7 Moving Beyond Linearity

Andrew Liang

11/20/2020

Notes

Polynomial Regression

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

up to degrees of d

- is a linear model with predictors $x_i, x_i^2, \ldots, x_i^d$
- usually d is no greater than 3 or 4 otherwise model can become overly flexible

Step Functions

- polynomial functions imposes a *global* structure on the non-linear function of X
- use step functions to avoid imposing global strcuture
- break the range of X into *bins*, then fit different constant in each bin
  - converts continuous variable into an *ordered categorical variable*

Create cutpoints $c_1, c_2, \ldots, c_K$ in the range of X, then construct $K+1$ new variables:

$$\begin{gathered} C_0(X) = I(X < c_1) \\ C_0(X) = I(c_1 < X < c_2) \\ C_0(X) = I(c_1 < X < c_2) \\ \ldots \\ C_0(X) = I(c_{K-1} < X < c_K) \\ C_0(X) = I(c_K < X) \end{gathered}$$

where $I(\cdot)$ is an *indicator function* that returns a 1 if the condition is true, 0 otherwise.

For any value of X, $C_0(X) + C_1(X) + \ldots + C_K(X) = 1$, since X must be in exactly one of the $K+1$ intervals

We can then use least sqaure to fit a linear model using $C_1(X), C_2(X), \ldots, C_K(X)$ as predictors:

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

- thus for a given X, at most one of $C_1, C_2, \ldots, C_K$ can be non-zero

Unless there are natural breakpoints in predictors, piecewise-constant functions can miss the action

Basis Functions

polynomial and piecewise-constant regression are special cases of basis function *approach* idea is to have a family of functions or transformations that can be applied to variable X: $b_1(X), b_2X, \ldots, b_K(X)$

Then fit the model:

$$y_i = \beta_o + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \ldots + \beta_K b_K(x_i) + \epsilon_i$$

* basis functions $b_1(\cdot), b_2(\cdot), \ldots, b_K(\cdot)$ must be fixed and known * since this is just a standard linear model with predictors $b_1(x_i), \beta_2 b_2(x_i), \ldots, b_K(x_i)$, all inference tools for linear models are still avaiable in this setting

Regression Splines

Piecewise Polynomials

- similar to polynomial regressions, but instead of fitting it over the entire range of X, we fit separate polynomial regressions over different regions of X
- a quadratic polynomial with a single *knot* at a point c takes the form:

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

- obviously more knots lead to a more flexible piecewise polynomial
  - K different knots through X results in $K + 1$ different polynomial regressions
- without constraints, the model will be discontinuous at each knot

Constraints and Splines

- in order to fix discontinuity problem, we apply a constraints that the model must be continuous, or that both *first* and *second* derivatives are continuous
  - setting both first and second derivatives to be continuous allows for piecewise polynomials to be smooth
    - continuous AT the knot
    - decreases degree of freedom by 3 (continuity, continuity of first derivative, continuity of second derivative)
  - setting only the model to be continuous allows for the model to be continuous but not as smooth (sudden changes in direction at the knots)
    - discontinuous AT the knot
  - each constraint lowers degree of freedom

Natural Cubic Splines

- splines can have high varaince at the outer range of predictors (Where X smaller than smallest knot, bigger than biggest knot)
  - to solve this, we add *boundary constraints*
    - enforce function to be linear at the boundary
      - will have lower CI at boundary regions

Choosing Number of Knots

- common practice to place knots in a uniform fashion
    - specify desired degrees of freedom
- can see which produces best looking curve, or use CV
- regression splines often perform superior to polynomial regression
    - especially at the boundary regions, where variance is highly volatile

Smoothing Splines

- to fit a smooth curve to data, want to find some function $g(x)$ so that:

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

is minimized

- however, no constraints on $g(x)$ would allow us to choose g such that it *interpolates* all of the y_{i}, in other words, we can simply just overfit the data to the extreme
    - to solve this, we can add a penalty term and minimize:

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

where λ is a nonnegative *tuning parameter* * want to minimize the integral of the second derivative of g because * it is the measure of the total change in the function $g'(t)$ in the range of t * if g is smooth, then $g'(t)$ will be close to constant and $\int g''(t)^2 dt$ will be small, vice versa * large λ values will penalize jumpy functions and as $\lambda \to \infty$, g will just be a straight line and thus perfectly smooth λ *controls bias-variance tradeoff of smoothing spline* a function $g(x)$ that minimizes the above equation actually places knots at *every* unique x values: $x_1, x_2, \ldots, x_n$! *NOT the same as a natural cubic spline, rather it is a *shrunken* version of it, where λ controls level of shrinkage

Choosing Smoothing Parameter

- seems like a smoothing spline might have far too many df, but λ effectively controls roughness of spline, and hence controls the *effective degrees of freedom*
- selecting λ is essentially equivalent to selecting how many df you want
- using LOOCV allows us to reduce RSS as small as possible:

$$RSS_{cv}(\lambda) = \sum_{i=1}^{n}(y_i - \hat{g}_\lambda^{(-i)}(x_i))^2$$

Local Regression

- idea is to fit a function at a target point x_0 using only the nearby training observations
- Local Regression Algorithm at $X = x_0$:

1) Gather fraction $s = k/n$ training points whose x_{i} are closest to x_{0}
2) Assign weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that point furthest from x_0 has weight 0, while closest has highest weight. All but k nearest neighbors get weight 0
3) Fit *weighted least squares regression* of y_i on x_i using aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize:

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

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

- the smaller value of s, the more local and wiggly our fit will be
- vice versa, the higher value of s leads to a more global fit

Generalized Additive Models (GAMs)

- extends standard linear model by allowing non-linear functions of each variable, while maintaining *additivity*
- can be applied to both quantitative/qualitative responses

GAMs for Regression

- replace each linear component $\beta_j x_{ij}$ with a smooth, non-linear function $f_j(x_{ij})$:

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

- it is an *additive* model because it calculates separate f_j for each X_j, then ads together all of their contributions
- can use all of the aforementioned methods as building blocks to fit an additive model

Pros of GAMs

- allow us to fit non-linear f_j to each X_j where standard linear regressions will fail to capture
- potentially allow more accurate predictions for response Y
- since model is additive, we can examine each effect X_j has on Y individually holding all other variables fixed - useful for inference
- smoothness of f_j for X_j can be summarized via degrees of freedom

Cons of GAMs

- model restricted to be additive, thus interactions between variables can be missed
  - however we can manually add interaction terms or low-dimensional interaction functions $f_{jk}(X_j, X_k)$ to the model

GAMs for Classification

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 f_1(X_1) + \cdots + \beta_p f_p(X_p)$$

Applied

```
library(ISLR)
attach(Wage) #using Wage data
```

Polynomial Regression and Step Functions

Fitting model of wage against age with degree 4

```
fit <- lm(wage ~ poly(age,4), data = Wage)
coef(summary(fit))
```

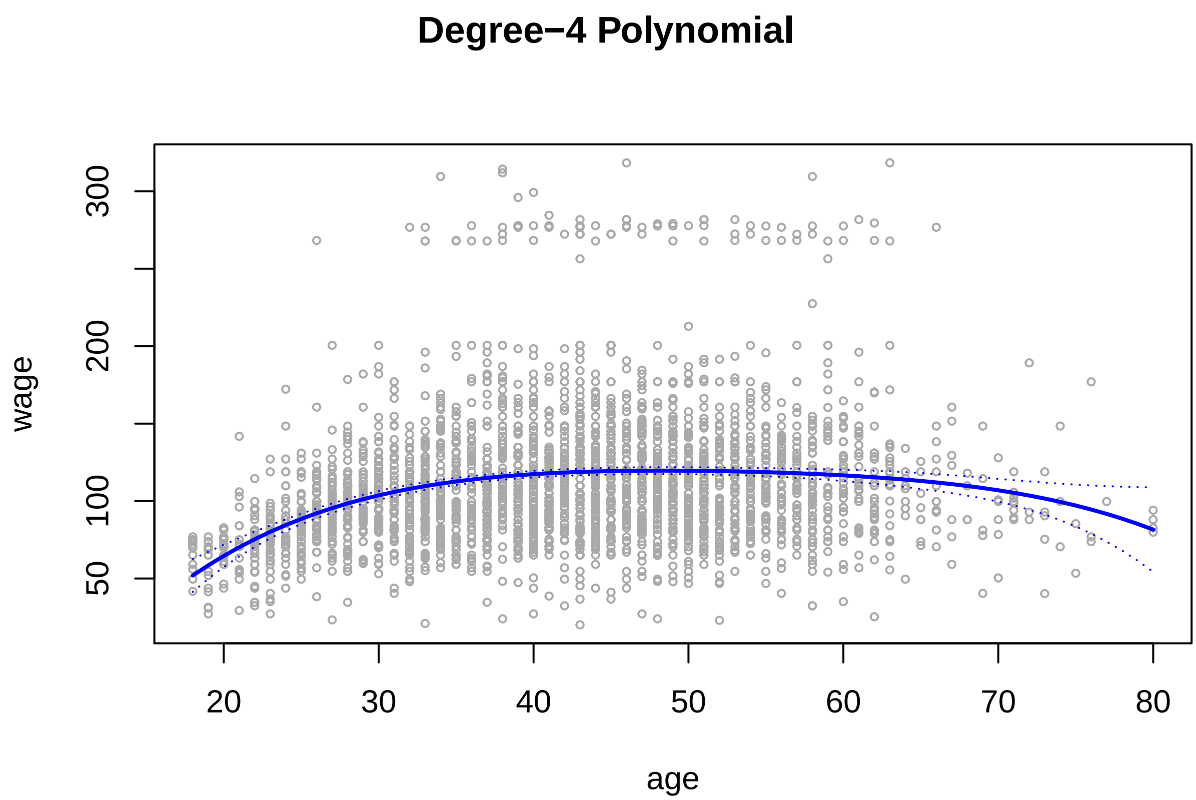
```
##                  Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)    111.70361  0.7287409 153.283015 0.000000e+00
## poly(age, 4)1  447.06785 39.9147851  11.200558 1.484604e-28
## poly(age, 4)2 -478.31581 39.9147851 -11.983424 2.355831e-32
## poly(age, 4)3  125.52169 39.9147851   3.144742 1.678622e-03
## poly(age, 4)4  -77.91118 39.9147851  -1.951938 5.103865e-02
```

Create a grid of values for age at which we want predictions:

```
agelims <- range(age)
age.grid <- seq(from = agelims[1], to = agelims[2]) # list of age values
preds <- predict(fit, newdata = list(age = age.grid), se=T) # use fitted model to predict new age value
se.bands <- cbind(preds$fit + 2*preds$se.fit, preds$fit - 2 * preds$se.fit)
```

Plot the data and add fit from degree-4 polynomial:

```
par(mfrow = c(1,1), mar = c(4.5,4.5,1,1), oma = c(0,0,4,0))
plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
title("Degree-4 Polynomial", outer=T)
lines(age.grid, preds$fit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



Now we fit from linear to a degree-5 polynomial to seek the simplest model sufficient to explain relationship between wage and age:

```
fit1 <- lm(wage ~ age, data = Wage)
fit2 <- lm(wage ~ poly(age, 2), data = Wage)
fit3 <- lm(wage ~ poly(age, 3), data = Wage)
fit4 <- lm(wage ~ poly(age, 4), data = Wage)
fit5 <- lm(wage ~ poly(age, 5), data = Wage)
anova(fit1, fit2, fit3, fit4, fit5)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8888  0.001679 **
## 4   2995 4771604  1      6070   3.8098  0.051046 .
## 5   2994 4770322  1      1283   0.8050  0.369682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Above is what is called a *nested* models, where each model is nested in the proceeding models after

Looking at p-values, it seems that a cubic or quartic polynomial appear to be reasonable to fit the data, but lower or higher order models are not justified. Instead of ANOVA, CV involving the polynomial degree could be used as well

Now we explore a logistic approach; predicting whether an individual earns more than $250,000 per year

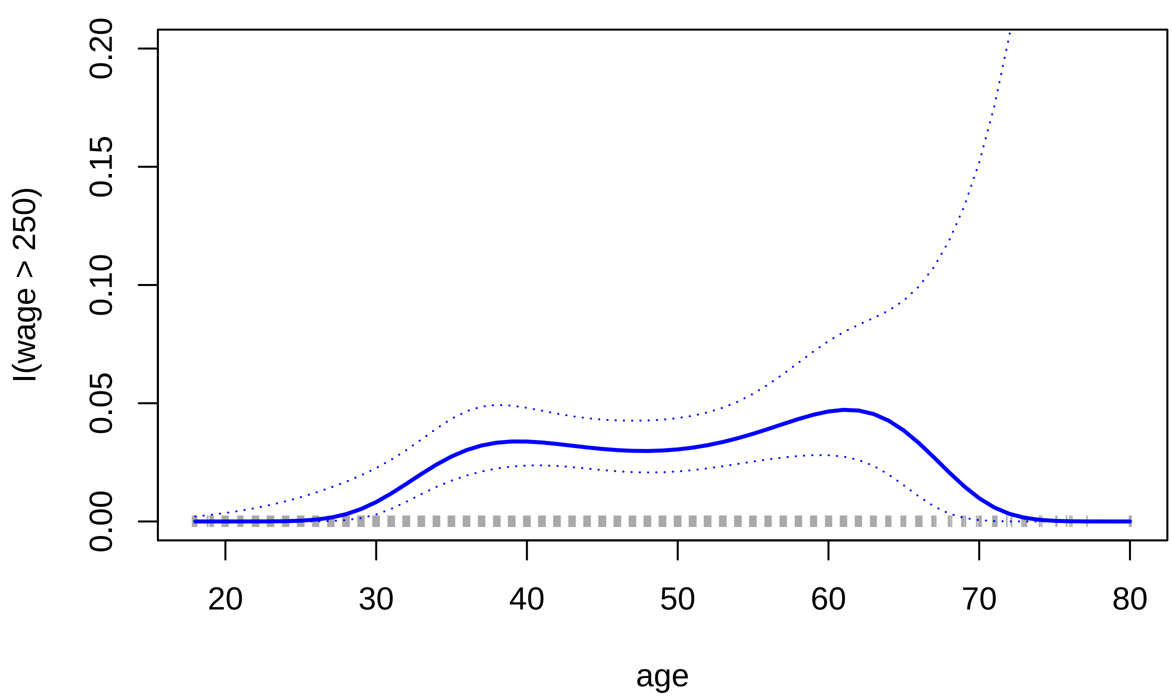
```
#classification
fit <- glm(I(wage > 250) ~ poly(age, 4), data = Wage, family = binomial)
preds <- predict(fit, newdata = list(age = age.grid), se = T) #predict using fitted model using age val
```

Creating confidence intervals for logit:

```
pfit <- exp(preds$fit)/(1+exp(preds$fit))
se.bands.logit <- cbind(preds$fit + 2 * preds$se.fit, preds$fit - 2 * preds$se.fit)
se.bands <- exp(se.bands.logit)/(1+exp(se.bands.logit))
```

Plotting logit:

```
plot(age, I(wage>250), xlim = agelims, type = "n", ylim = c(0,.2))
points(jitter(age), I((age>250)/5), cex = .5, pch = "l", col = "darkgrey")
lines(age.grid, pfit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



Fitting a step function:

```r
table(cut(age,4)) #split ages up into 4 groups
```

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

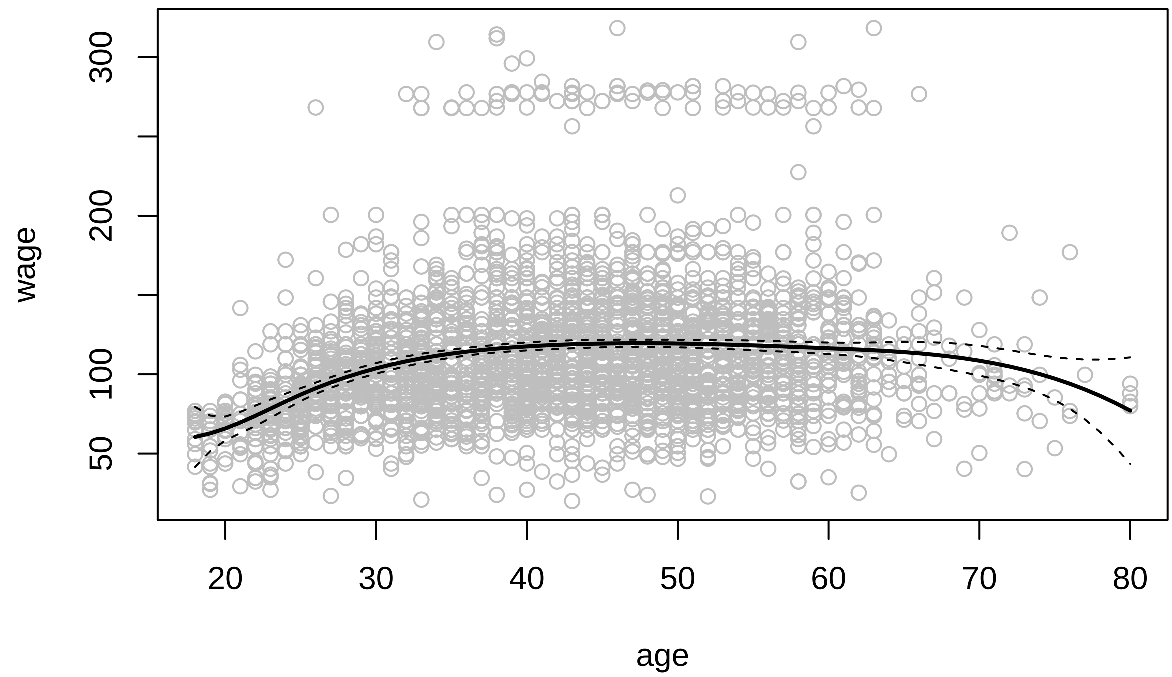
```r
fit <- lm(wage ~ cut(age,4), data = Wage)
coef(summary(fit))
```

```
##                         Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)            94.158392   1.476069 63.789970 0.000000e+00
## cut(age, 4)(33.5,49]   24.053491   1.829431 13.148074 1.982315e-38
## cut(age, 4)(49,64.5]   23.664559   2.067958 11.443444 1.040750e-29
## cut(age, 4)(64.5,80.1]  7.640592   4.987424  1.531972 1.256350e-01
```

It seems that the age groups that are highly significant in predicing wage belong in the first two groups (ages 33.5~64.5)

Splines

```r
library(splines)

fit <- lm(wage ~ bs(age, knots = c(25,40,60)), data = Wage)
pred <- predict(fit, newdata = list(age = age.grid), se = T)
plot(age,wage,col="gray")
lines(age.grid, pred$fit, lwd=2)
lines(age.grid,pred$fit + 2*pred$se, lty = "dashed")
lines(age.grid,pred$fit - 2*pred$se, lty = "dashed")
```



Above we specified knots at ages 25,50,60. Thus, six basis functions were used. From the above, a default of cubic splines are produced. (Cubic spline with three knots produces seven degrees of freedom: one intercept + 6 basis functions)

Could also use df option to produce splines at knots with uniform quantiles of the data:

```
attr(bs(age,df=6),"knots")
```

```
##   25%   50%   75%
## 33.75 42.00 51.00
```

Fitting a natural spline:

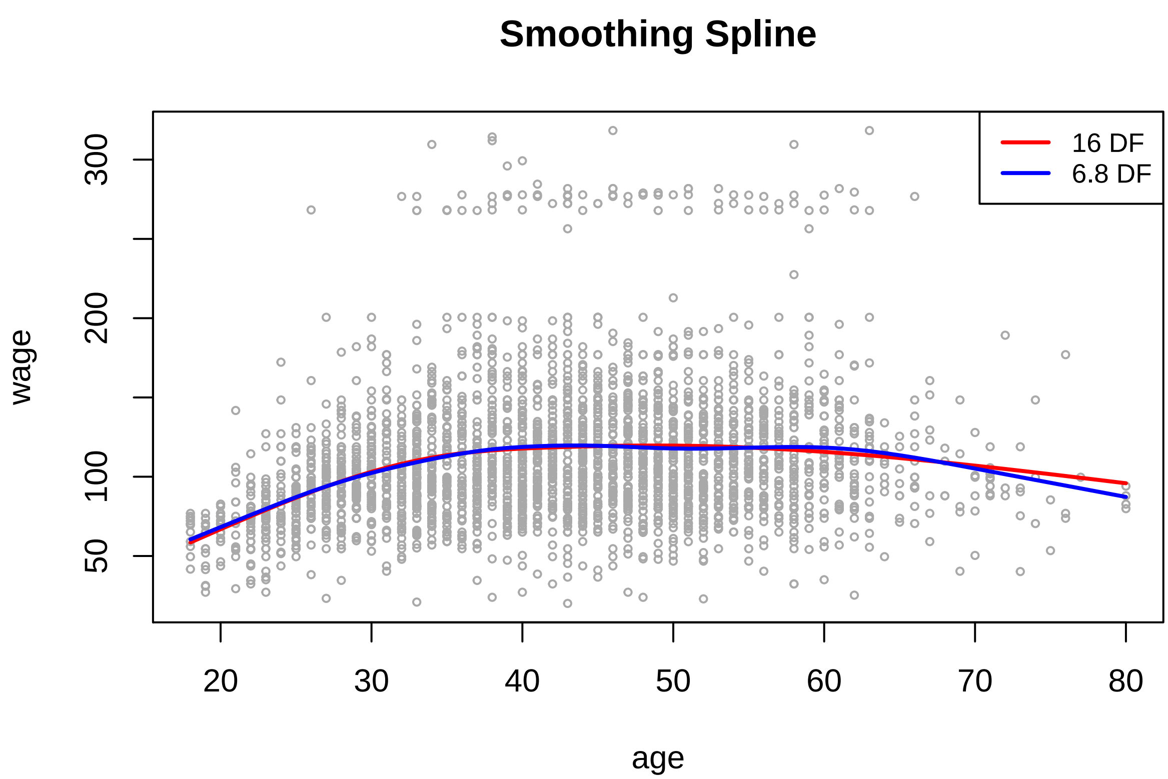
```
fit2 <- lm(wage~ns(age,df=4),data=Wage)
pred2 <- predict(fit2,newdata=list(age=age.grid),se=T)

#plot
plot(age,wage,xlim=agelims,cex=.5,col="darkgrey")
lines(age.grid, pred2$fit,col="red",lwd=2)
title("Smoothing Spline")
fit <- smooth.spline(age,wage,df=16) # function then determines which value of lambda leads to 16df
fit2 <- smooth.spline(age,wage,cv=T) # select smoothness by CV
```

```
## Warning in smooth.spline(age, wage, cv = T): cross-validation with non-unique
## 'x' values seems doubtful
```

```r
print(fit2$df)
```

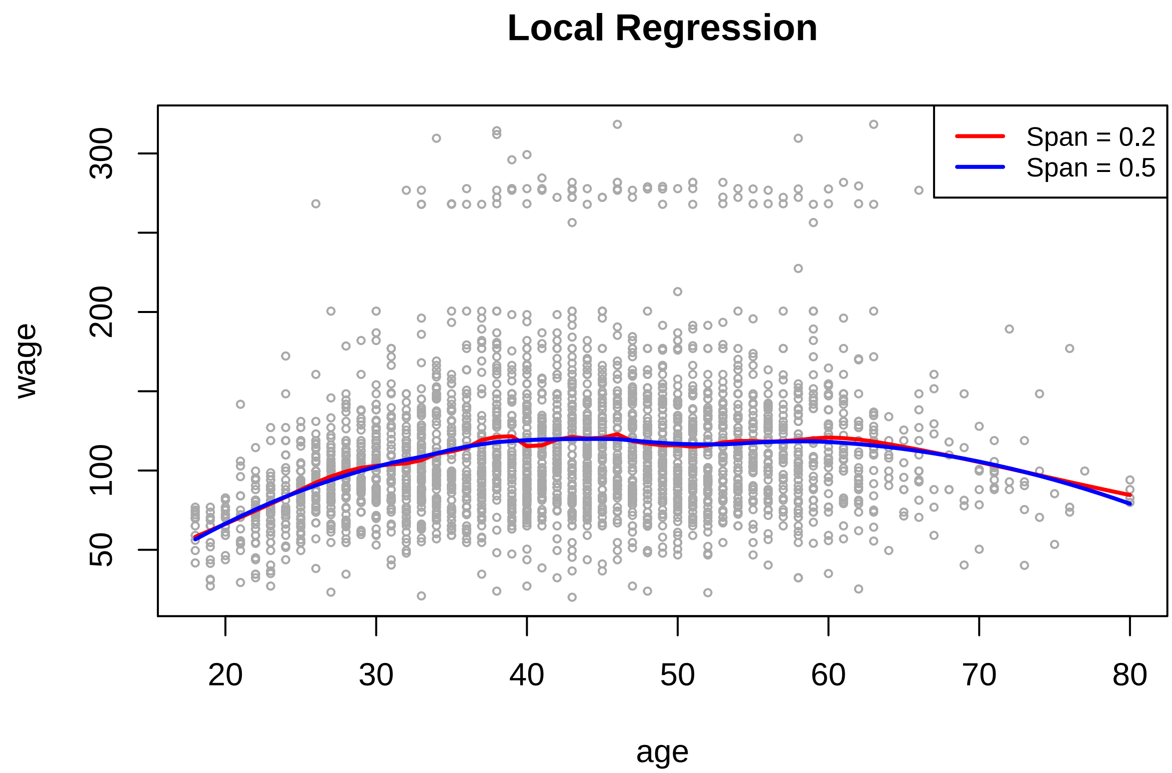
```
## [1] 6.794596
```

```r
lines(fit2,col="blue",lwd=2)
legend("topright",legend=c("16 DF","6.8 DF"), col=c("red","blue"),lty=1,lwd=2,cex=.8)
```



Fitting a local regression:

```r
plot(age,wage,xlim=agelims,cex=.5,col="darkgrey")
title("Local Regression")
fit <- loess(wage~age, span=.2, data=Wage) # each neighborhood consists of 20% of data
fit2 <- loess(wage~age, span=.5, data = Wage) # 50% of data
lines(age.grid,predict(fit,data.frame(age=age.grid)),col="red",lwd=2)
lines(age.grid,predict(fit2,data.frame(age=age.grid)),col="blue",lwd=2)
legend("topright",legend=c("Span = 0.2","Span = 0.5"), col=c("red","blue"),lty=1,lwd=2,cex=.8)
```



GAMs

```
gam1 <- lm(wage~ns(year,4)+ns(age,5)+education, data = Wage)
```

```
library(gam)
```

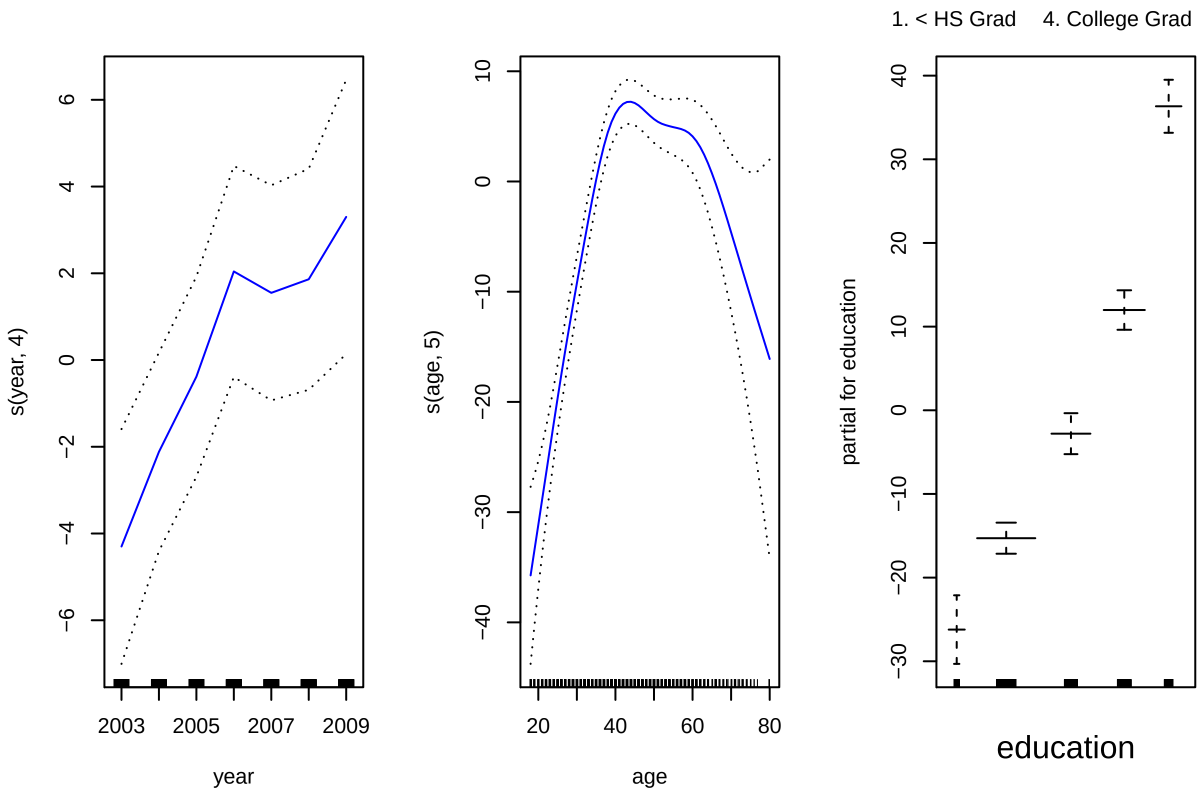
```
## Loading required package: foreach
```

```
## Loaded gam 1.20
```

```
#now use smoothing splines from gam library
gam.m3 <- gam(wage~s(year,4)+s(age,5)+education, data = Wage)
```

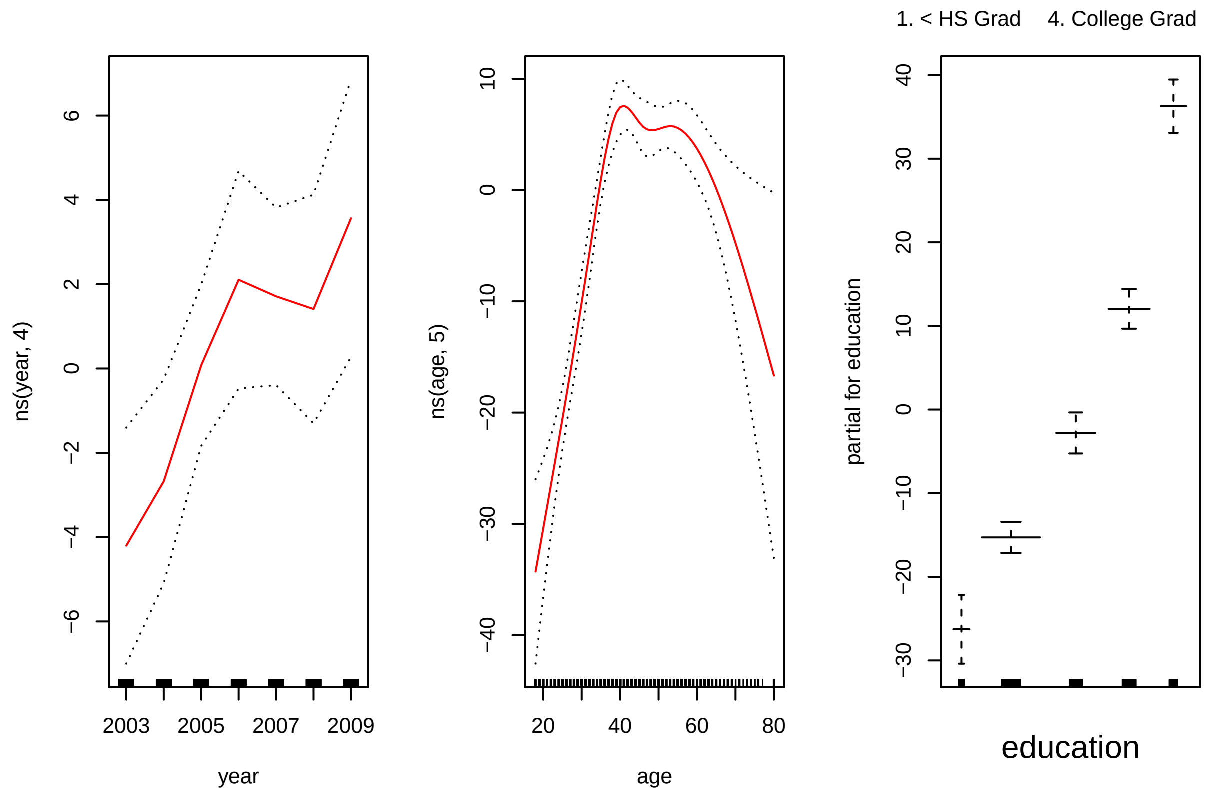
Now we can plot it:

```
par(mfrow=c(1,3))
plot(gam.m3, se=T,col="blue")
```



Using plot.Gam instead, on the model with natural splines:

```
par(mfrow=c(1,3))
plot.Gam(gam1, se=T, col="red")
```



We can see that smoothing and natural splines project very similar results

Now we can use ANOVA testes to determine which of these three models is best:

```
gam.m1 <- gam(wage~s(age,5) + education,data=Wage)
gam.m2 <- gam(wage~year+s(age,5) + education,data=Wage)

anova(gam.m1,gam.m2,gam.m3,test = "F")
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance       F    Pr(>F)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 14.4771 0.0001447 ***
## 3      2986    3689770  3   4071.1  1.0982 0.3485661
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Find that the GAM with a linear function of **year** is better than GAM with no **year** variable. But there is no evidence that a non-linear function of year is needed. This is reinforced by:

```r
summary(gam.m3)
```

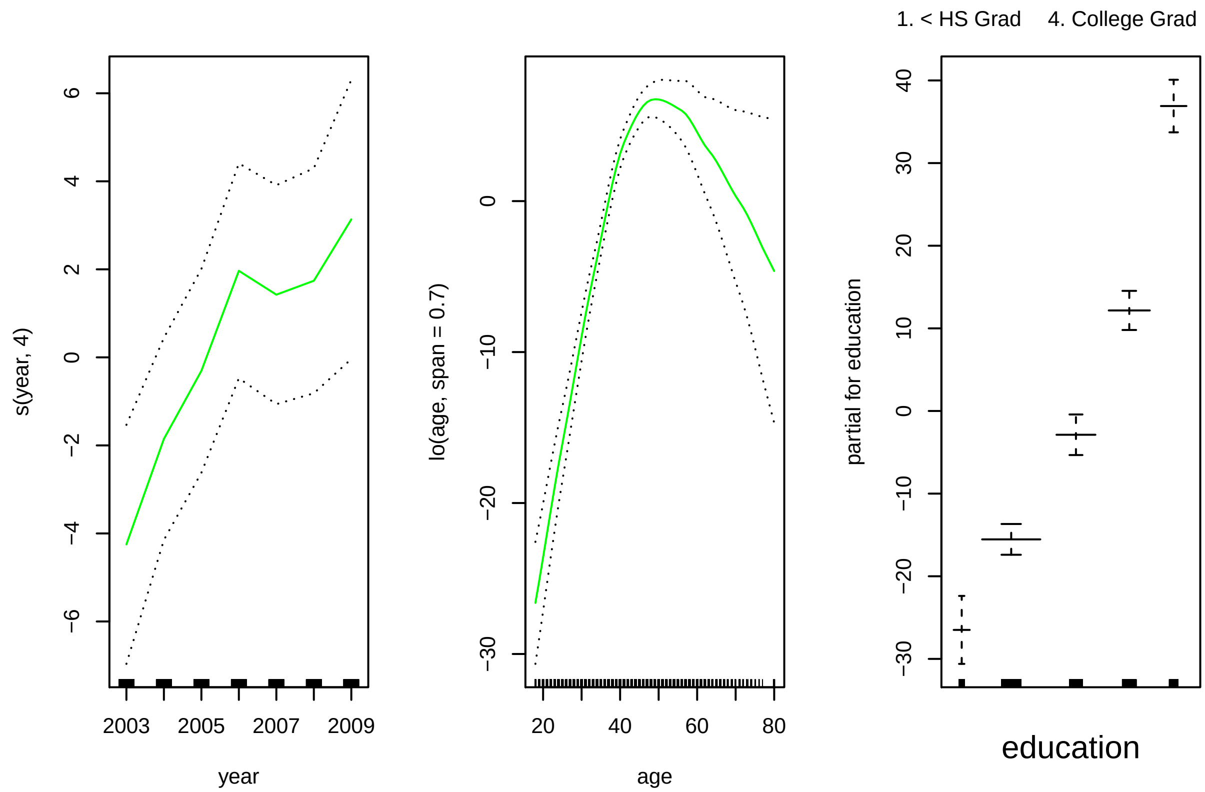
```
##
## Call: gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -119.43  -19.70   -3.33   14.17  213.48
##
## (Dispersion Parameter for gaussian family taken to be 1235.69)
##
##     Null Deviance: 5222086 on 2999 degrees of freedom
## Residual Deviance: 3689770 on 2986 degrees of freedom
## AIC: 29887.75
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##              Df  Sum Sq Mean Sq F value    Pr(>F)
## s(year, 4)    1   27162   27162  21.981 2.877e-06 ***
## s(age, 5)     1  195338  195338 158.081 < 2.2e-16 ***
## education     4 1069726  267432 216.423 < 2.2e-16 ***
## Residuals  2986 3689770    1236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar F  Pr(F)
## (Intercept)
## s(year, 4)        3  1.086 0.3537
## s(age, 5)         4 32.380 <2e-16 ***
## education
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Just like before, we can make predictions for gam objects:

```r
preds <- predict(gam.m2, newdata = Wage)
```

We can implement local regression fits as buildings blocks in GAM:

```r
gam.lo <- gam(wage~s(year,4)+lo(age,span=0.7)+education,data=Wage)

par(mfrow = c(1,3))
plot.Gam(gam.lo, se=T, col="green")
```

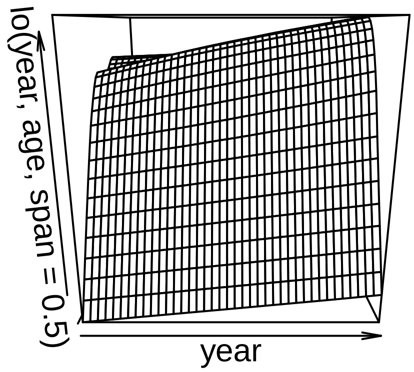


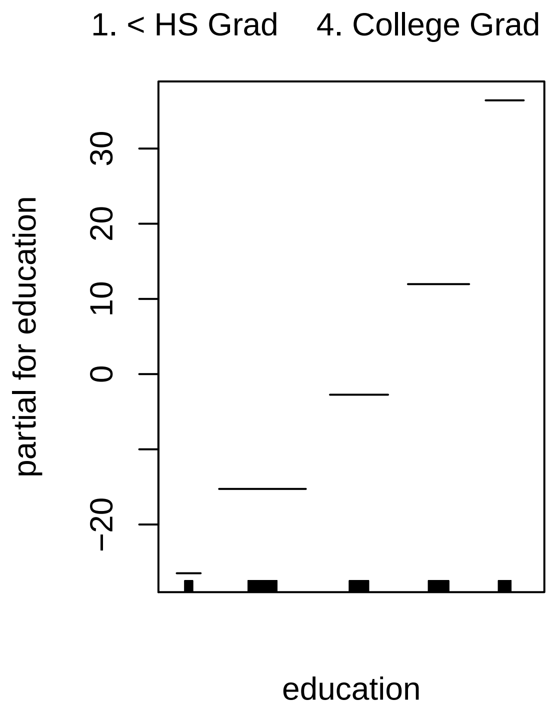
We cacn also use lo() to create interactions before calling gam():

```
gam.lo.i <- gam(wage~lo(year,age,span=0.5)+education, data=Wage)
```

```
## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : liv
## too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : lv
## too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : liv
## too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : lv
## too small. (Discovered by lowesd)
```

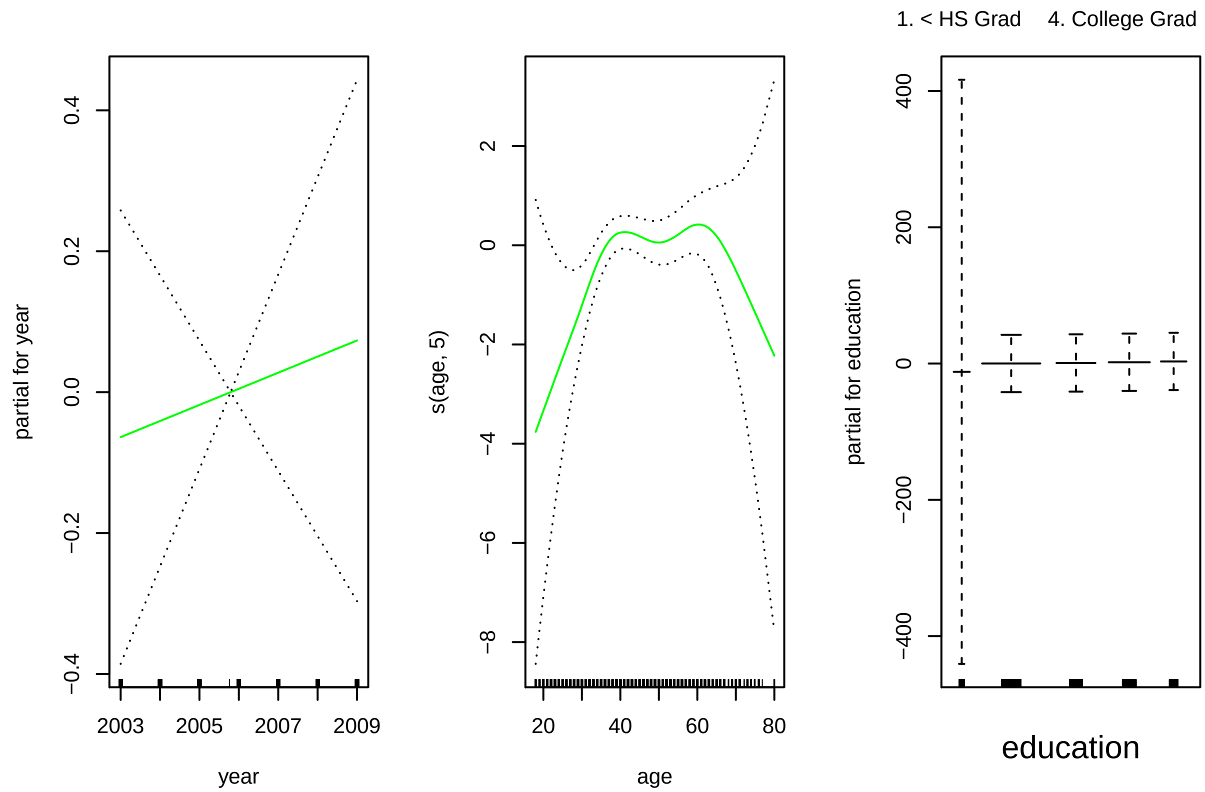
```
library(akima)
par(mfrow = c(1,2))
plot(gam.lo.i)
```





For logistic regression GAMs:

```r
gam.lr <- gam(I(wage>250)~year+s(age,5)+education, family = binomial,data=Wage)
par(mfrow=c(1,3))
plot(gam.lr,se=T, col="green")
```



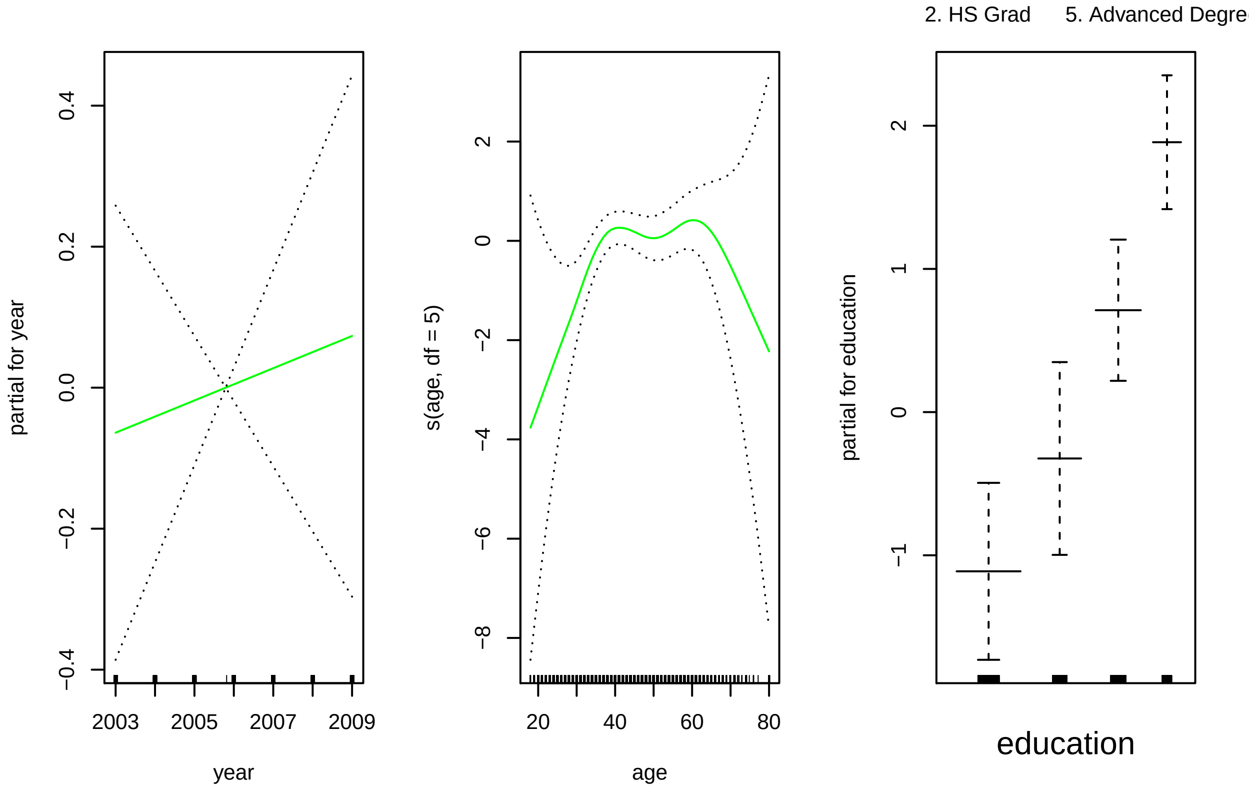
From the education graph, we can see that there are no high earners <HS grad:

```
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
```

Thus, we fit a logistic regression GAM using all but this category:

```
gam.lr.s <- gam(I(wage>250)~year+s(age,df=5)+education,family = binomial, data = Wage, subset = (educati

par(mfrow=c(1,3))
plot(gam.lr.s,se=T,col = "green")
```



2. HS Grad
5. Advanced Degre
partial for year
0.4
0.2
0.0
−0.2
−0.4
2003
2005
2007
2009
year
s(age, df = 5)
2
0
−2
−4
−6
−8
20
40
60
80
age
partial for education
2
1
0
−1
education