Statistical Learning

Lecture 07b - Some Non-Linearity

ANU - RSFAS

Last Updated: Thu Apr 21 13:02:44 2022

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

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.

- May not be interested in the coefficients; but more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \cdots \hat{\beta}_k x_0^k$$

- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}$s, we can get a simple expression for pointwise-variances

$$Var[\hat{f}(x_0)]$$

at any value x_0.

- The we can compute **confidence intervals** (or we could use the bootstrap):

$$\hat{f}(x_0) \pm 1.96\ se[\hat{f}(x_0)]$$

- Recall the matrix form!

$$\boldsymbol{x}_0^{*'} = \left(1, x_0, x_0^2, \ldots, x_0^k\right)$$

$$\hat{\boldsymbol{\beta}} = \left(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)$$

$$\begin{aligned} Var[\hat{f}(x_0)] = Var[\boldsymbol{x}_0^{*'}\hat{\boldsymbol{\beta}}] &= \boldsymbol{x}_0^{*'} V[\hat{\boldsymbol{\beta}}]\boldsymbol{x}_0^{*} \\ &= \sigma^2 \boldsymbol{x}_0^{*'}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_0^{*} \\ &\Rightarrow \hat{\sigma}^2 \boldsymbol{x}_0^{*'}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_0^{*} \end{aligned}$$

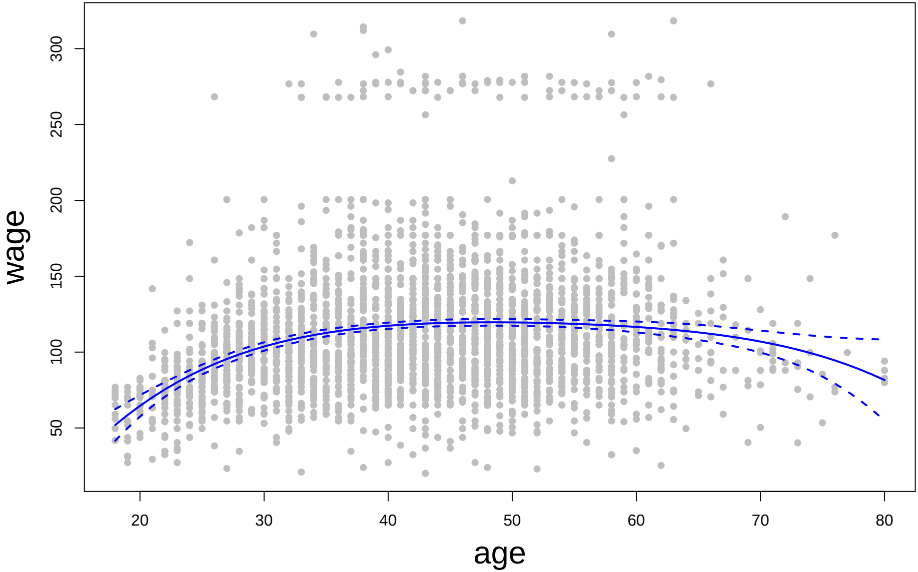
$$se[\hat{f}(x_0)] = \sqrt{Var[\hat{f}(x_0)]}$$

- Let's consider the following data set on Wages (3000 workers).

```
library(ISLR)
data(Wage)
attach(Wage)
head(Wage)
```

```
##        year age            maritl     race       education
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad
## 86582  2004  24 1. Never Married 1. White 4. College Grad
## 161300 2003  45        2. Married 1. White 3. Some College
## 155159 2003  43        2. Married 3. Asian 4. College Grad
## 11443  2005  50       4. Divorced 1. White      2. HS Grad
## 376662 2008  54        2. Married 1. White 4. College Grad
##                    region       jobclass         health
## 231655 2. Middle Atlantic  1. Industrial      1. <=Good
## 86582  2. Middle Atlantic 2. Information 2. >=Very Good
## 161300 2. Middle Atlantic  1. Industrial      1. <=Good
## 155159 2. Middle Atlantic 2. Information 2. >=Very Good
## 11443  2. Middle Atlantic 2. Information      1. <=Good
## 376662 2. Middle Atlantic 2. Information 2. >=Very Good
##        health_ins  logwage      wage
## 231655      2. No 4.318063  75.04315
## 86582       2. No 4.255273  70.47602
## 161300     1. Yes 4.875061 130.98218
## 155159     1. Yes 5.041393 154.68529
## 11443      1. Yes 4.318063  75.04315
## 376662     1. Yes 4.845098 127.11574
```

```r
mod <- lm(wage ~ poly(age, degree = 4))

plot(age, wage, pch = 16, col = "gray",
    cex.lab = 2)
x <- seq(18, 80)

pred.y <- predict(mod, data.frame(age = x),
    se.fit = TRUE)
lines(x, pred.y$fit, col = "blue", lwd = 2)
lines(x, pred.y$fit + 1.96 * pred.y$se.fit,
    col = "blue", lwd = 2, lty = 2)
lines(x, pred.y$fit - 1.96 * pred.y$se.fit,
    col = "blue", lwd = 2, lty = 2)
```


wage
300
250
200
150
100
50
20
30
40
50
60
70
80
age

Logistic Regression

- Consider logistic regression. For example, we model

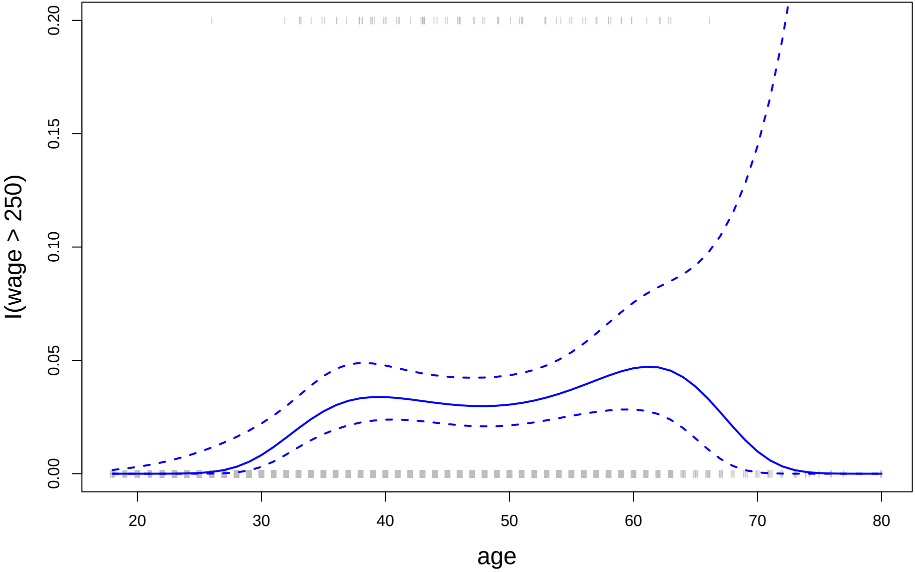
$$Pr(y_i > 250|x_i) = \frac{exp(\hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \cdots \hat{\beta}_k x_0^k)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \cdots \hat{\beta}_k x_0^k)}$$

- To get confidence intervals, compute upper and lower bounds on on the logit scale, and then invert to get on probability scale.
- Recall on the logit scale we have a linear function of the $\hat{\beta}$s:

$$\log\left(\widehat{\frac{\pi_i}{1-\pi_i}}\right) = \hat{\eta}_i = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \cdots \hat{\beta}_k x_0^k$$

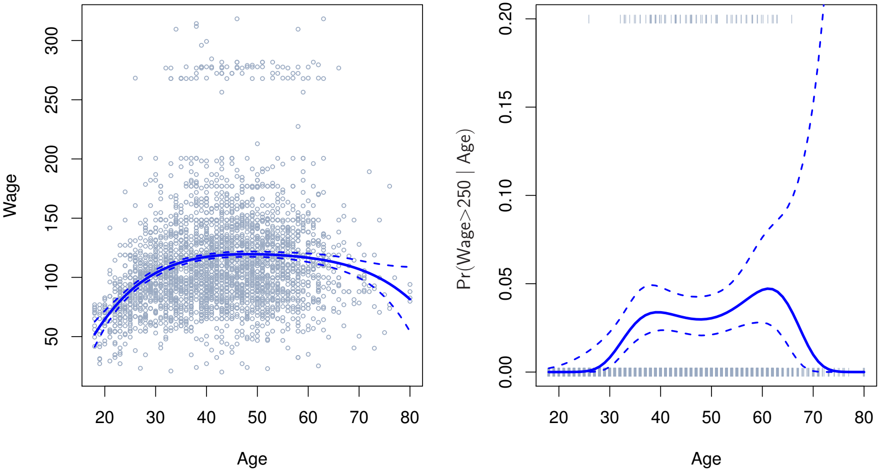
- Caveat: polynomials have notorious tail behavior — very bad for extrapolation.
- We can fit these models using y ∼ poly(x, degree = k) in an R formula.

```r
mod <- glm(I(wage > 250) ~ poly(age,
    degree = 4), family = "binomial")

pred.y <- predict(mod, data.frame(age = x),
    se.fit = TRUE)

pfit <- exp(pred.y$fit)/(1 + exp(pred.y$fit))

se.bands.logit <- cbind(pred.y$fit +
    1.96 * pred.y$se.fit, pred.y$fit -
    1.96 * pred.y$se.fit)

se.bands <- exp(se.bands.logit)/(1 +
    exp(se.bands.logit))

plot(age, I(wage > 250), type = "n",
    ylim = c(0, 0.2), cex.lab = 1.5)

lines(x, pfit, col = "blue", lwd = 2)
matlines(x, se.bands, col = "blue",
    lty = 2, lwd = 2)
```



I(wage > 250)
0.00
0.05
0.10
0.15
0.20
20
30
40
50
60
70
80
age

Degree–4 Polynomial



Step Functions

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

$$C_1(X) = \mathbb{I}(X < 35), C_2(X) = \mathbb{I}(35 \leq X < 50), \ldots, C_k(X) = \mathbb{I}(X > 65)$$

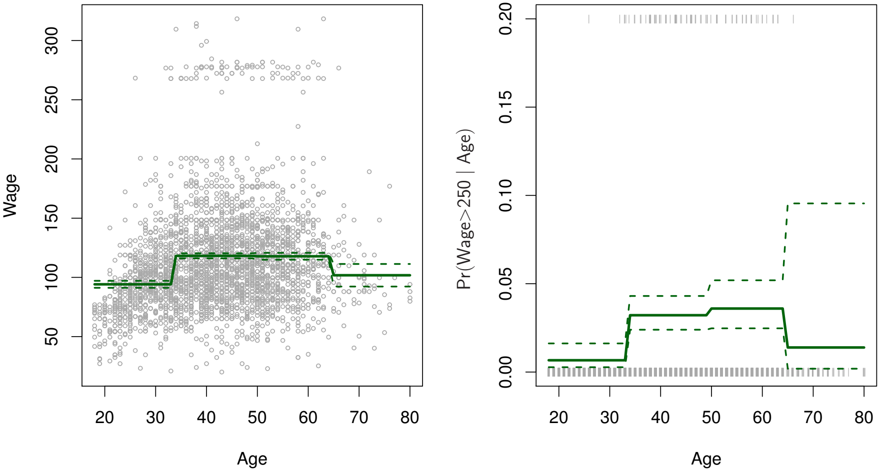
- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of **Year** and **Age**:

$$\mathbb{I}(Year < 2005) \times Age, \quad \mathbb{I}(Year \geq 2005) \times Age$$

would allow for different linear functions in each age category.

- In R: `I(year<2005)` or `cut(age,c(18,25,40,65,90))`.
- The break points are called knots.
- Choice of cut-points or knots can be problematic. For creating nonlinearities, smoother alternatives such as splines are available.

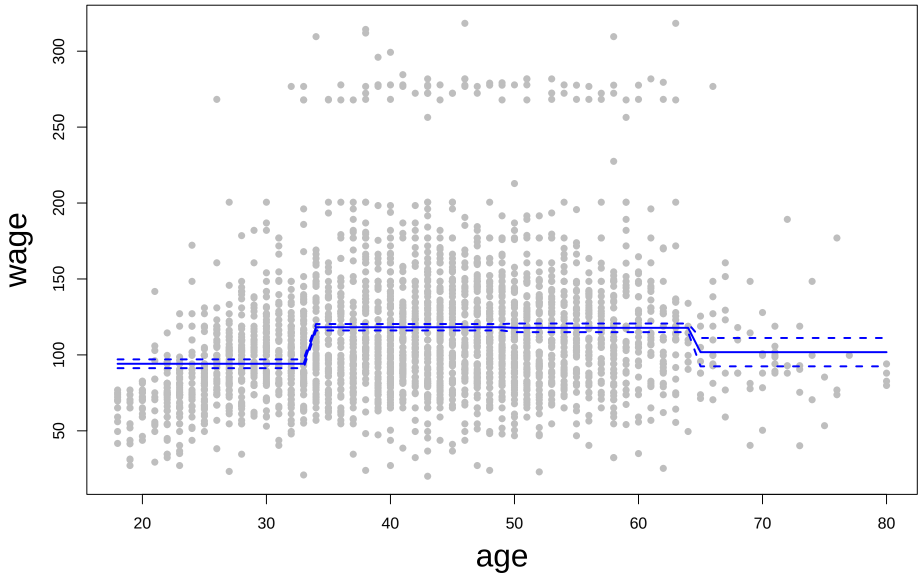
Piecewise Constant



```r
table(cut(age, 4))
```

```
##
## (17.9,33.5]   (33.5,49]   (49,64.5] (64.5,80.1]
##         750        1399         779          72
```

```r
mod <- lm(wage ~ cut(age, 4))

plot(age, wage, pch = 16, col = "gray",
    cex.lab = 2)
x <- seq(18, 80)

pred.y <- predict(mod, data.frame(age = x),
    se.fit = TRUE)
lines(x, pred.y$fit, col = "blue", lwd = 2)
lines(x, pred.y$fit + 1.96 * pred.y$se.fit,
    col = "blue", lwd = 2, lty = 2)
lines(x, pred.y$fit - 1.96 * pred.y$se.fit,
    col = "blue", lwd = 2, lty = 2)
```



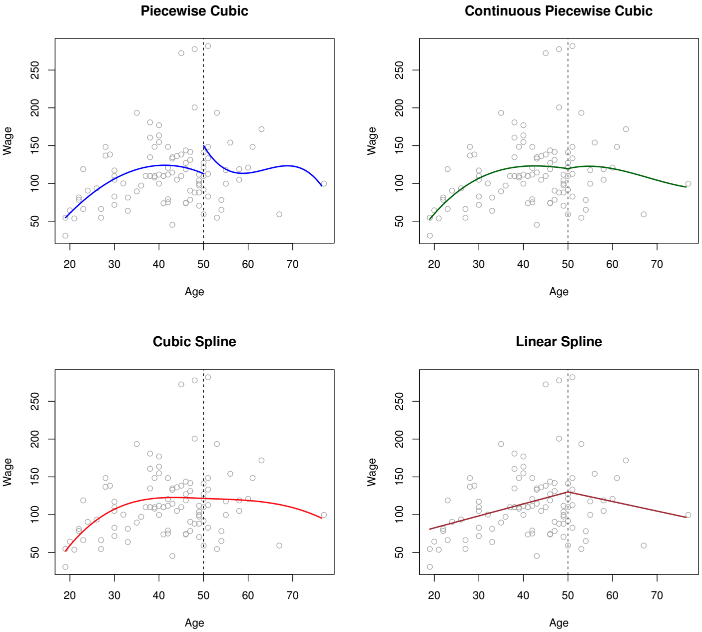
wage
300
250
200
150
100
50
20
30
40
50
60
70
80
age

Let's combine the Ideas - Piece-wise Polynomials

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

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

- 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 piece-wise linear polynomial continuous at each knot.

- We can represent this model as

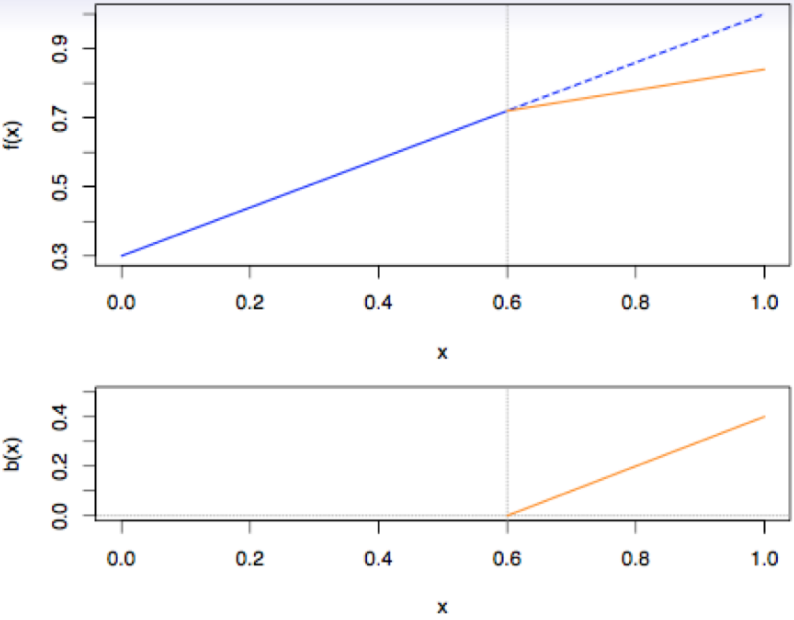
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+1} b_{K+1}(x_i) + \epsilon_i$$

where the b_k are basis functions.

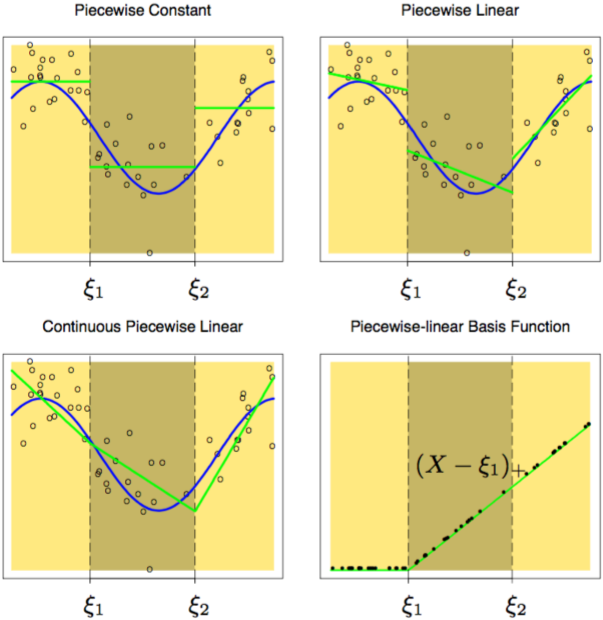
$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \ldots, K \end{aligned}$$

- Here, the $()_+$ means the positive part:

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



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


Piecewise Constant
Piecewise Linear
Continuous Piecewise Linear
Piecewise-linear Basis Function
$(X - \xi_1)_+$
ξ_1
ξ_2

- But we like something that is continuous (bottom left panel).
  - Let's add a constraint!
  - We want the function to the left of ξ_1 to be equal to the function to the right of ξ_1.

$$f(\xi_1^-) = f(\xi_1^+)$$

  - A direct way to do this, is to use a basis function which satisfies this constraint.

$$b_0(x_i) = 1,\ b_1(x_i) = x_i,\ b_2(x_i) = (x_i - \xi_1)_+,\ b_3(x_i) = (x_i - \xi_2)_+$$

Cubic Spline

- With knots at ξ_k, $k = 1, \ldots, K$ is a piece-wise cubic polynomial with continuous derivatives up to order 2 at each knot.

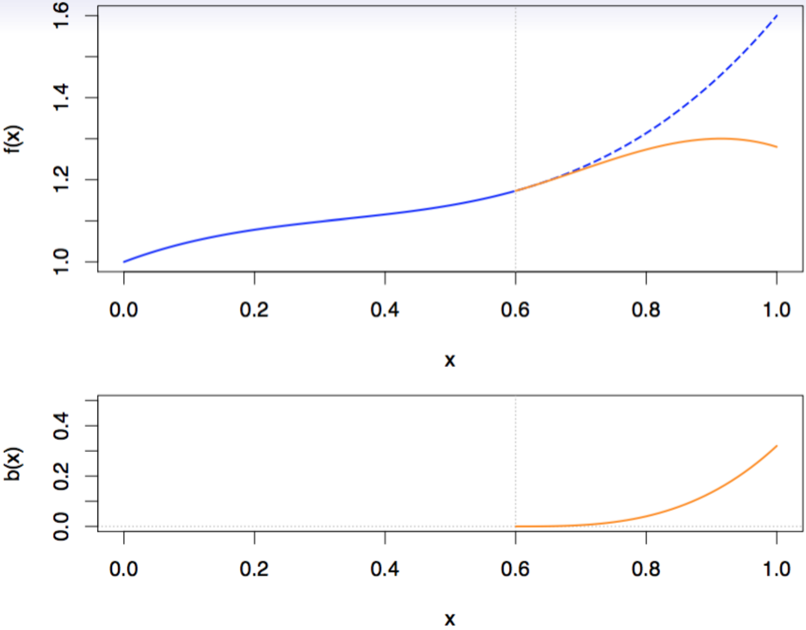
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$$

- Again we can represent this model with truncated power basis functions:

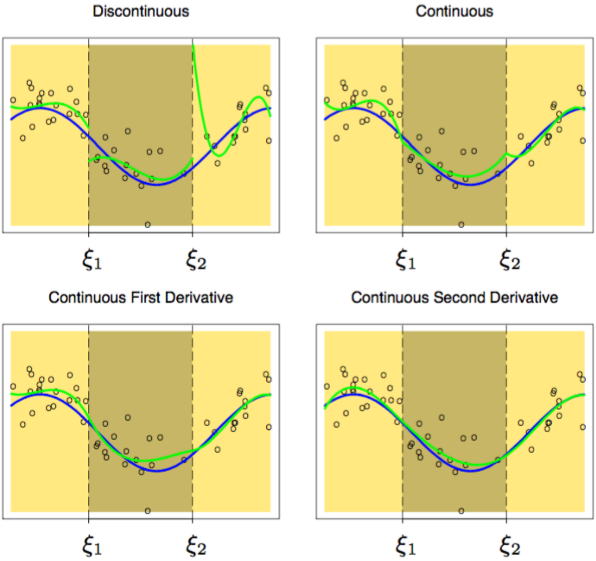
$$\begin{aligned}
b_1(x_i) &= x_i \\
b_2(x_i) &= x_i^2 \\
b_3(x_i) &= x_i^3 \\
b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \ldots, K
\end{aligned}$$

- Where:

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


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

Piecewise Cubic Polynomials

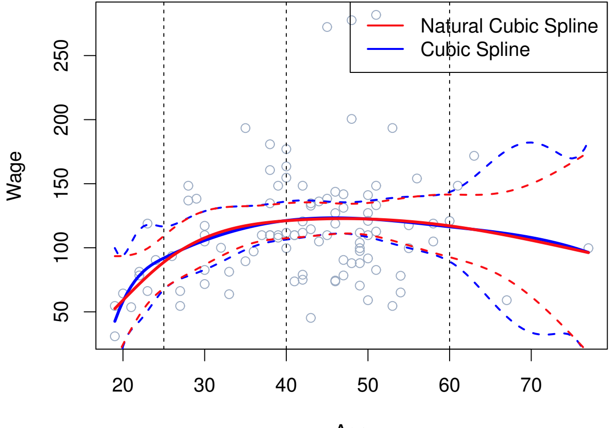


- The bottom right panel is represented by:

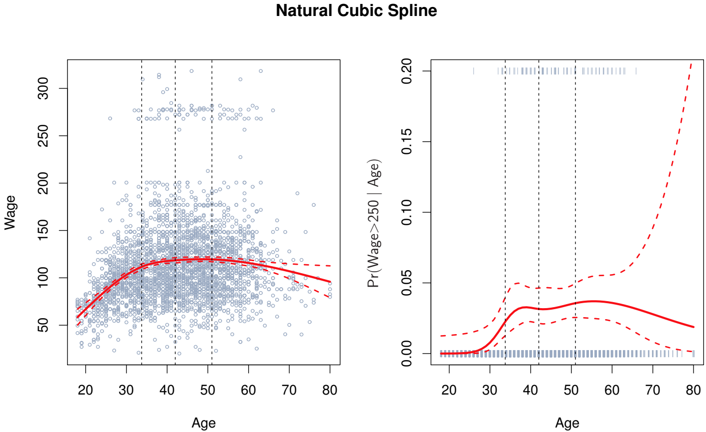
$$b_0(x_i) = 1,\ \ b_1(x_i) = x_i,\ \ b_2(x_i) = x_i^2, b_3(x_i) = x_i^3$$
$$b_4(x_i) = (x_i - \xi_1)_+^3, b_5(x_i) = (x_i - \xi_2)_+^3$$

Natural Cubic Splines

- A natural cubic spline extrapolates linearly beyond the boundary knots. This means there is an extra constraint!



- Fitting splines in R is easy: bs(x, ...) for any degree splines, and ns(x, ...) for natural cubic splines, in package splines2. Actually the package we want is splines, but only archive versions are available, but we can get splines by installing splines2.



- The default is for a cubic spline so the degree=3.
- We can set the knots.
- Or we can set degrees-of-freedom (df) we want to use. We need to set them at least as large as the degree.
- The number of dfs over the degree is the number of knots.
- If degree=3 and df=4 then there is one knot at the median.
- If degree=3 and df=5 then there are two knots at the $1/3$ and $2/3$ quantiles.

```
bs(x, df = NULL, knots = NULL, degree = 3, intercept = FALSE,
Boundary.knots = range(x))
```

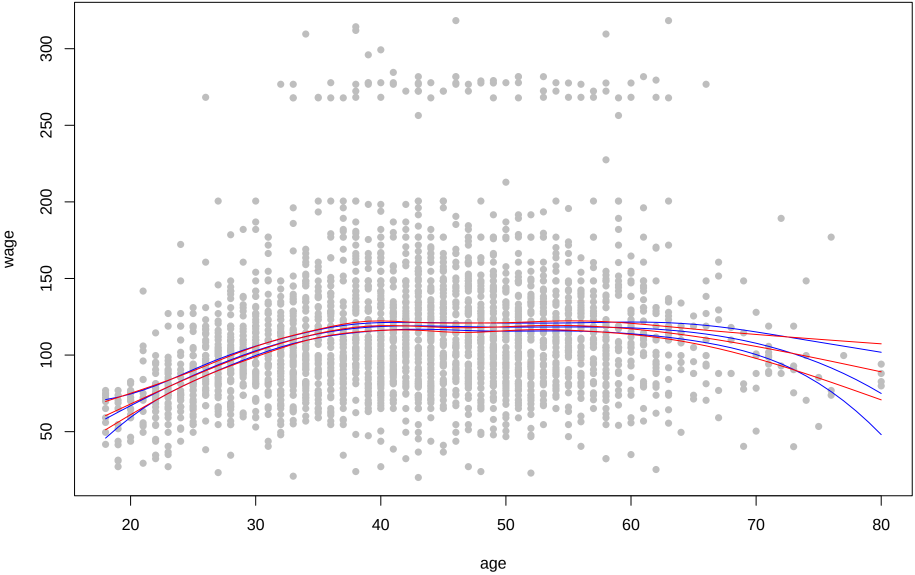
```r
library(splines)

set.seed(1001)
x <- sort(rnorm(10))

test <- bs(x, degree = 3, df = 5)
attr(test, "knots")
```

```
##  33.33333%  66.66667%
## -0.6229437 -0.1775473
```

- Note: for ns() you cannot change the degree, it is set at 3.

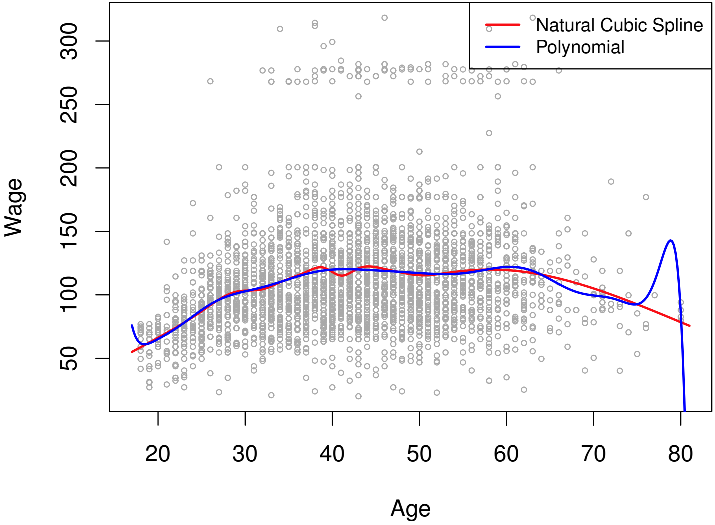
```r
mod1 <- lm(wage ~ bs(age, degree = 3,
    df = 5))
mod2 <- lm(wage ~ ns(age, df = 5))
```

```r
plot(age, wage, pch = 16, col = "gray")
x <- seq(18, 80)

pred.y <- predict(mod1, data.frame(age = x),
    se.fit = TRUE)
lines(x, pred.y$fit, col = "blue")
lines(x, pred.y$fit + 1.96 * pred.y$se.fit,
    col = "blue")
lines(x, pred.y$fit - 1.96 * pred.y$se.fit,
    col = "blue")

pred.y2 <- predict(mod2, data.frame(age = x),
    se.fit = TRUE)
lines(x, pred.y2$fit, col = "red")
lines(x, pred.y2$fit + 1.96 * pred.y2$se.fit,
    col = "red")
lines(x, pred.y2$fit - 1.96 * pred.y2$se.fit,
    col = "red")
```



300
250
200
150
100
50
wage
20
30
40
50
60
70
80
age

- Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.
- ns(age, df=14)
- poly(age, deg=14)


Natural Cubic Spline
Polynomial
Wage
Age
50
100
150
200
250
300
20
30
40
50
60
70
80

Smoothing Splines - More Fun!

- 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 the 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 $g(x_i)$ becomes linear.

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

- 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 ($\boldsymbol{S}$ is just the hat matrix $\boldsymbol{H}$):

$$\hat{\boldsymbol{g}}_\lambda = \boldsymbol{S}_\lambda \boldsymbol{y}$$

- The effective degrees of freedom are given by (note that is just the trace of the matrix):

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

- 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

$$CV_{LOOCV} = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda^{(-i)}(x_i)}{1 - \{\boldsymbol{S}_\lambda\}_{ii}} \right]^2$$

- In R: smooth.spline(age, wage, cv=TRUE)

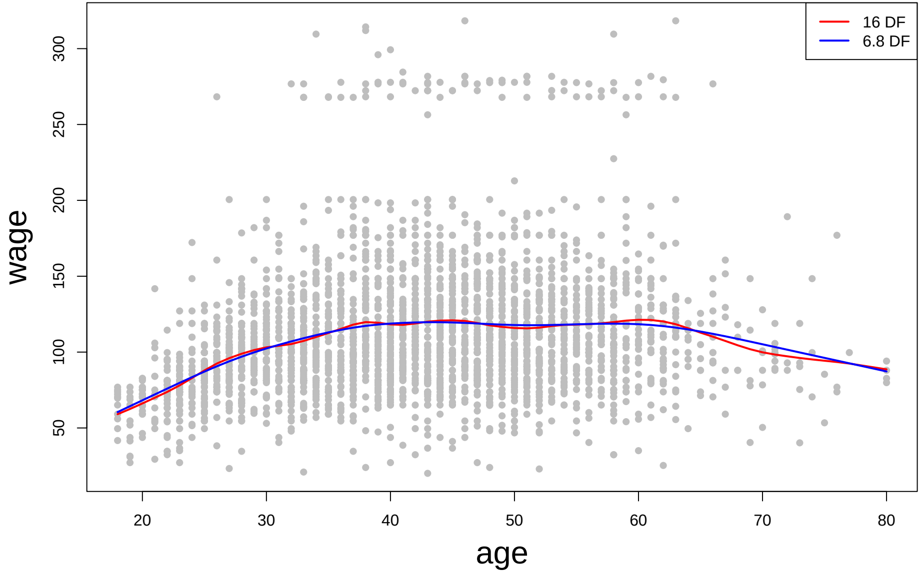
```r
fit <- smooth.spline(age, wage, df = 16)
fit$lambda
```

```
## [1] 0.0006537868
```

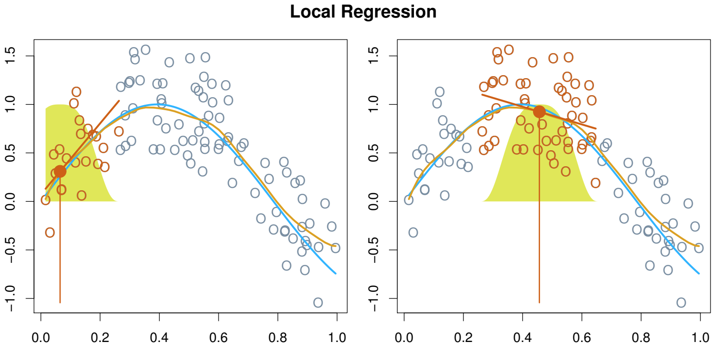
```r
fit2 <- smooth.spline(age, wage, cv = TRUE)
fit2$df
```

```
## [1] 6.794596
```

```r
plot(age, wage, pch = 16, col = "gray")



lines(fit, col = "red ", lwd = 2)
lines(fit2, col = " blue", lwd = 2)

legend("topright", legend = c("16 DF",
    "6.8 DF"), col = c("red", "blue"),
    lty = 1, lwd = 2, cex = 0.8)
```


16 DF
6.8 DF
wage
300
250
200
150
100
50
20
30
40
50
60
70
80
age

Local Regression - Yet Another Approach



- With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares. Use the loess() function in R.

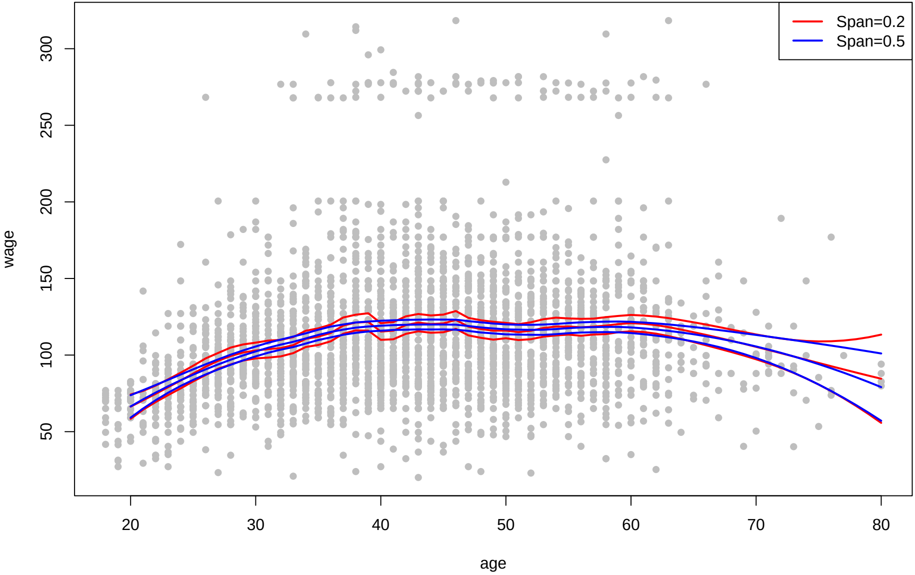
```r
fit <- loess(wage ~ age, span = 0.2,
    data = Wage)
fit
```

```
## Call:
## loess(formula = wage ~ age, data = Wage, span = 0.2)
##
## Number of Observations: 3000
## Equivalent Number of Parameters: 16.42
## Residual Standard Error: 39.92
```

```r
fit2 <- loess(wage ~ age, span = 0.5,
    data = Wage)
fit2
```

```
## Call:
## loess(formula = wage ~ age, data = Wage, span = 0.5)
##
## Number of Observations: 3000
## Equivalent Number of Parameters: 7.13
## Residual Standard Error: 39.89
```

```r
plot(age, wage, pch = 16, col = "gray")
age.grid <- 20:80

pred1 <- predict(fit, data.frame(age = age.grid),
    se = TRUE)
lines(age.grid, pred1$fit, col = "red ",
    lwd = 2)
lines(age.grid, pred1$fit + 1.96 * pred1$se.fit,
    col = "red ", lwd = 2)
lines(age.grid, pred1$fit - 1.96 * pred1$se.fit,
    col = "red ", lwd = 2)

pred2 <- predict(fit2, data.frame(age = age.grid),
    se = TRUE)
lines(age.grid, pred2$fit, col = "blue ",
    lwd = 2)
lines(age.grid, pred2$fit + 1.96 * pred2$se.fit,
    col = "blue", lwd = 2)
lines(age.grid, pred2$fit - 1.96 * pred2$se.fit,
    col = "blue", lwd = 2)

legend("topright", legend = c("Span=0.2",
    "Span=0.5"), col = c("red", "blue"),
    lty = 1, lwd = 2)
```


Span=0.2
Span=0.5
wage
300
250
200
150
100
50
20
30
40
50
60
70
80
age

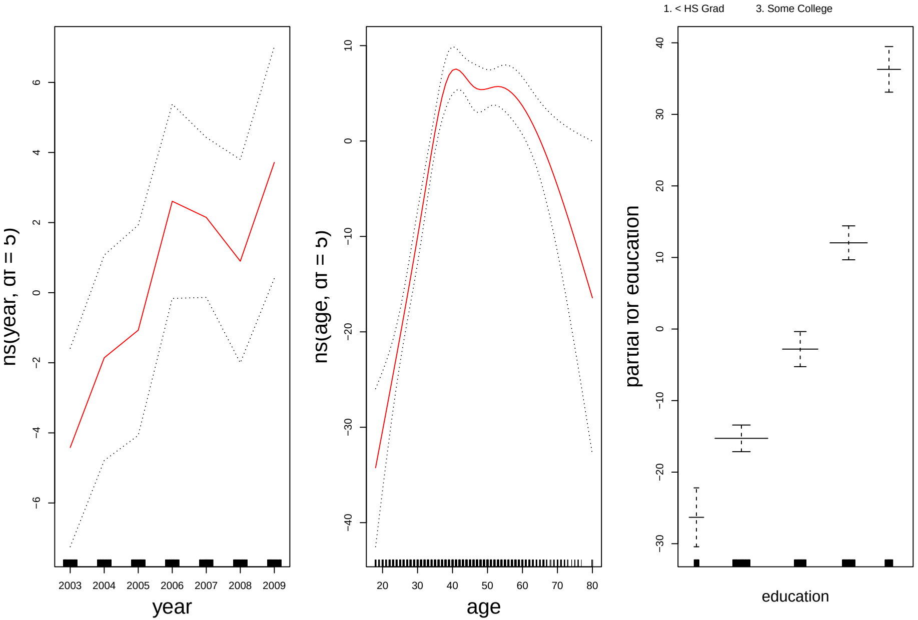
Generalized Additive Models

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

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

- You can fit a GAM using gam() — you can also just use lm() when using bs() or ns().

```r
library(gam)
mod <- gam(wage ~ ns(year, df = 5) +
    ns(age, df = 5) + education)
par(mfrow = c(1, 3))
plot(mod, se = TRUE, col = "red", cex.lab = 2)
```


ns(year, df = 5)
year
2003
2004
2005
2006
2007
2008
2009
ns(age, df = 5)
age
partial for education
education
1. < HS Grad
3. Some College

- Can mix terms — some linear, some nonlinear — and use anova() to compare models.

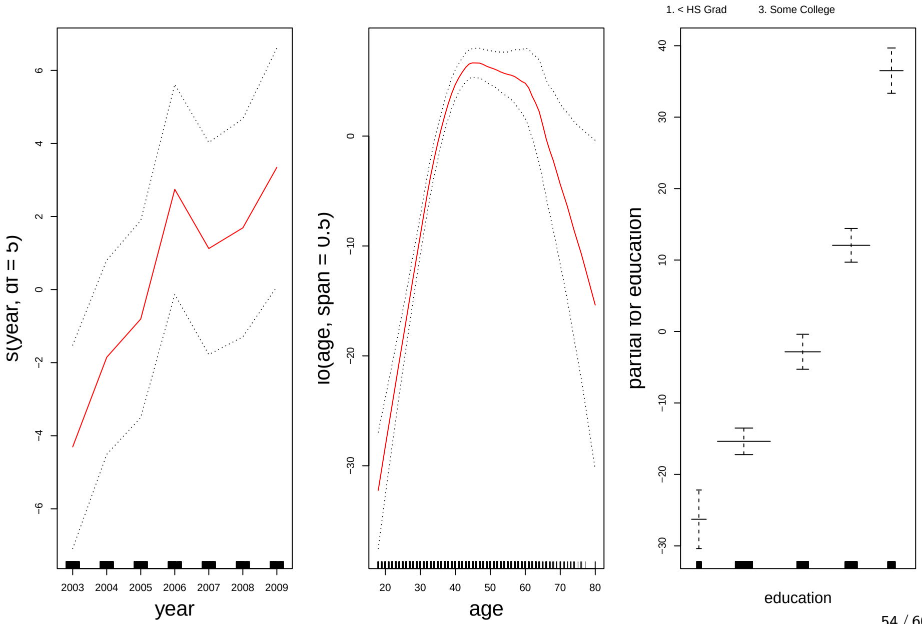
```r
mod <- gam(wage ~ ns(year, df = 5) +
    ns(age, df = 5) + education)
mod2 <- gam(wage ~ ns(age, df = 5) +
    education)

anova(mod2, mod)
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ ns(age, df = 5) + education
## Model 2: wage ~ ns(year, df = 5) + ns(age, df = 5) + education
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      2990    3712881
## 2      2985    3690702  5    22179 0.003025 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Can use smoothing splines or local regression as well:

```
mod <- gam(wage ~ s(year, df = 5) +
    lo(age, span = 0.5) + education)
par(mfrow = c(1, 3))
plot(mod, se = TRUE, col = "red", cex.lab = 2)
```

- Can use smoothing splines or local regression as well:

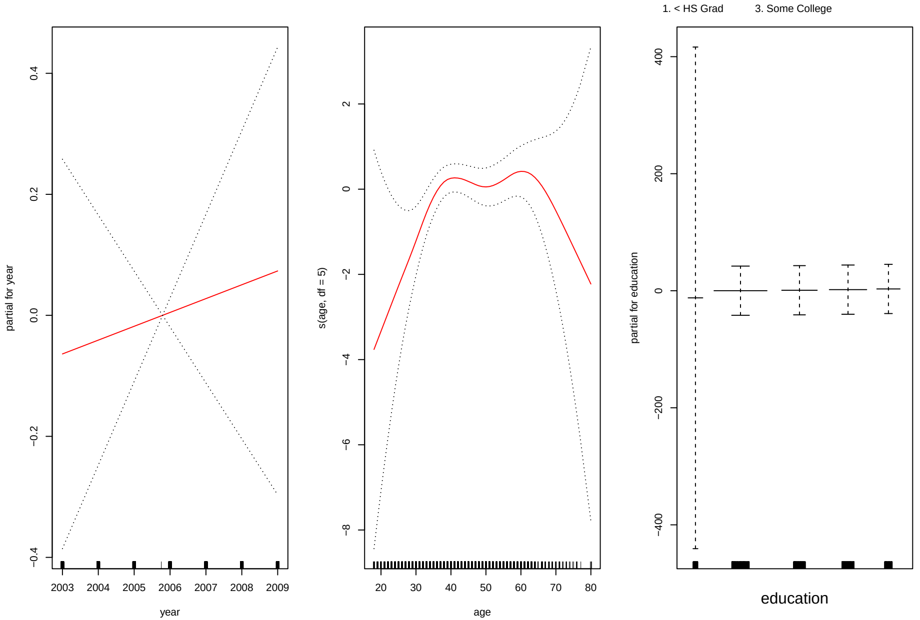


- GAMs are additive, although low-order interactions can be included in a natural way using, e.g. bivariate smoothers or interactions of the form `ns(age,df=5):ns(year,df=5)`.

GAMs for Classification

$$\log\left(\frac{Pr(x_i)}{1 - Pr(x_i)}\right) = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip})$$

```
mod <- gam(I(wage > 250) ~ year + s(age,
    df = 5) + education, family = binomial)
par(mfrow = c(1, 3))
plot(mod, se = TRUE, col = "red")
```



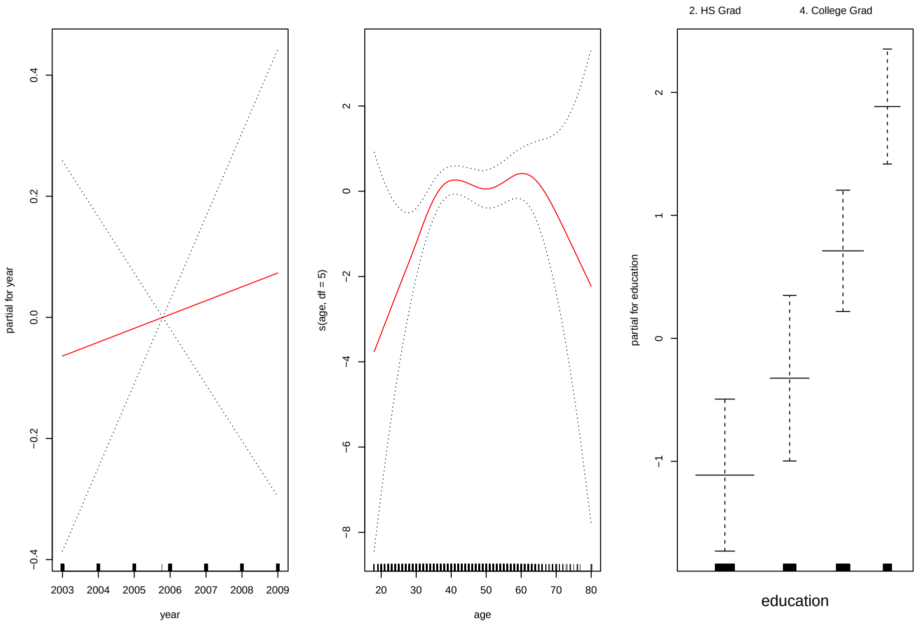
partial for year
0.4
0.2
0.0
−0.2
−0.4
2003
2004
2005
2006
2007
2008
2009
year
s(age, df = 5)
2
0
−2
−4
−6
−8
20
30
40
50
60
70
80
age
1. < HS Grad
3. Some College
partial for education
400
200
0
−200
−400
education

```r
table(education, I(wage > 250))
```

```
##
## education            FALSE TRUE
##   1. < HS Grad         268    0
##   2. HS Grad           966    5
##   3. Some College      643    7
##   4. College Grad      663   22
##   5. Advanced Degree   381   45
```

- Let's fit the model again without the '1. $<$ HS Grad' category.

```r
mod <- gam(I(wage > 250) ~ year + s(age,
    df = 5) + education, family = binomial,
    subset = (education != "1. < HS Grad"))
par(mfrow = c(1, 3))
plot(mod, se = TRUE, col = "red")
```


partial for year
year
2003
2004
2005
2006
2007
2008
2009
s(age, df = 5)
age
20
30
40
50
60
70
80
partial for education
education
2. HS Grad
4. College Grad