LECTURE 16:
BEYOND LINEARITY PT. 2

November 8, 2017

SDS 293: Machine Learning

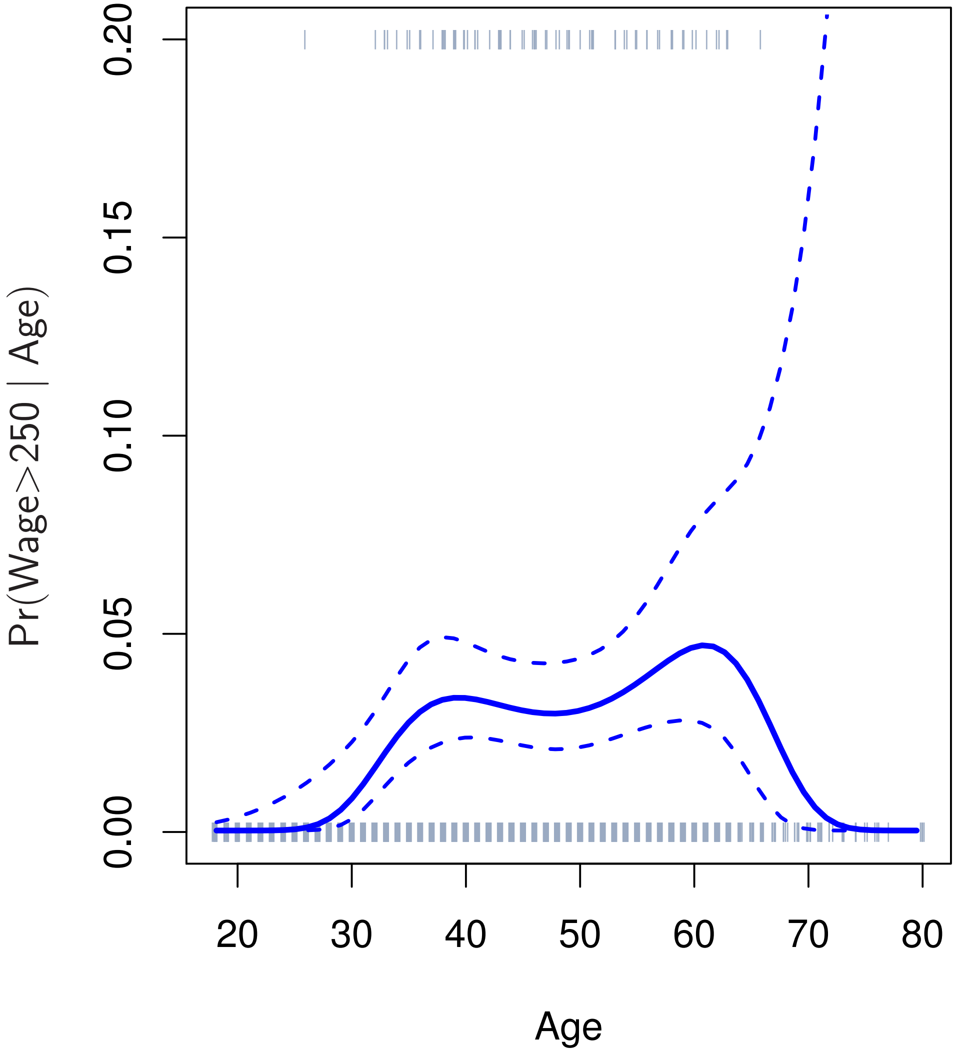
Outline

- Moving beyond linearity
  - ✓ Polynomial regression
  - ✓ Step functions
  - Splines
  - ~~Local regression~~
  - Generalized additive models (GAMs)
- Lab

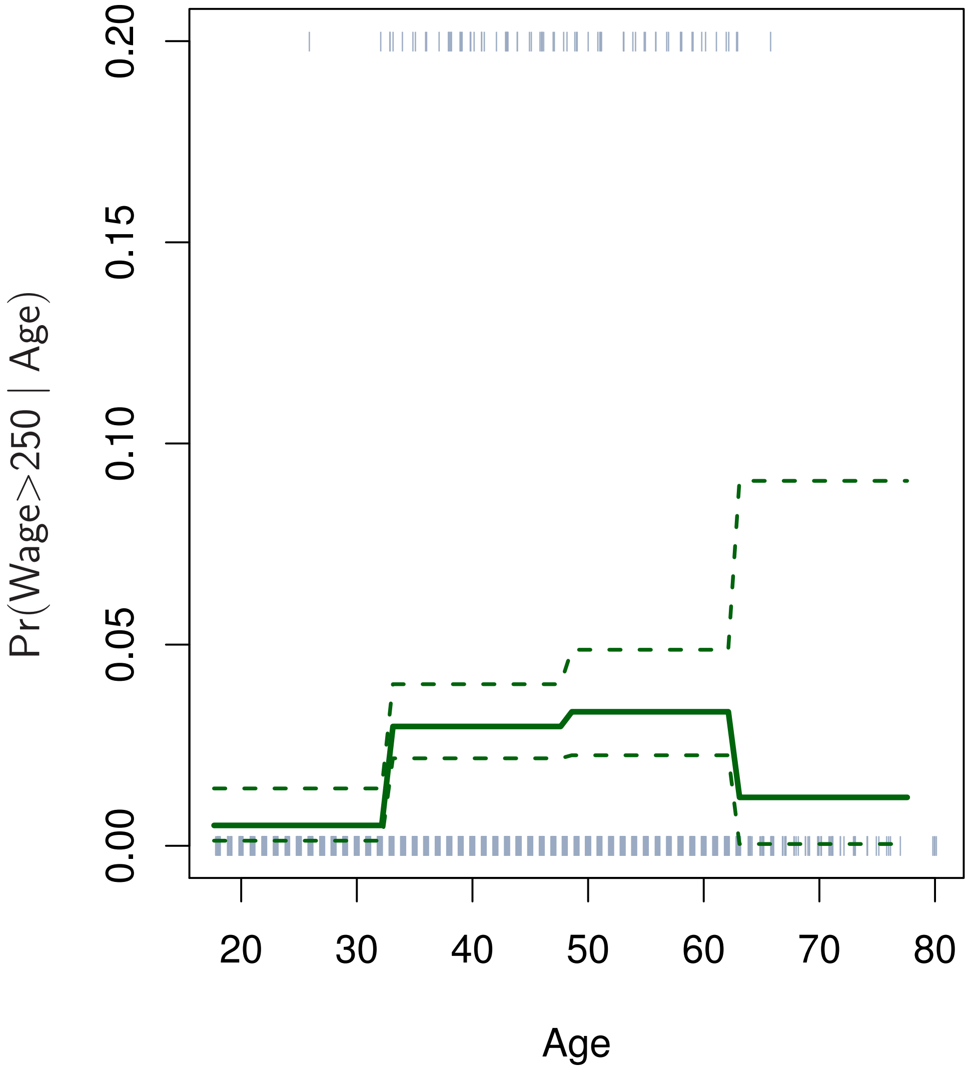
Recap: polynomial regression

Big idea:

extend the linear model by adding extra predictors that are **powers of** X



Recap: step functions



Big idea:

break X into pieces, fit a separate model on each piece, and **glue them together**

Discussion

- **Question**: what do these approaches have in common?
- **Answer**: they both apply some set of **transformations** to the predictors. These transformations are known more generally as **basis functions**:
  - Polynomial regression: $b_j(x_i) = x_i^j$
  - Step functions: $b_j(x_i) = I\left(c_j \leq x_i < c_{j+1}\right)$

Lots of other functions we could try as well!

Piecewise polynomials

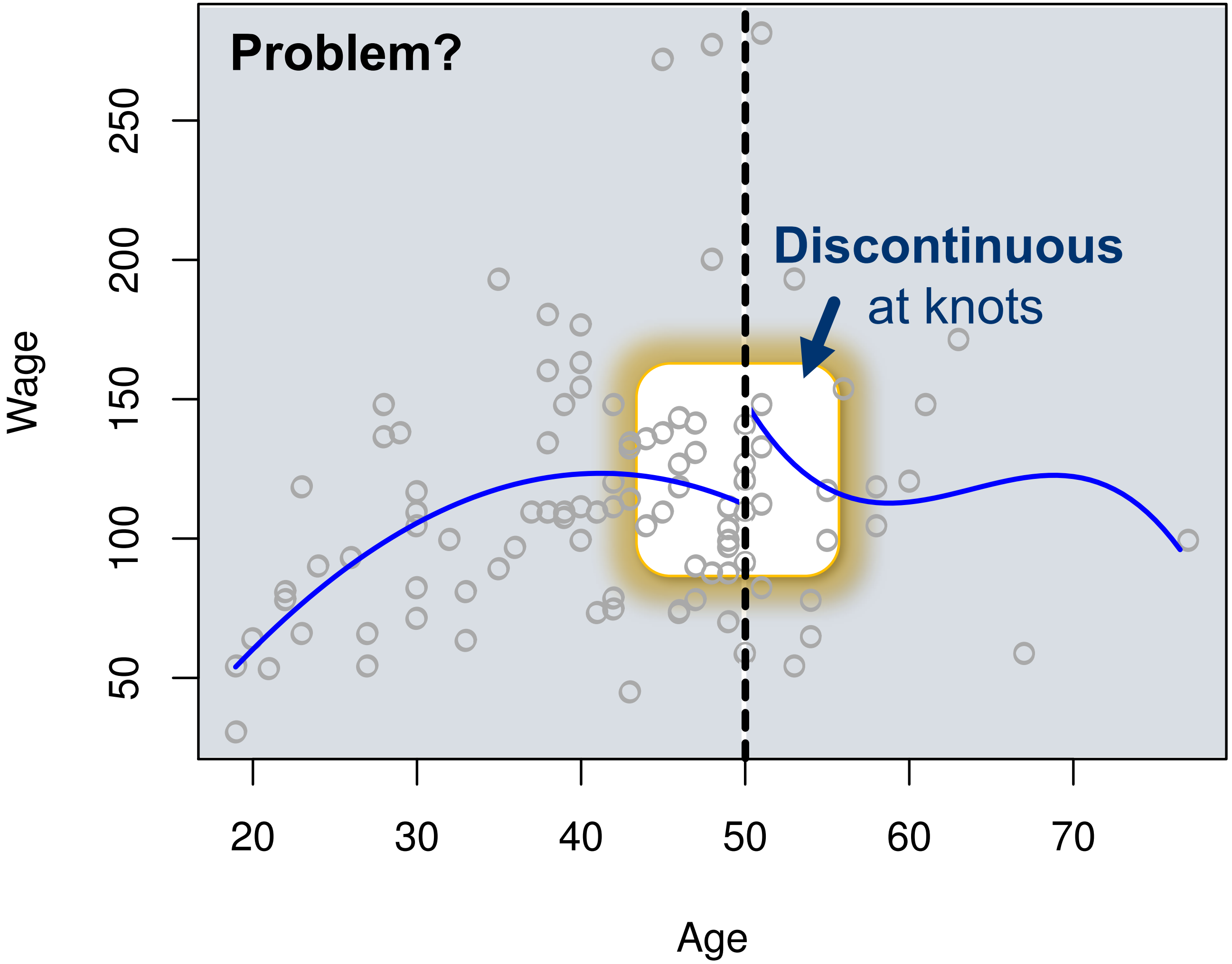
- What if we **combine** polynomials and step functions?
- Ex:

$$y_i = \beta_0 + \underbrace{\boxed{\beta_1} x_i + \boxed{\beta_2} x_i^2 + \boxed{\beta_3}}_{\text{may take different values in different parts of } X} x_i^3 + \varepsilon_i$$

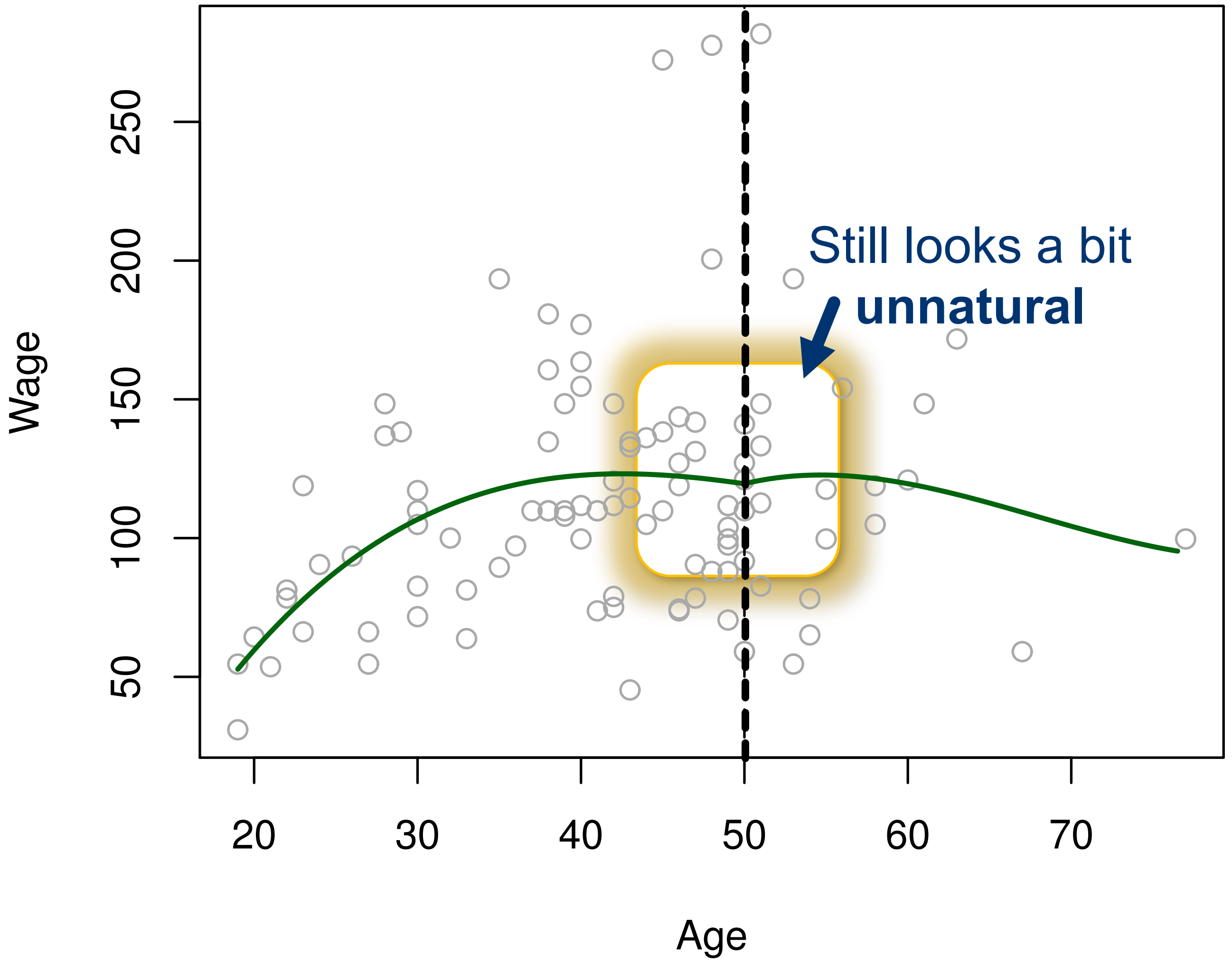
Points where coefficients change = “**knots**”

$$y_i = \left\{\begin{array}{l} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \varepsilon_i \ \text{ if} \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \varepsilon_i \ \text{ if} \end{array}\right. \boxed{\begin{array}{l} x_i < c \\ x_i \geq c \end{array}}$$

Ex: `Wage` data subset



One way to fix it: require continuity



Still looks a bit **unnatural**

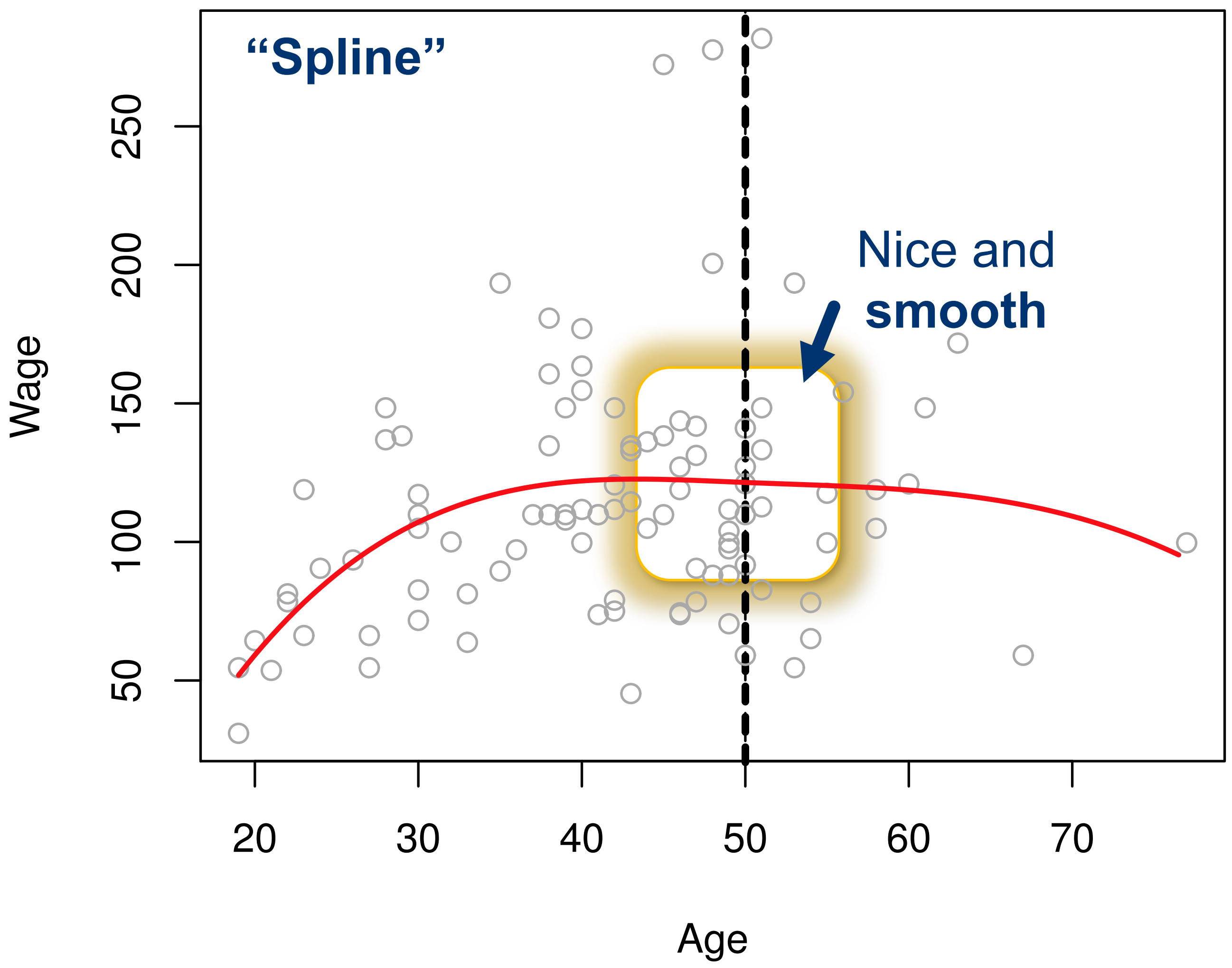
Degrees of freedom vs. constraints

- In our piecewise cubic function with one knot, we had **8 degrees of freedom**:

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

- We can add *constraints* to **remove degrees of freedom**:
  1. Function must be continuous
  2. Function must have continuous 1st derivative (slope)
  3. Function must have continuous 2nd derivative (curvature)

Better way: constrain function & derivatives



Nice and **smooth**

Regression splines

- **Question**: how do we we fit a piecewise degree-d polynomial while requiring that it (and possibly its first $d-1$ derivatives) be **continuous**?
- **Answer**: use the **basis model** we talked about previously

Fitting regression splines

- Let's say we want a **cubic spline*** **with K knots:**

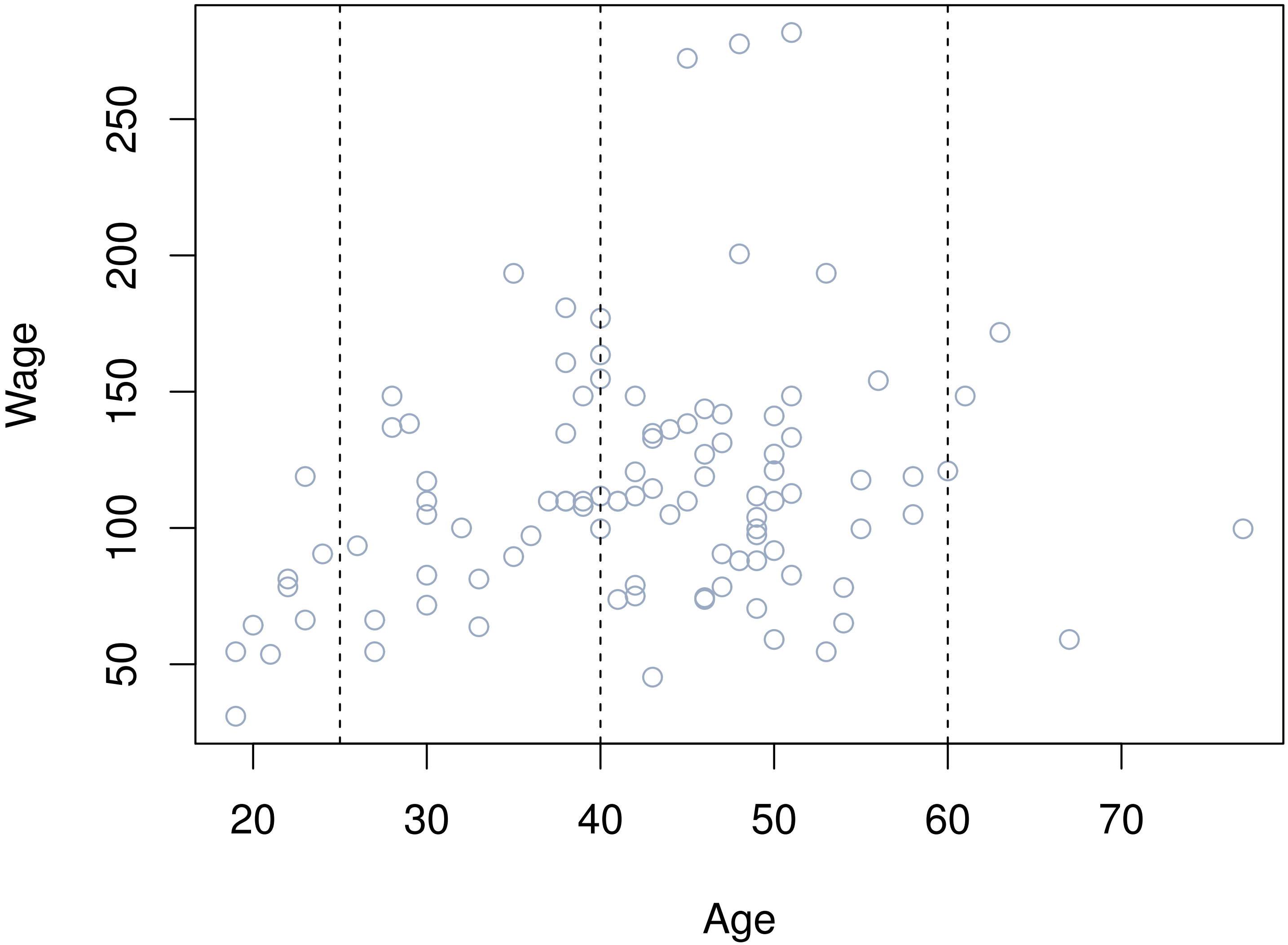
$$y_i = \beta_0 + \beta_1 \underbrace{\boxed{b_1}(x_i) + \beta_2 \boxed{b_2}(x_i) + \cdots + \beta_{K+3} \boxed{b_{K+3}}}_{\text{just need to choose appropriate basis functions}}(x_i) + \varepsilon_i$$

- One common approach is to start with a standard basis for a cubic polynomial (x, x^2, x^3) and then add one **truncated power basis** function per knot ξ:

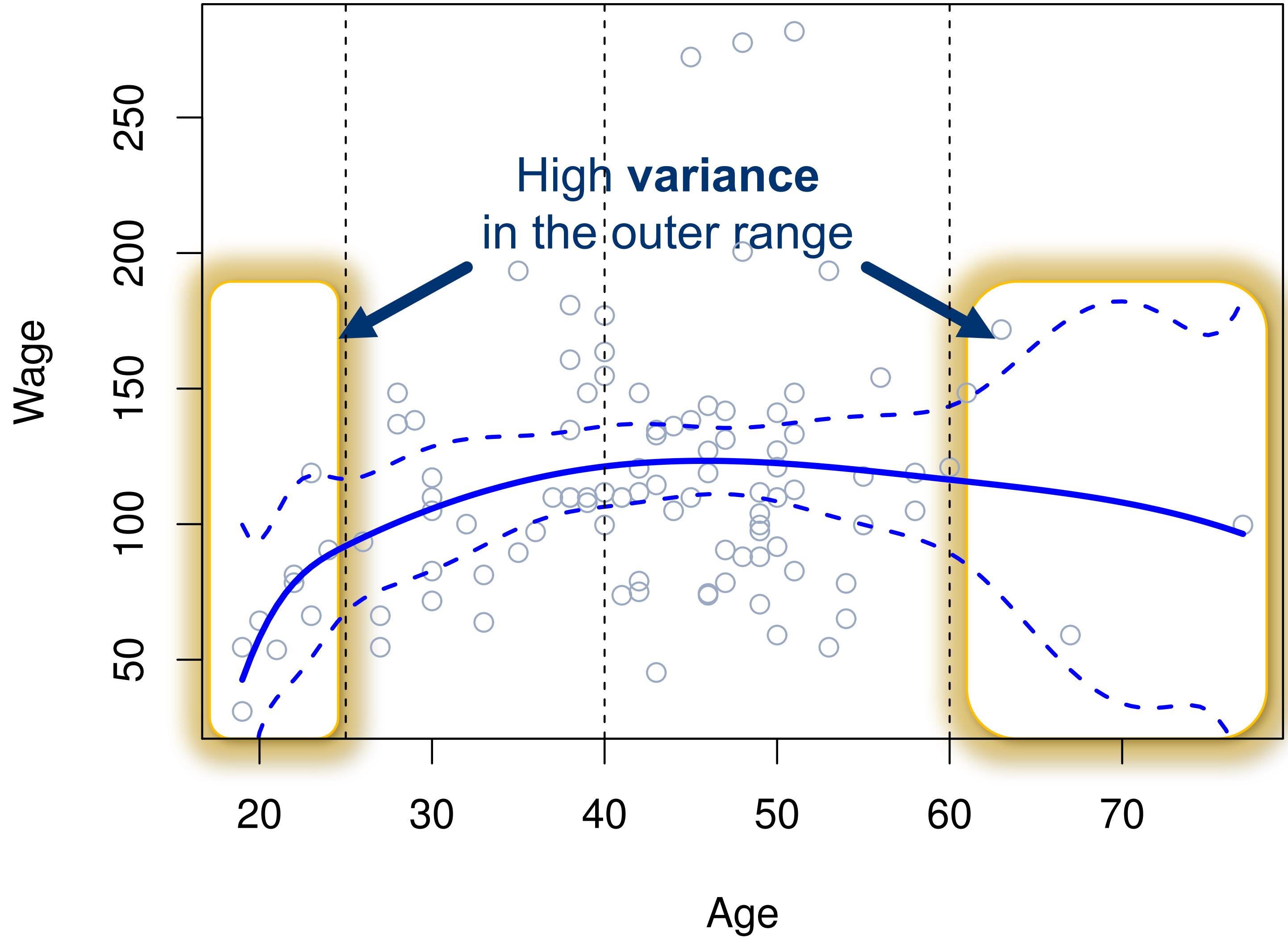
$$h(x, \xi) = \begin{cases} (x - \xi)^3 \text{ if } x > \xi \\ 0 \text{ otherwise} \end{cases}$$

*Cubic splines are popular because most human eyes cannot detect the discontinuity at the knots

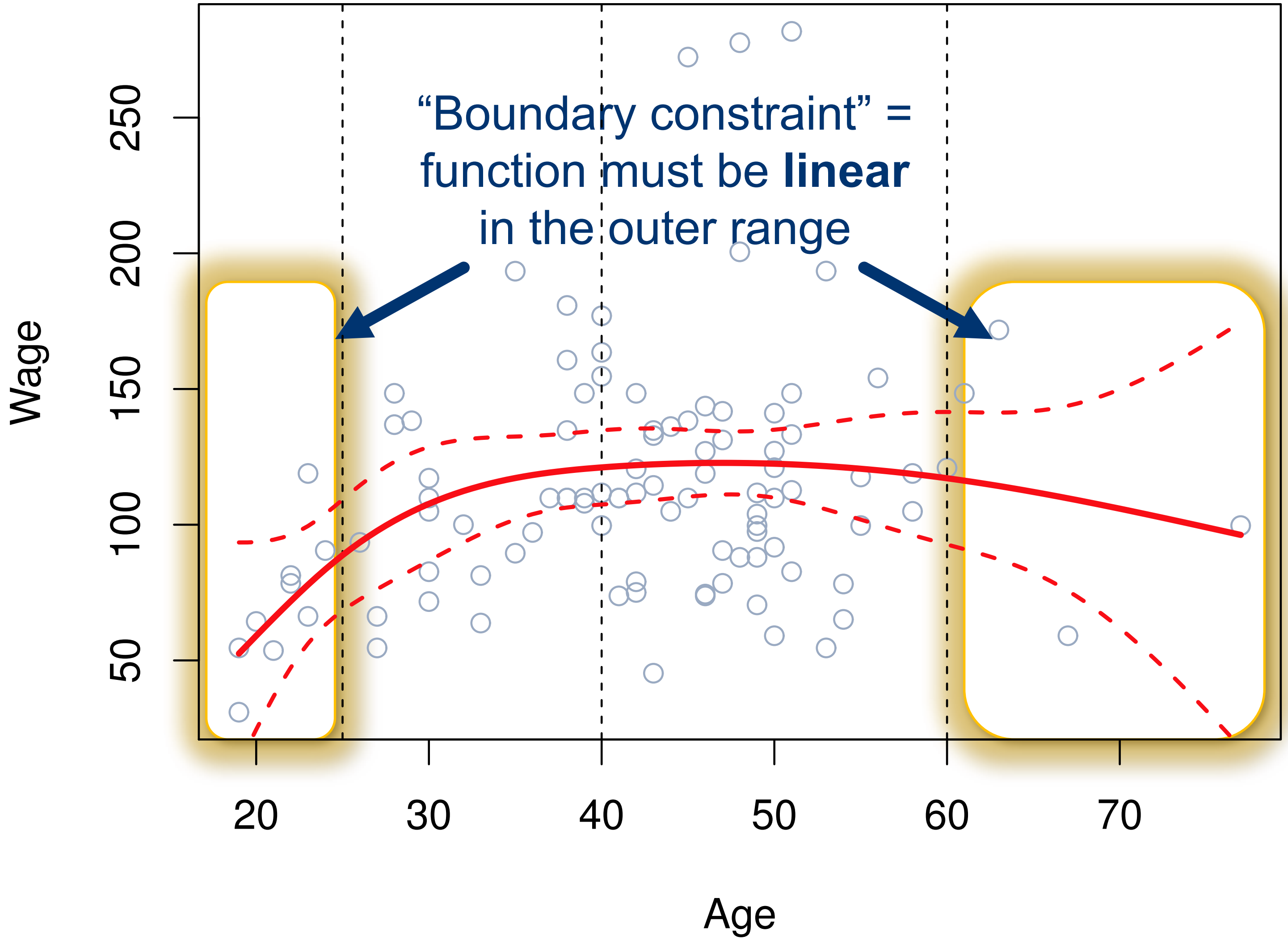
Ex: Wage data, 3 knots



Ex: Wage data, 3 knots, cubic spline



Ex: `Wage` data, 3 knots, **natural** spline

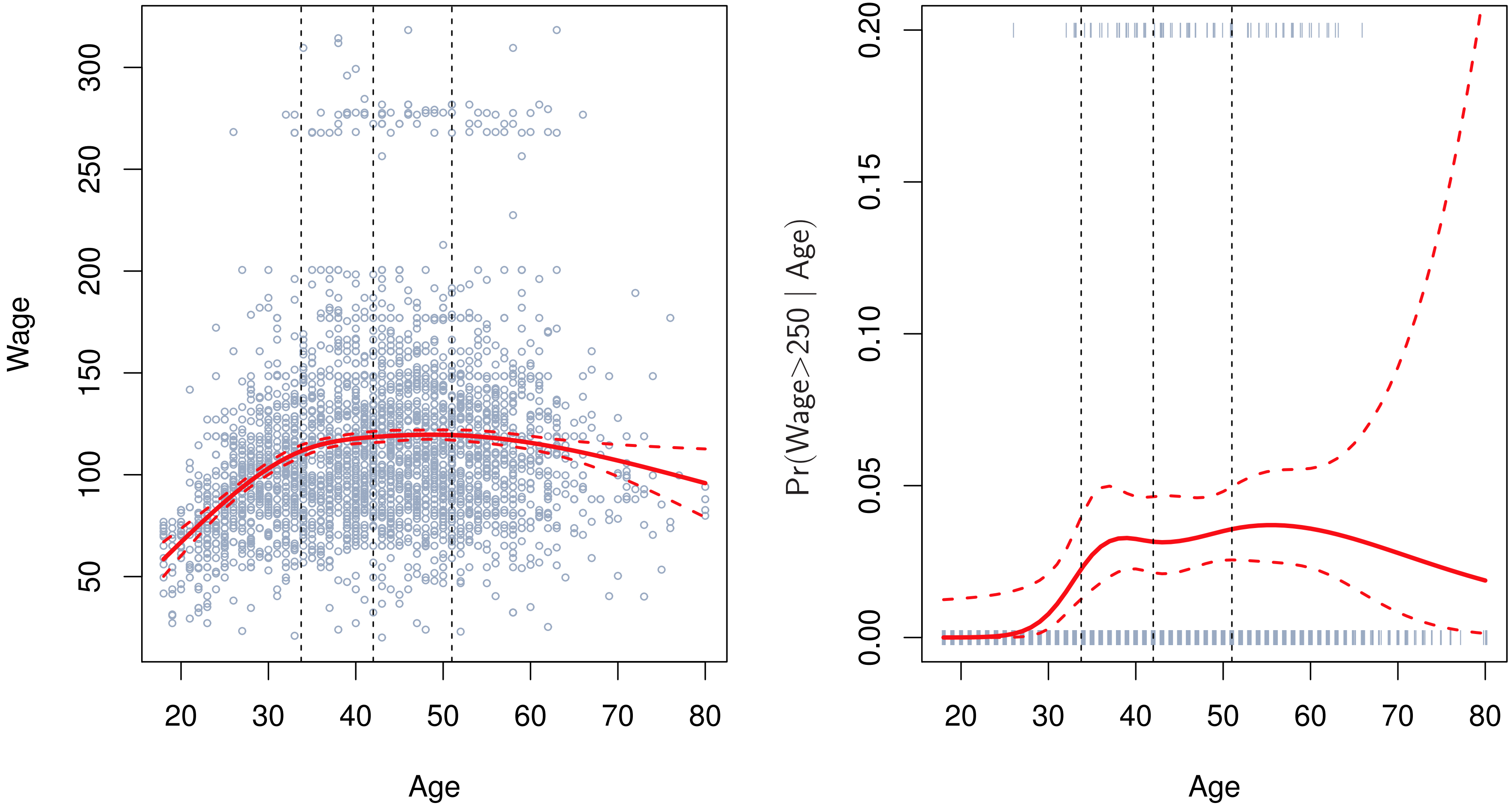


“Boundary constraint” = function must be **linear** in the outer range

Regression splines

- **Question**: how do we figure out how many **knots** to use, and **where to put them**?
- **Answer**: the methods we used to determine the best number of predictors can help us figure out how many knots to use. For placement, we have several options:
  - Place them **uniformly** across the domain
  - Put more knots in places where the data **varies a lot**
  - Place them at **percentiles** of interest (e.g. 25th, 50th, and 75th)

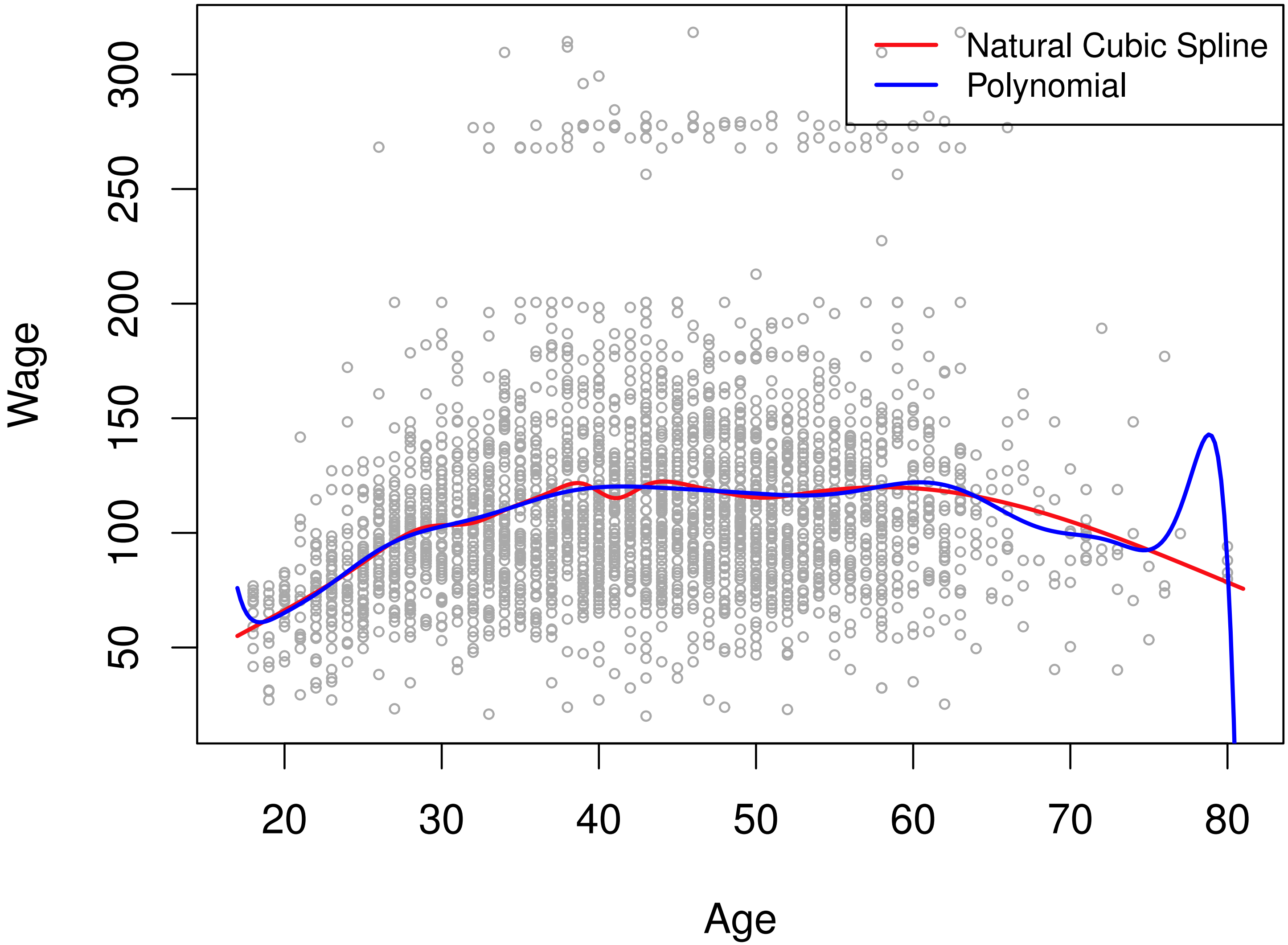
Ex: `wage` data, 3 knots at 25th, 50th, & 75th



Comparison with polynomial regression

- **Question**: how would you expect this to compare to **polynomial regression**?
- **Answer**: regression splines often give better results than polynomial regression because they can add flexibility in places where it is needed by **adding more knots**, without having to **add more predictors**

Ex: `Wage` data, polynomial vs. spline



Discussion

- **Regression splines**: specify knots, find good basis functions, and use least squares to estimate coefficients
- **Goal:** find a function $g(x)$ that fits the data well, i.e.

$$RSS = \sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2$$

is **small**

- **Question:** what’s a trivial way to minimize RSS?
- **Answer:** interpolate over all the data (overfit to the max!)

Smoothing splines

- **Goal:** find a g that makes RSS small but that is also **smooth**
- Dust off your calculus* and we can find g that minimizes:

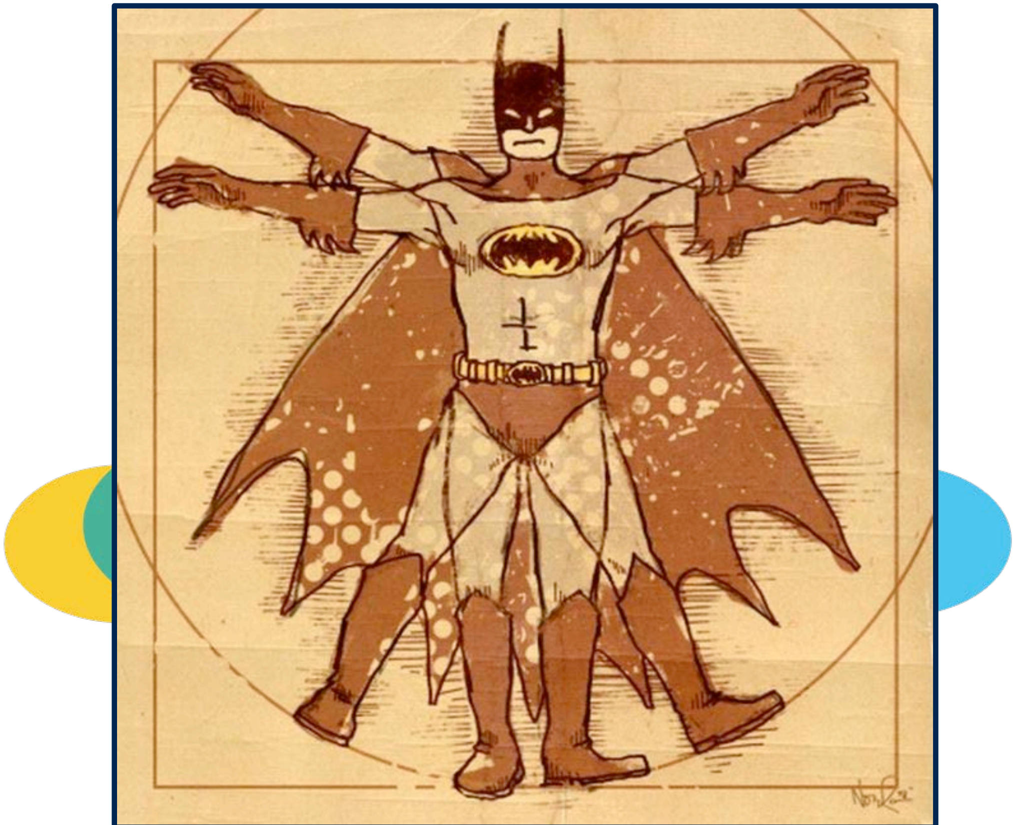
$$RSS = \underbrace{\sum_{i=1}^{n}\left(y_i - g(x_i)\right)^2}_{\text{“make sure you } \textbf{fit the data}\text{”}} + \underbrace{\lambda \int g''(t)^2 dt}_{\text{“make sure you’re } \textbf{smooth}\text{”}}$$

- **Fun fact**: this is minimized by a shrunken version of the natural cubic spline with knots at **each training observation**

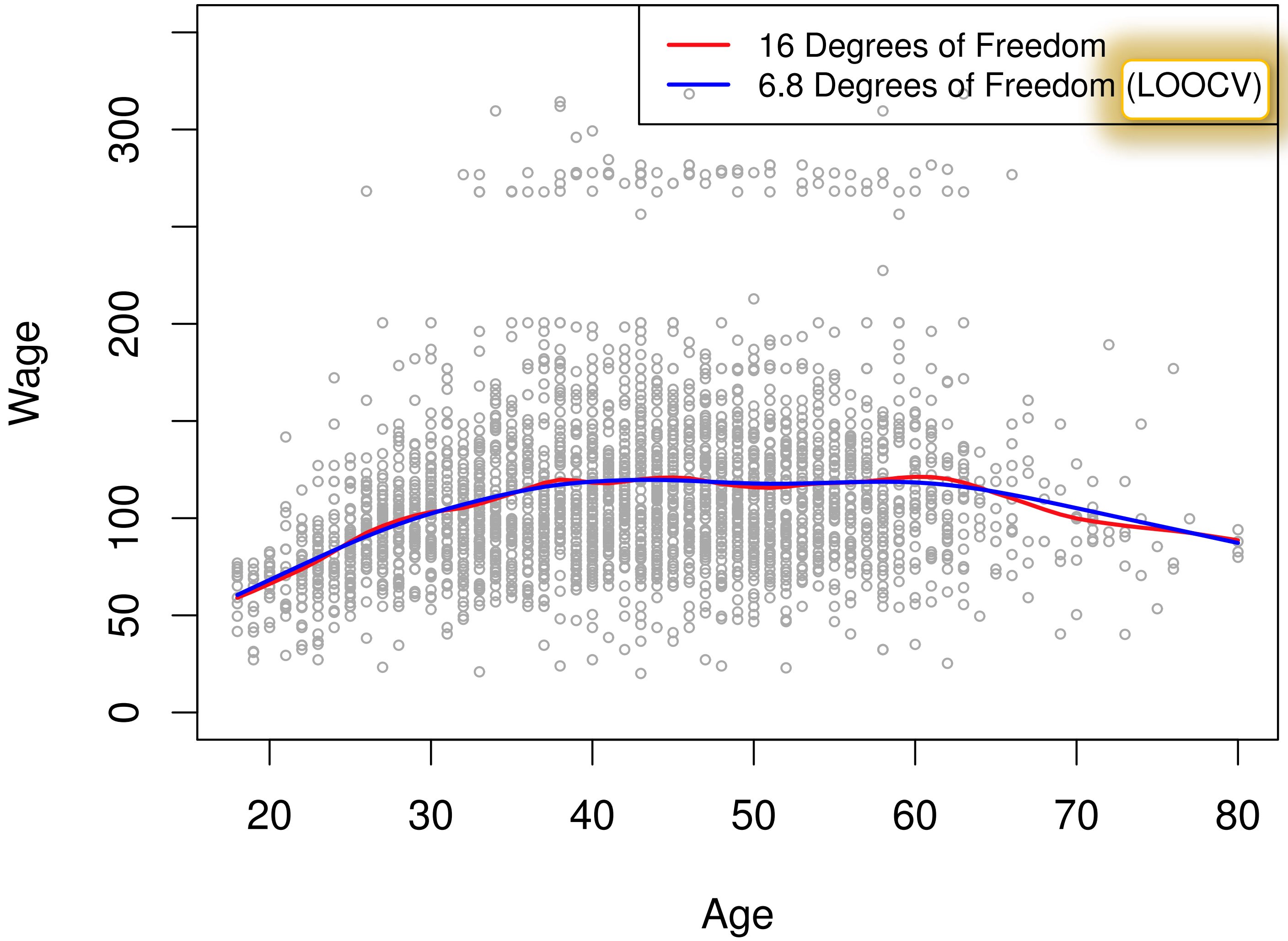
*The second derivative of a function is a measure of its **roughness**: it is large in absolute value if $g(t)$ is very wiggly near t, and close to zero otherwise

Whoa… knots at **every** training point?

- **Question**: shouldn’t this give us way too much flexibility?
- **Answer**: the key is in the shrinkage parameter λ, which influences our **effective** degrees of freedom

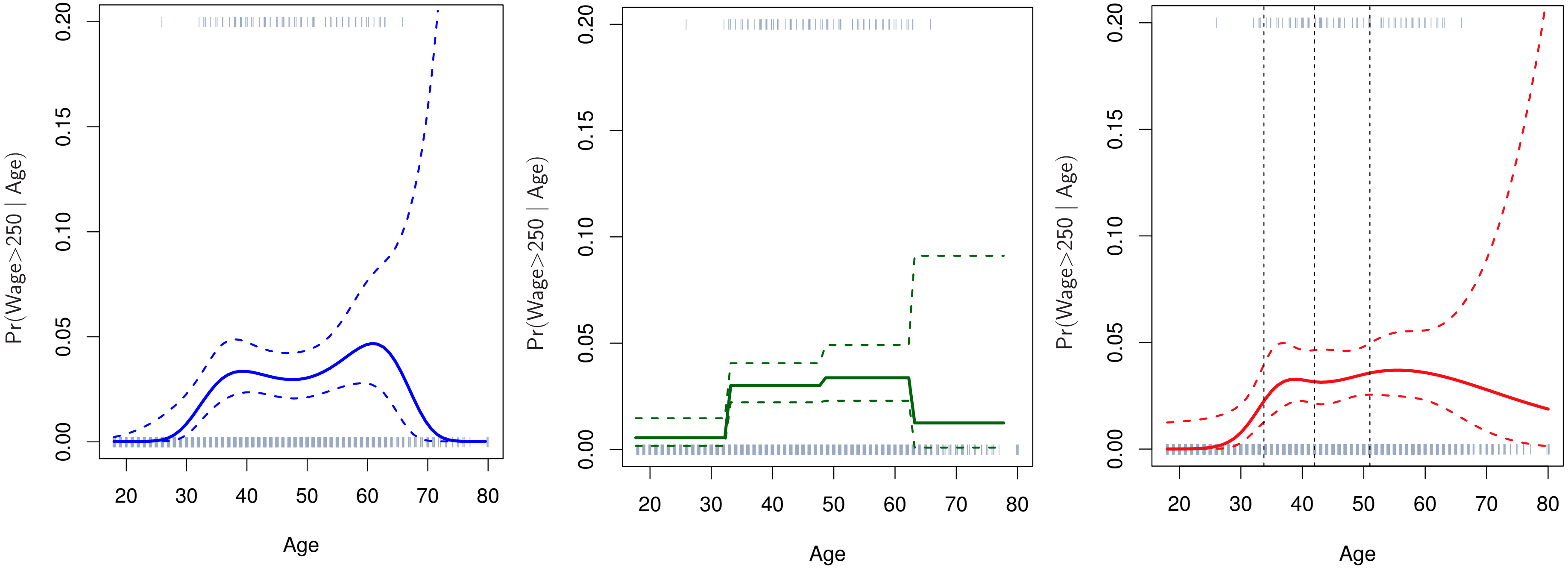


Ex: Wage data, smoothing splines w/ different λ



Recap

- **So far**: flexibly predict Y on the basis of one predictor X



= extensions of simple linear regression

- **Question**: what seems like the next logical step?

Generalized additive models (GAMs)

- **Big idea**: extend these methods to multiple predictors and non-linear functions of those predictors just like we did in with linear models before

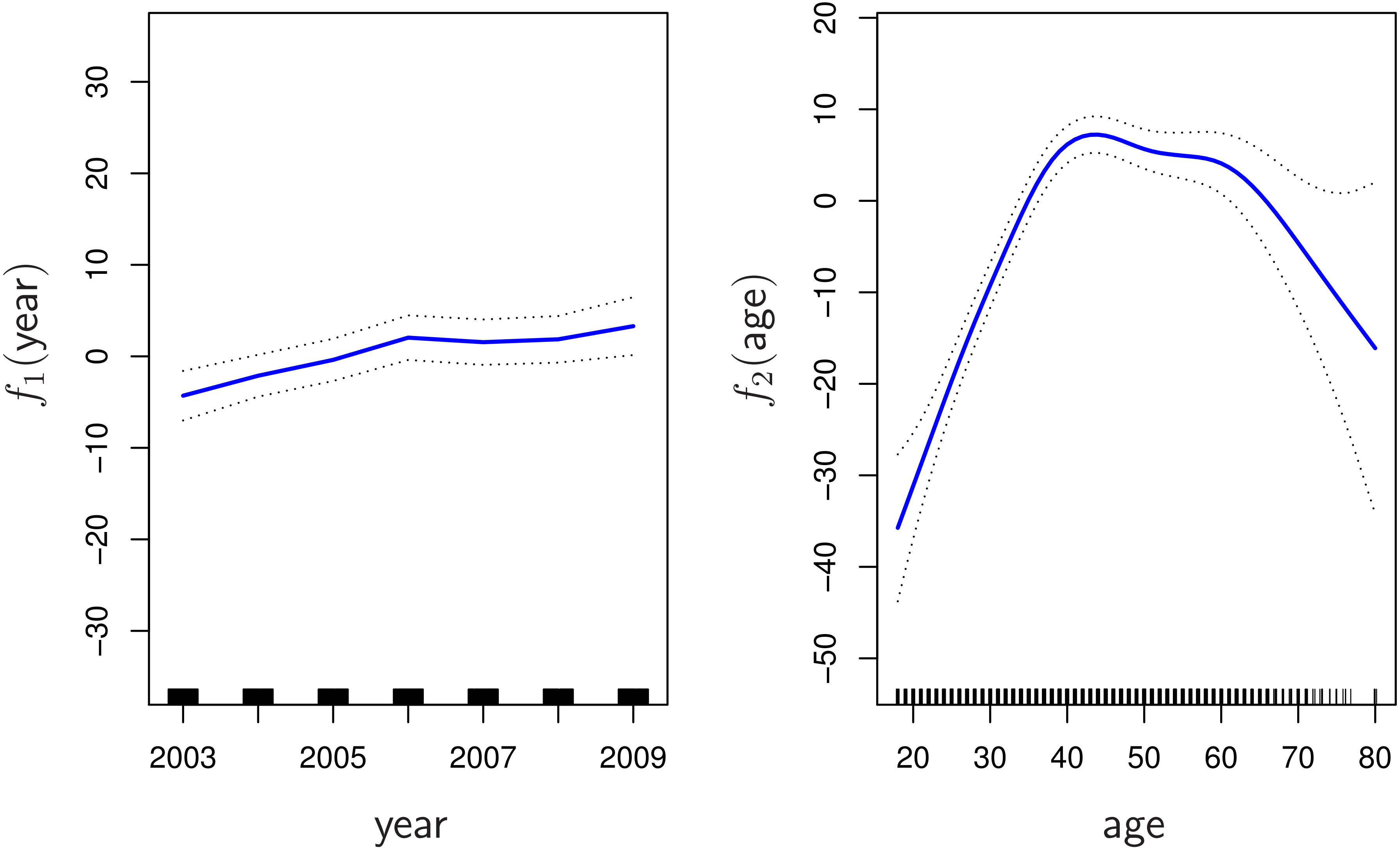
- **Multiple linear regression:**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

- **GAM:**

$$y = \beta_0 + \underbrace{f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p)}_{\text{polynomials, step functions, splines…}} + \varepsilon$$

Ex: `Wage` data, GAM with splines



Pros and Cons of GAMs

Good stuff

- Non-linear functions are potentially more accurate
- Adds local flexibility w/o incurring a global penalty
- Because model is still additive, can still see the effect of each X_j on Y

Bad stuff

- Just like in multiple regression, have to add interaction terms manually*

*In the next chapter, we'll talk about fully general models that can (finally!) deal with this

Lab: Splines and GAMs

- To do today's lab in R: **`spline`**, **`gam`**
- To do (the first half of) today's lab in python: **`patsy`**
- Instructions and code:

  [course website]/labs/lab13-r.html

  [course website]/labs/lab13-py.html
- Full version can be found beginning on p. 293 of ISLR

Up Next

- FP1 due tonight by 11:59pm
- A7 out tonight, due Thursday by 11:59pm
- **Next week**: tree-based methods