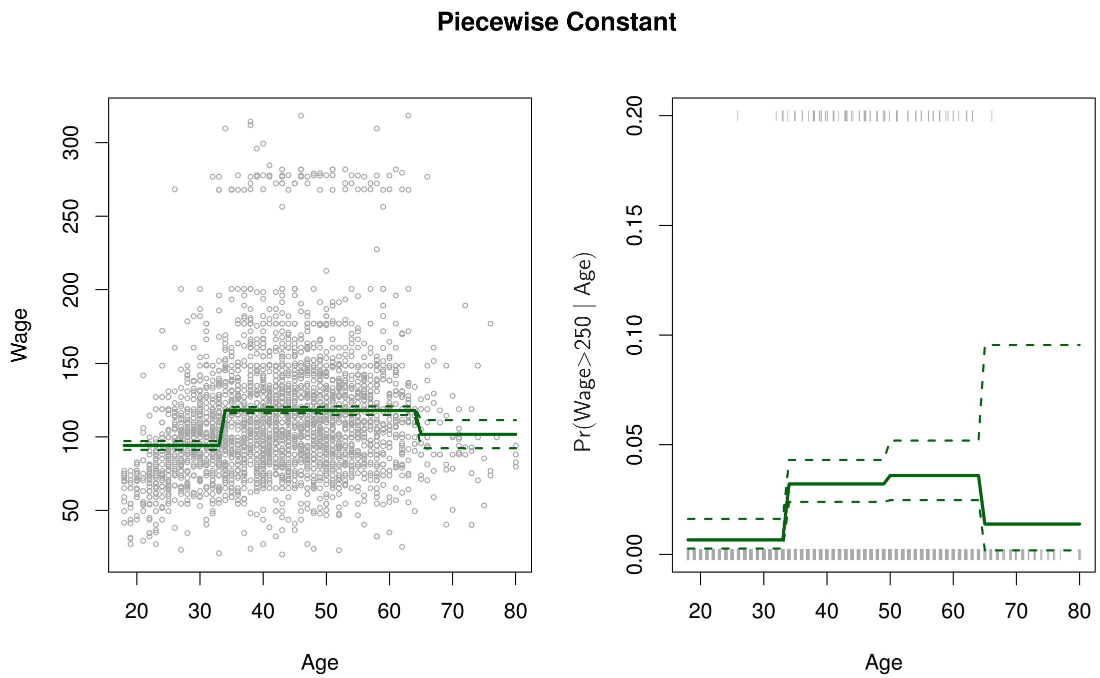
- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_\ell$
  - pointwise-variances $Var[\hat{f}(x_0)]$ at any value x_0 can be estimated,
  - in the previous figure, $\hat{f}(x_0) \pm 2se[\hat{f}(x_0)]$ is shown,
- we either fix the degree d at some reasonably low value, else use cross-validation to choose d.

Step functions

- cut the independent variable into distinct regions, construct stepwise models,
- example of dummy variables:

  $C_0(X) = I(X < 35),\ C_1(X) = I(35 \leq X < 50),\ \dots,\ C_K(X) = I(X \geq 35)$
- linear model with the dummy variables as predictors

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



Step functions

- Easy to work with, creates a series of dummy variables representing groups,
- useful way of creating interactions that are easy to interpret,
- for example, interaction effect of Year and Age:

$$I(Year < 2005) \cdot Age,\ I(Year \geq 2005) \cdot Age$$

- 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.

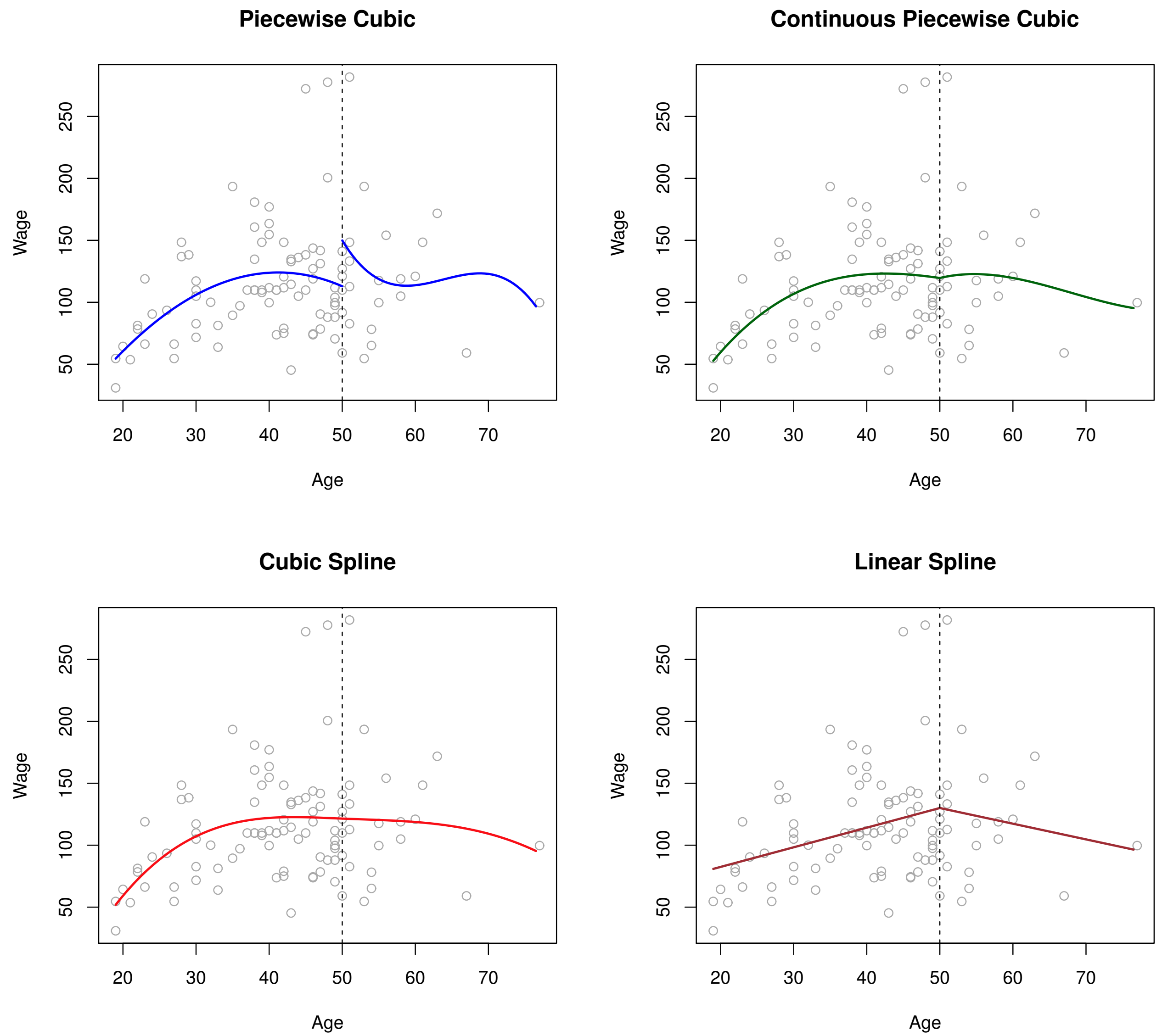
Piecewise polynomials

- Instead of a single polynomial in X over its whole domain,
- we can rather use different polynomials in regions defined by **knots**
  - knots = the points where the models/coefficients change,
- an example of a piecewise cubic polynomial (without constraints)

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

- better to add constraints to the polynomials, e.g. continuity,
- **splines** have the "maximum" amount of continuity
  - degree-d spline is a piecewise degree-d polynomial with the constraints of continuity as well as continuity in derivatives up to degree d-1 at each knot.

Piecewise polynomials



Take A Break

Difficult predictive tasks and risk of overfitting . . .

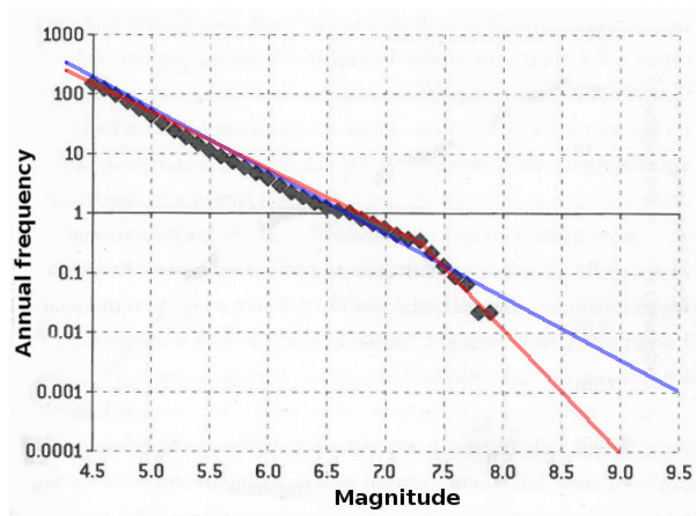
- Guess the most difficult predictive task

  1. where and when a major earthquake strikes,
  2. tornado warnings for timely evacuation,
  3. presidential elections.

Difficult predictive tasks and risk of overfitting ...

- Guess the most difficult predictive task
  1. where and when a major earthquake strikes,
  2. tornado warnings for timely evacuation,
  3. presidential elections.

- Fukushima 2011 nuclear disaster [Silver: The signal and the noise]



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

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

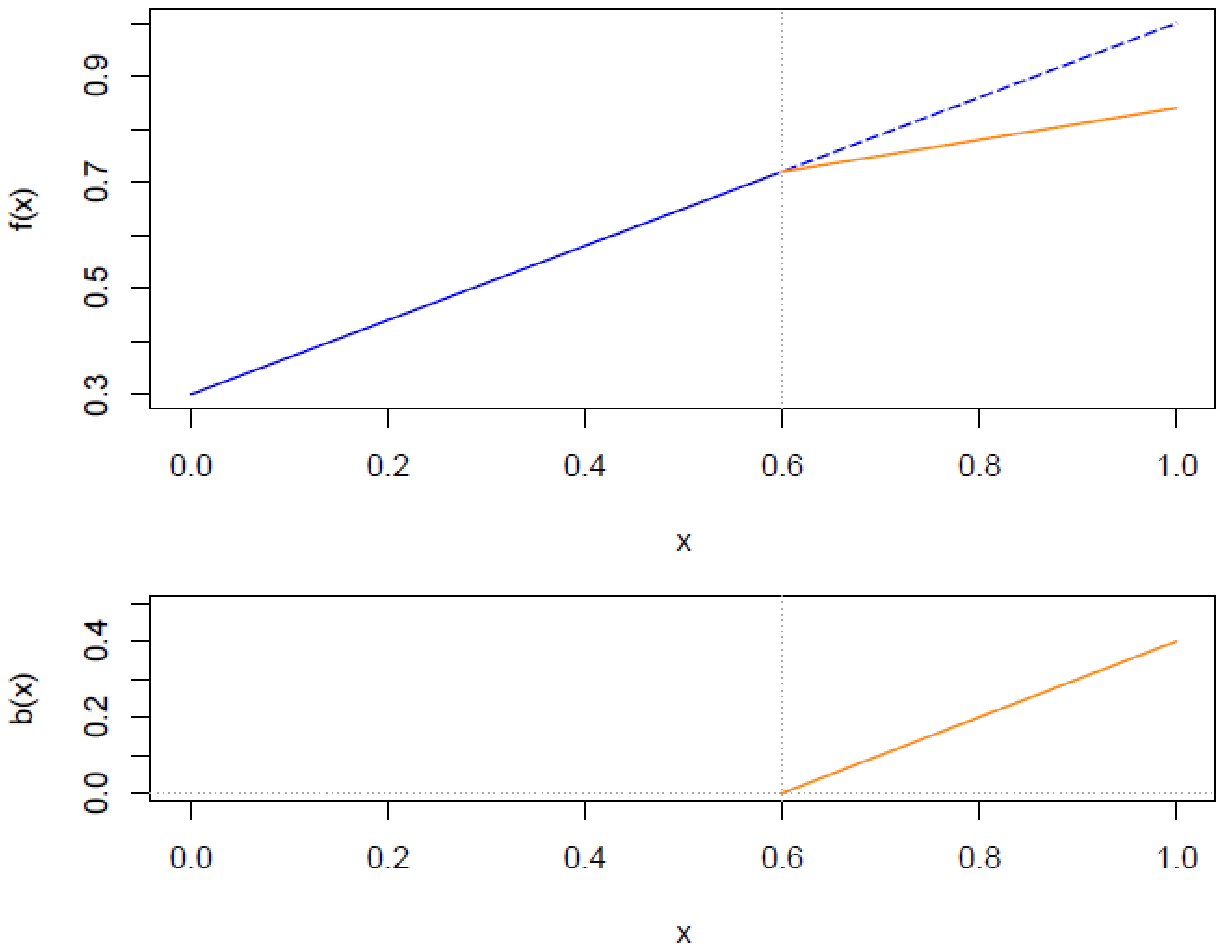
- where the b_k are **basis functions**

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

- where the $()_+$ means positive part, i.e.

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

Linear 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,
- we can represent this model with truncated power basis functions

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

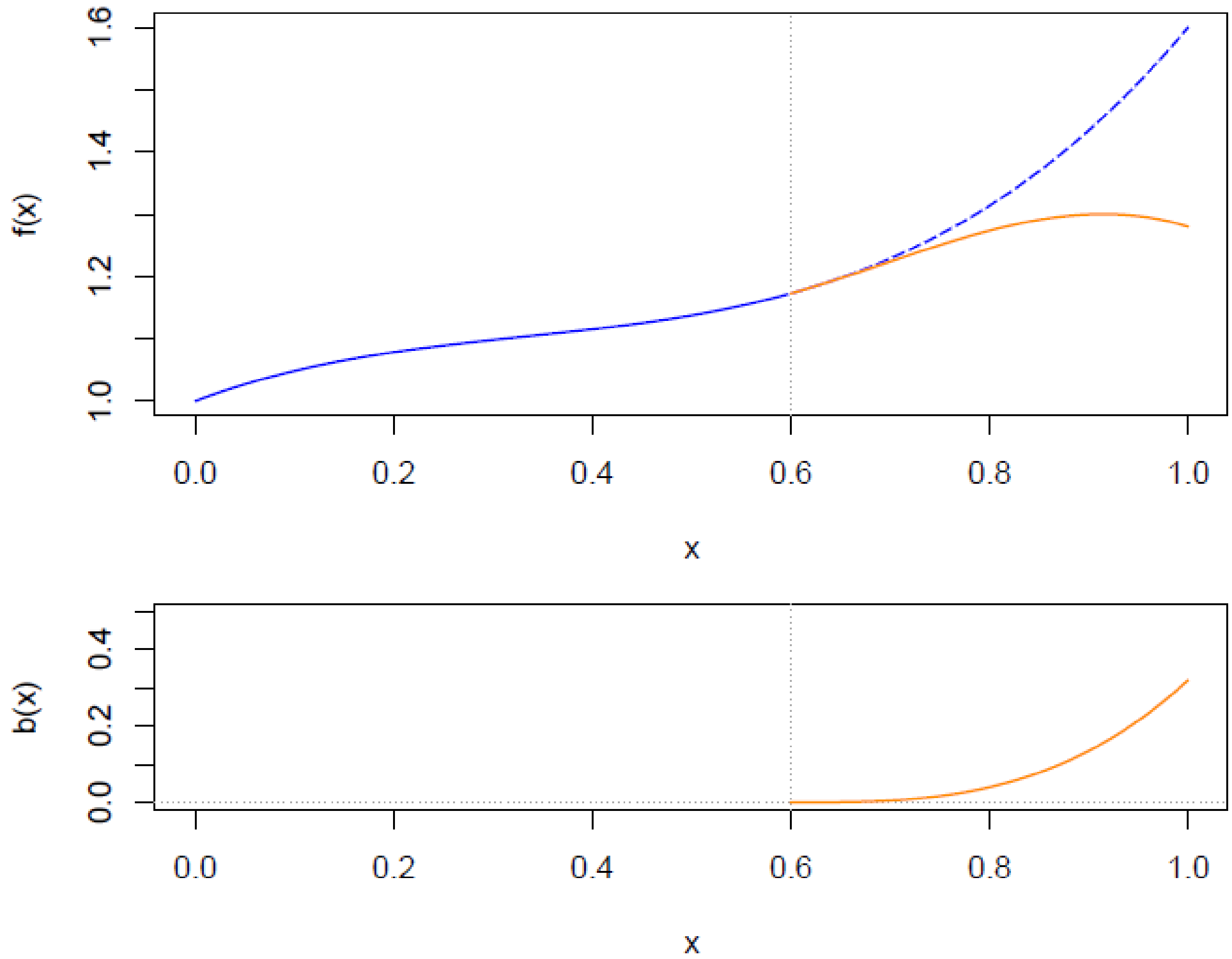
- where the b_k are **basis functions**

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

- where

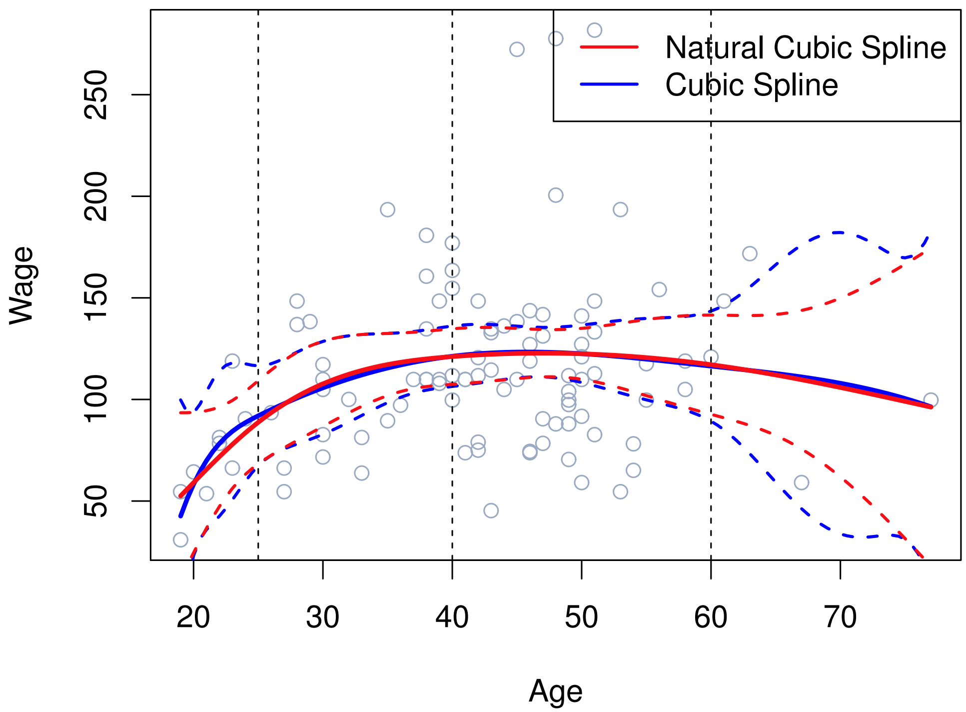
$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$

Cubic splines



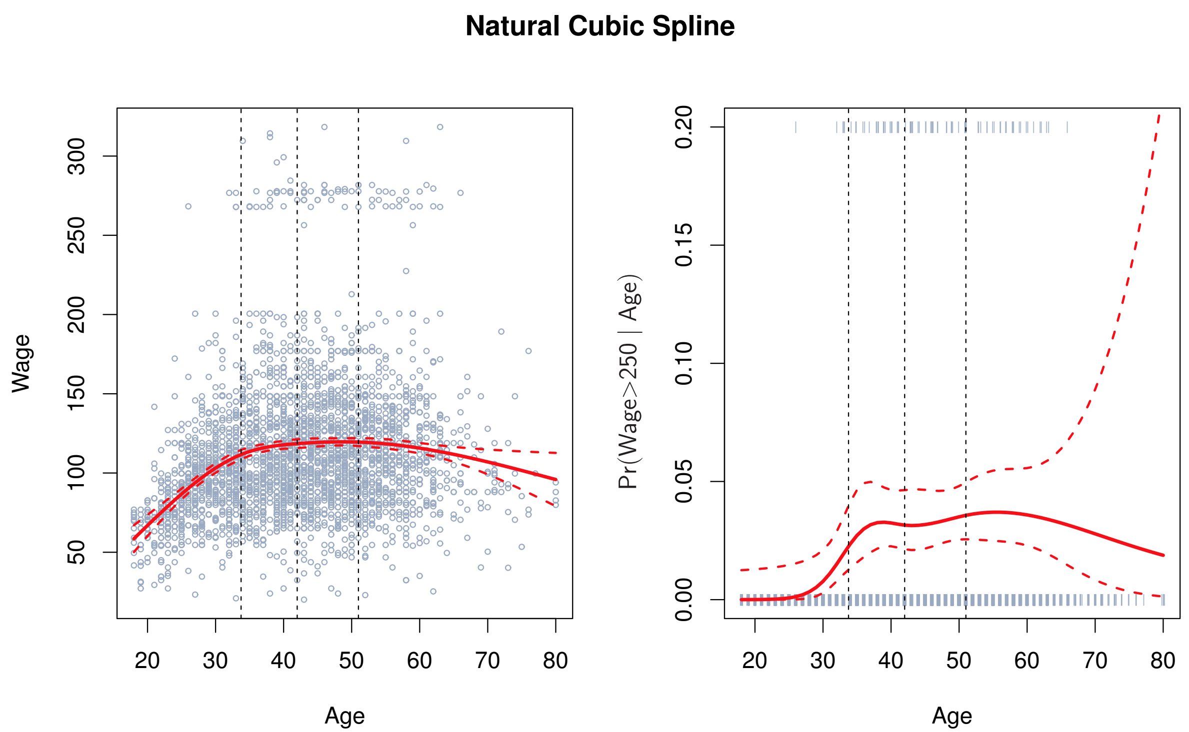
Natural cubic splines

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



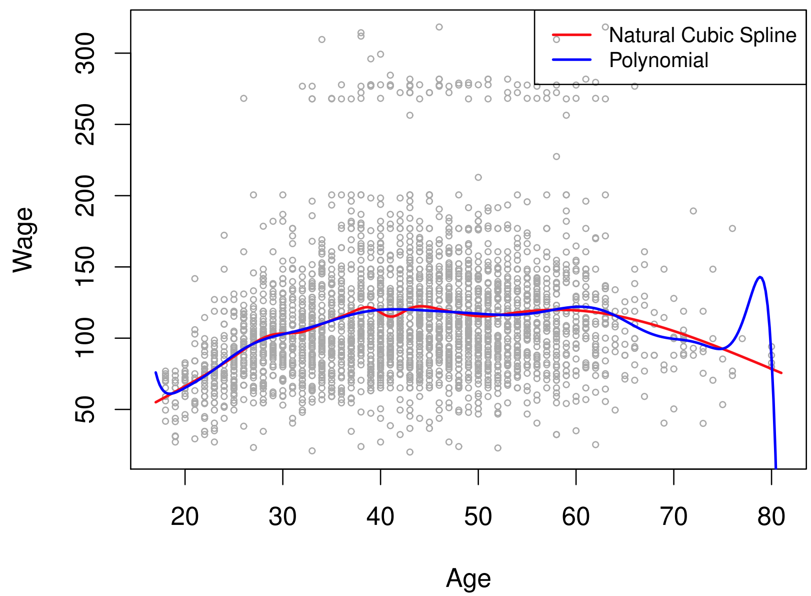
Natural cubic splines

- Fitting splines in R with the package `splines`
  - `bs(x, ...)` for any degree splines,
  - `ns(x, ...)` for natural cubic 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,
- below comparison of a degree-14 polynomial (`poly(age, deg=14)`) and a natural cubic spline (`ns(age, df=14)`), each with 15df.



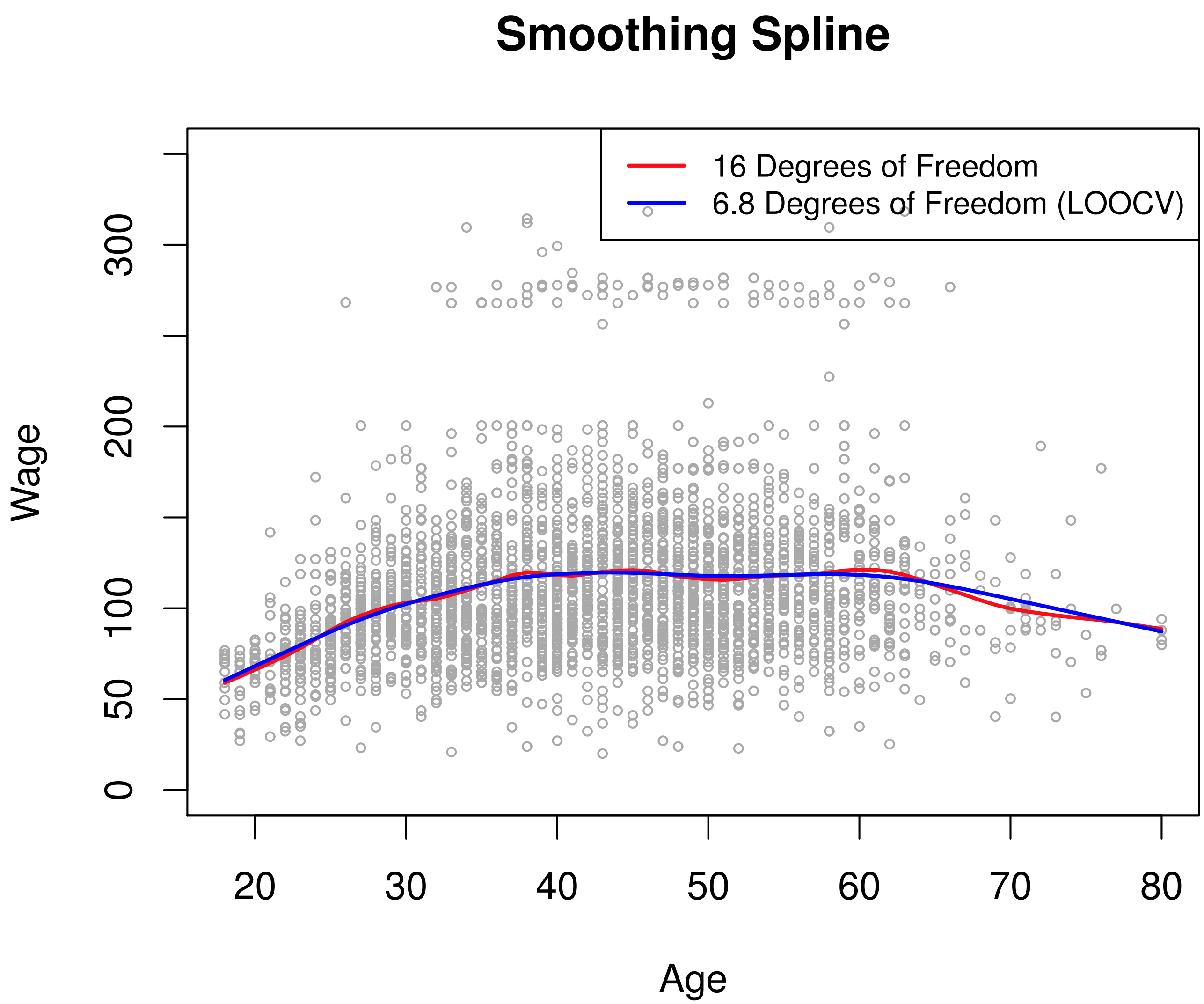
Smoothing splines

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

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

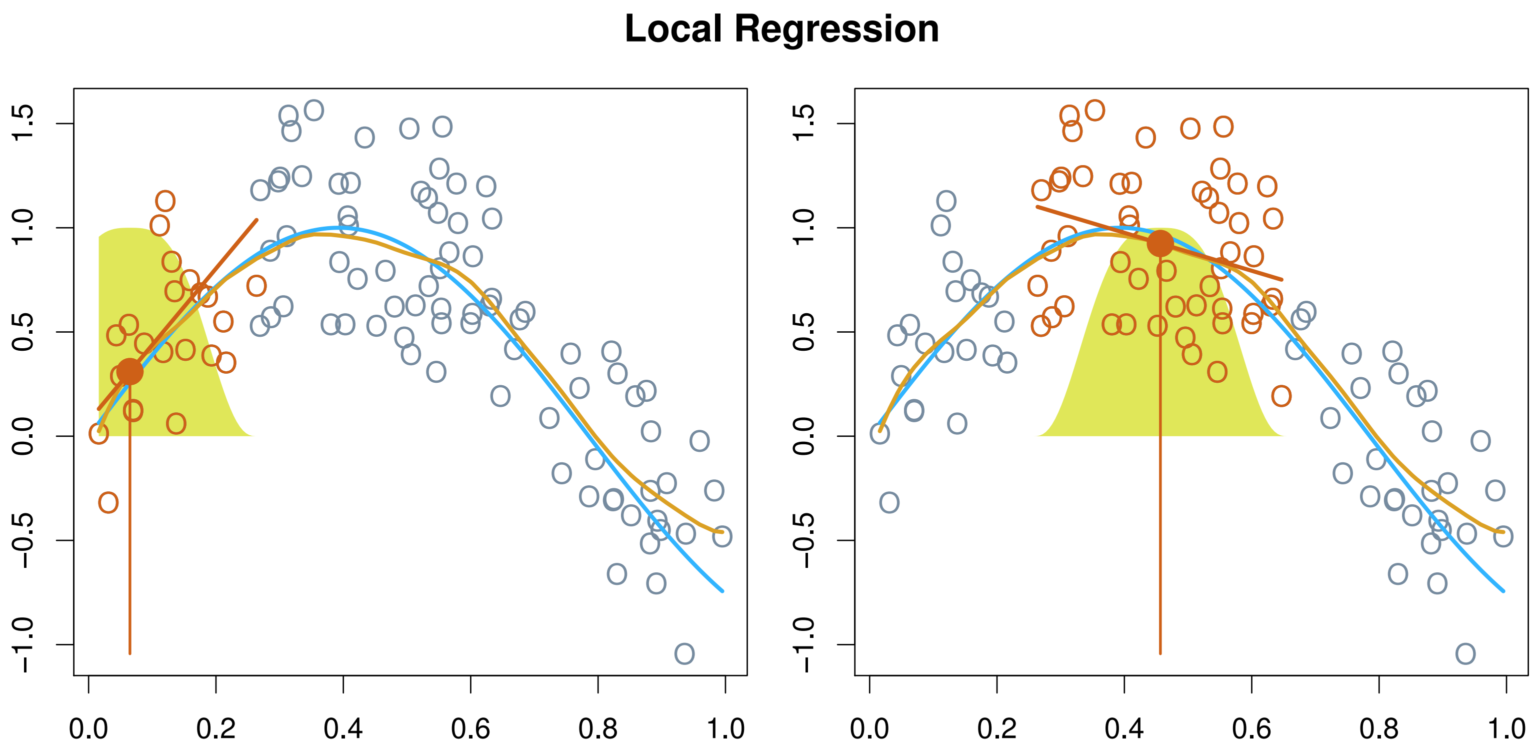
  - the first term is RSS, and tries to make $g(x)$ match the data at each x_i,
  - the second term is a roughness penalty and controls how wiggly $g(x)$ is,
  - roughness is modulated by the tuning parameter $\lambda \geq 0$
    - the smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$,
    - As $\lambda \to \infty$, the function $g(x)$ becomes linear,
- the solution is a natural cubic spline, with a knot at every unique value of x_i
  - however, smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen,
  - in R, the function `smooth.spline()` fits a smoothing spline,
  - this function enables to specify the number of effective dfs instead of λ.

Smoothing splines



Local regression

- Build separate linear fits over the range of X by weighted least squares,
- employ a sliding weight function (kernel), in R call `loess()`.

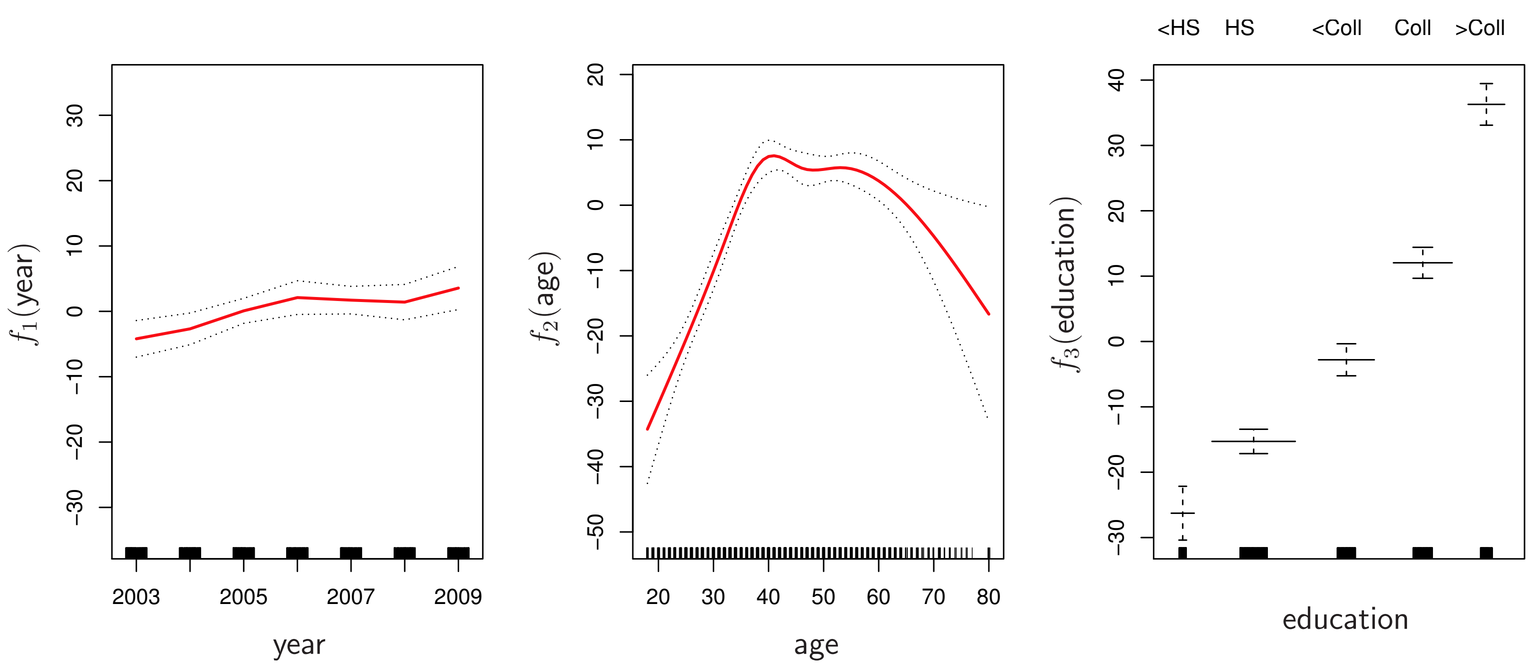


Local regression, simulated data
Blue – the model used to generate the data; orange – the local regression estimate of the model, red – a local fit

Generalized additive models

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

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



Generalized additive models

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

  `lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + 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

  `gam(wage ~ s(year, df = 5) + lo(age, span = .5) + education)`

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

The main references

:: Resources (slides, scripts, tasks) and reading

- G. James, D. Witten, T. Hastie and R. Tibshirani: **An Introduction to Statistical Learning with Applications in R.** Springer, 2014.
- K. Markham: **In-depth Introduction to Machine Learning in 15 hours of Expert Videos**. Available at R-bloggers.