Nonlinear

Yang Wang

2/20/2023

```
# illustrate the use of nonlieaner models in R
```

```
library(ISLR)
data("Wage")
```

Polynomial Regression

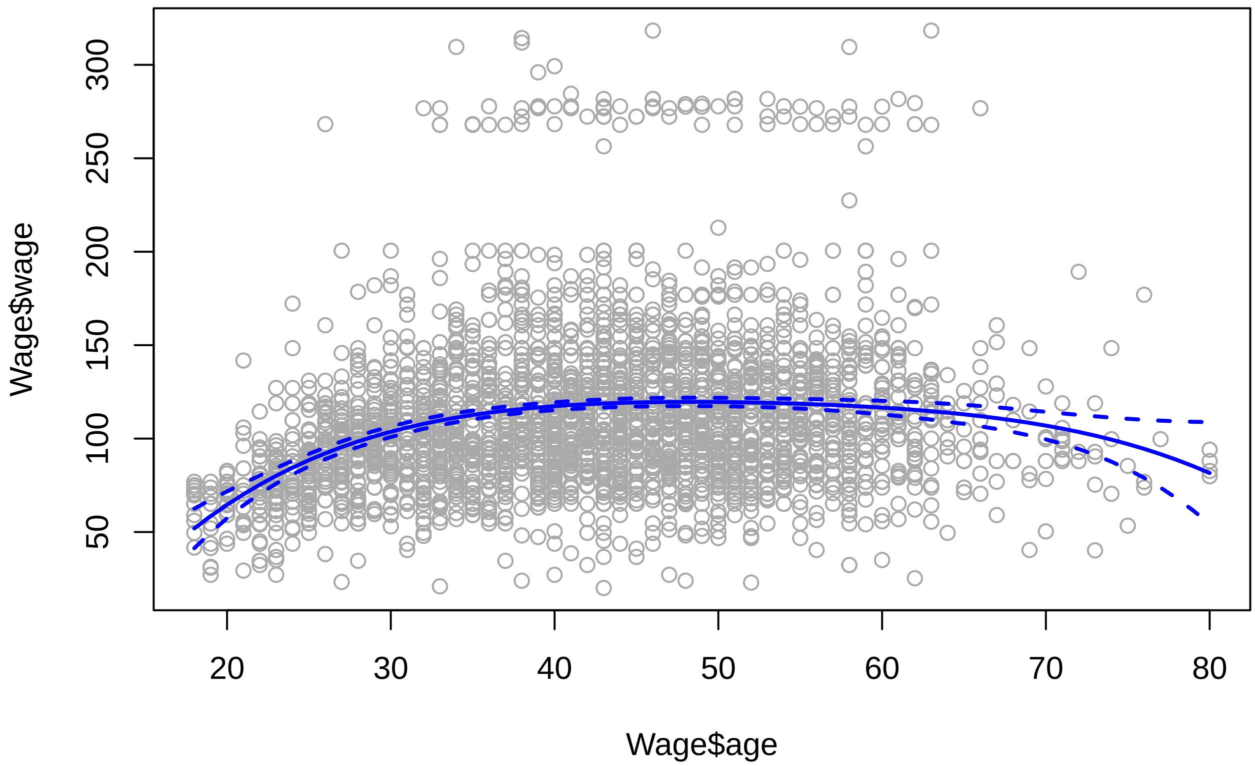
The poly() function generates a basis of orthoganal polynomials.

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

```
##
## Call:
## lm(formula = wage ~ poly(age, 4), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.707 -24.626  -4.993  15.217 203.693
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    111.7036     0.7287 153.283  < 2e-16 ***
## poly(age, 4)1  447.0679    39.9148  11.201  < 2e-16 ***
## poly(age, 4)2 -478.3158    39.9148 -11.983  < 2e-16 ***
## poly(age, 4)3  125.5217    39.9148   3.145  0.00168 **
## poly(age, 4)4  -77.9112    39.9148  -1.952  0.05104 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.91 on 2995 degrees of freedom
## Multiple R-squared:  0.08626,    Adjusted R-squared:  0.08504
## F-statistic: 70.69 on 4 and 2995 DF,  p-value: < 2.2e-16
```

let's make a plot of the fitted function, along with the standard errors of the fit.

```
agelims = range(Wage$age)
age.grid = seq(from = agelims[1], to=agelims[2])
preds = predict(fit, newdata = list(age = age.grid), se = TRUE)
se.band = cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
plot(Wage$age, Wage$wage, col = "darkgrey")
lines(age.grid, preds$fit, lwd = 2, col = "blue")
matlines(age.grid, se.band, lty = 2, col = "blue", lwd = 2)
```



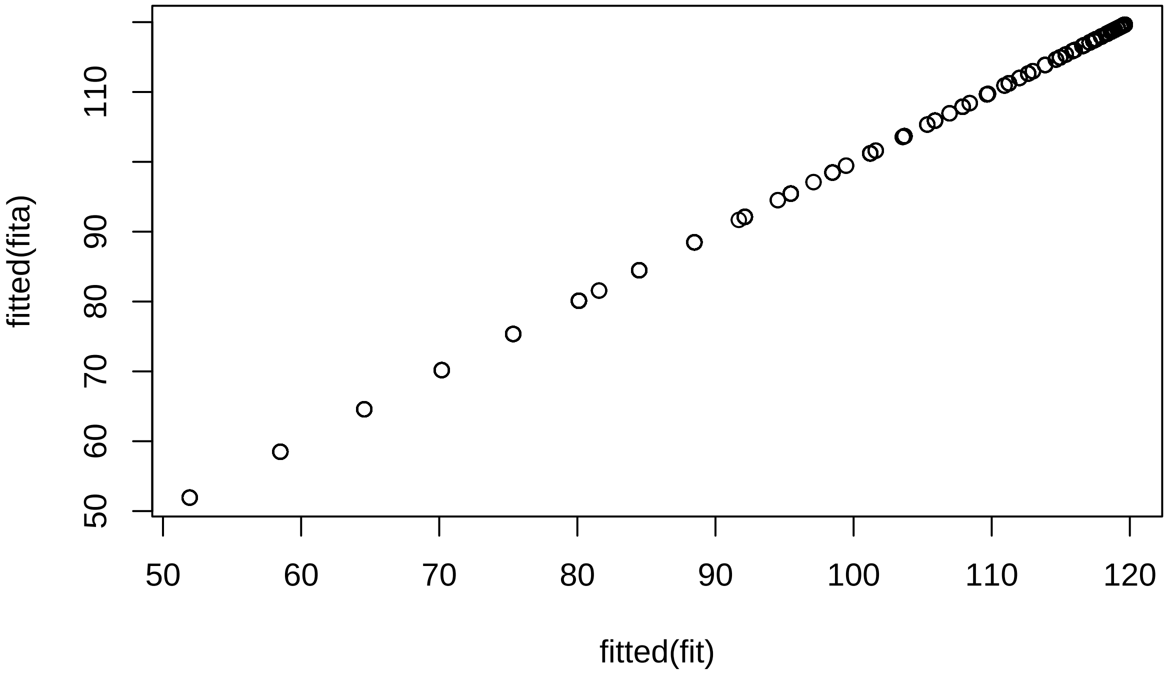
Other more direct ways of doing this in R. “age^2” means something else to the formula language. I() is a “wrapper” function. I(age^2) is protected.

```
fita = lm(wage~age+I(age^2)+I(age^3)+I(age^4), data = Wage)
summary(fita)
```

```
##
## Call:
## lm(formula = wage ~ age + I(age^2) + I(age^3) + I(age^4), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.707 -24.626  -4.993  15.217 203.693
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.842e+02  6.004e+01  -3.067 0.002180 **
## age          2.125e+01  5.887e+00   3.609 0.000312 ***
## I(age^2)    -5.639e-01  2.061e-01  -2.736 0.006261 **
## I(age^3)     6.811e-03  3.066e-03   2.221 0.026398 *
## I(age^4)    -3.204e-05  1.641e-05  -1.952 0.051039 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.91 on 2995 degrees of freedom
## Multiple R-squared:  0.08626,    Adjusted R-squared:  0.08504
## F-statistic: 70.69 on 4 and 2995 DF,  p-value: < 2.2e-16
```

two method provide the same fits

```
plot(fitted(fit), fitted(fita))
```



compare models using the function anova()

```
fita = lm(wage ~ education, data = Wage)
fitb = lm(wage ~ education + age, data = Wage)
fitc = lm(wage ~ education + poly(age,2), data = Wage)
fitd = lm(wage ~ education + poly(age,3), data = Wage)
anova(fita, fitb, fitc, fitd)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ education
## Model 2: wage ~ education + age
## Model 3: wage ~ education + poly(age, 2)
## Model 4: wage ~ education + poly(age, 3)
##   Res.Df     RSS Df Sum of Sq        F Pr(>F)
## 1   2995 3995721
## 2   2994 3867992  1    127729 102.7378 <2e-16 ***
## 3   2993 3725395  1    142597 114.6969 <2e-16 ***
## 4   2992 3719809  1      5587   4.4936 0.0341 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Polynomial logistic regression

Now we fit a logistic regression model to a binary response variable. we code the big earners (>250k) as 1, else 0

```
fit = glm(I(wage>250)~ poly(age,3), data = Wage, family = binomial)
summary(fit)
```

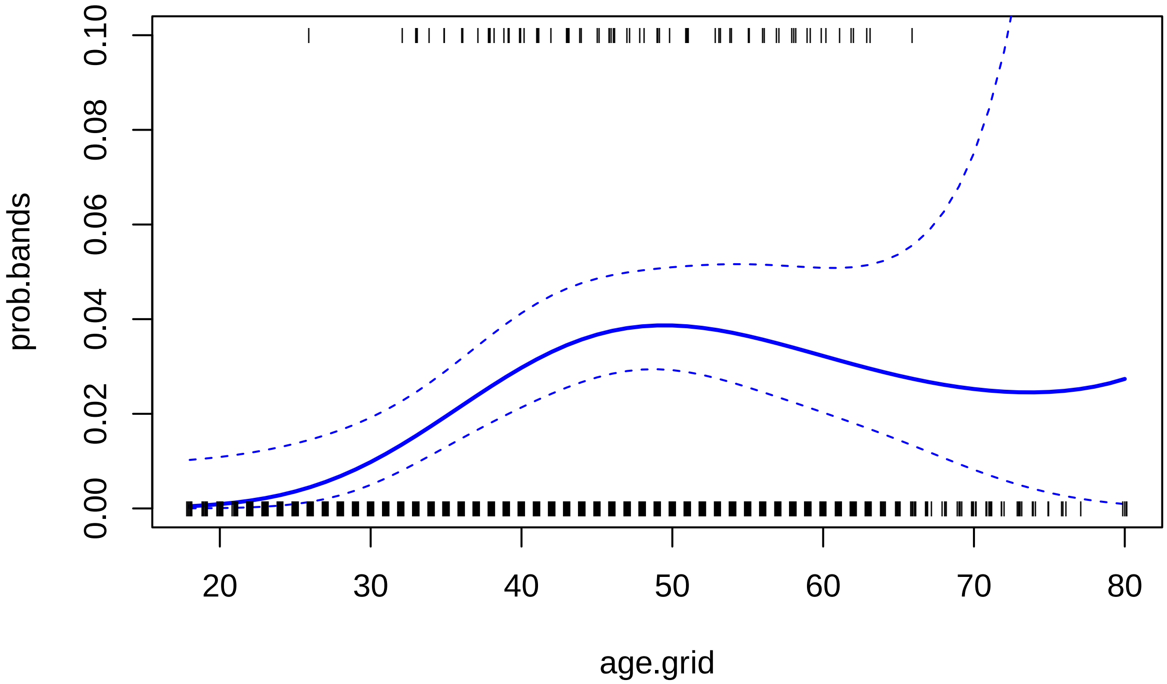
```
##
```

```
## Call:
## glm(formula = I(wage > 250) ~ poly(age, 3), family = binomial,
##     data = Wage)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.2808  -0.2736  -0.2487  -0.1758   3.2868
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.8486     0.1597 -24.100  < 2e-16 ***
## poly(age, 3)1  37.8846    11.4818   3.300 0.000968 ***
## poly(age, 3)2 -29.5129    10.5626  -2.794 0.005205 **
## poly(age, 3)3   9.7966     8.9990   1.089 0.276317
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 730.53  on 2999  degrees of freedom
## Residual deviance: 707.92  on 2996  degrees of freedom
## AIC: 715.92
##
## Number of Fisher Scoring iterations: 8
```

```
preds = predict(fit, list(age = age.grid), se = T)
se.bands = preds$fit + cbind(fit = 0,lower = -2*preds$se.fit, upper = 2*preds$se)
```

We have done the computation (fit, confidence interval) on the logit scale. To transform we need to apply the inverse logit mapping $p = \frac{e^{\eta}}{1+e^{\eta}}$

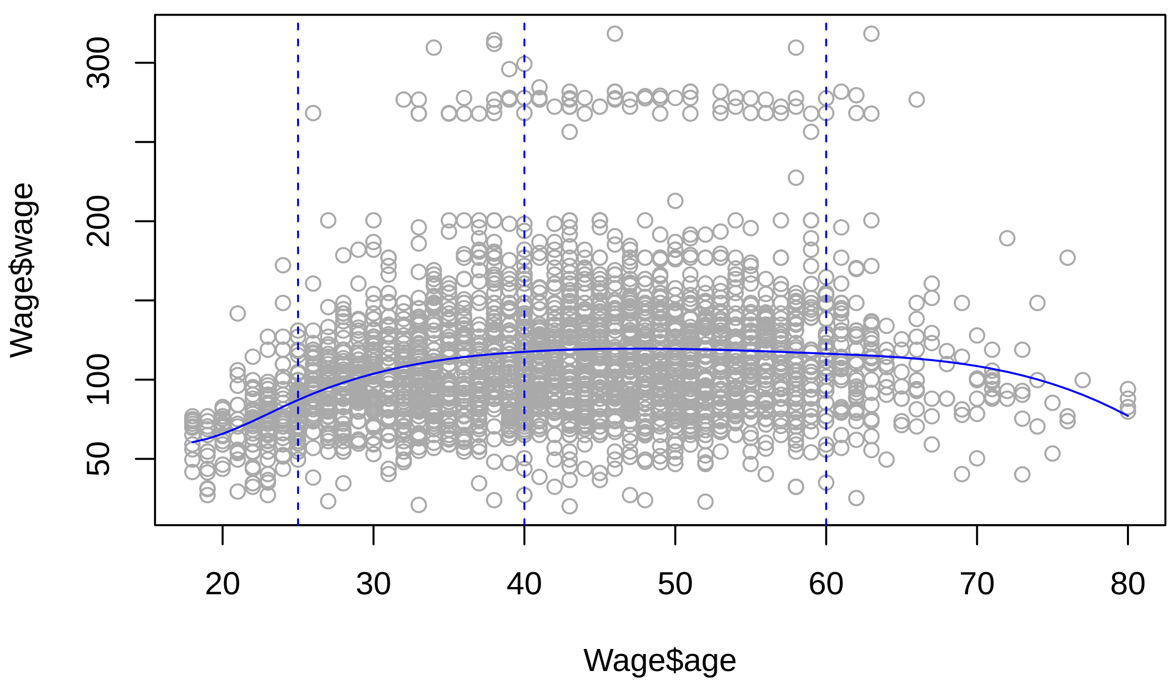
```
prob.bands = (exp(se.bands))/(1+exp(se.bands))
matplot(age.grid, prob.bands, col = "blue", lwd = c(2,1,1), lty = c(1,2,2), type = "l", ylim = c(0,0.1))
points(jitter(Wage$age), I(Wage$wage>250)/10, pch = "|", cex = 0.5)
```



Splines

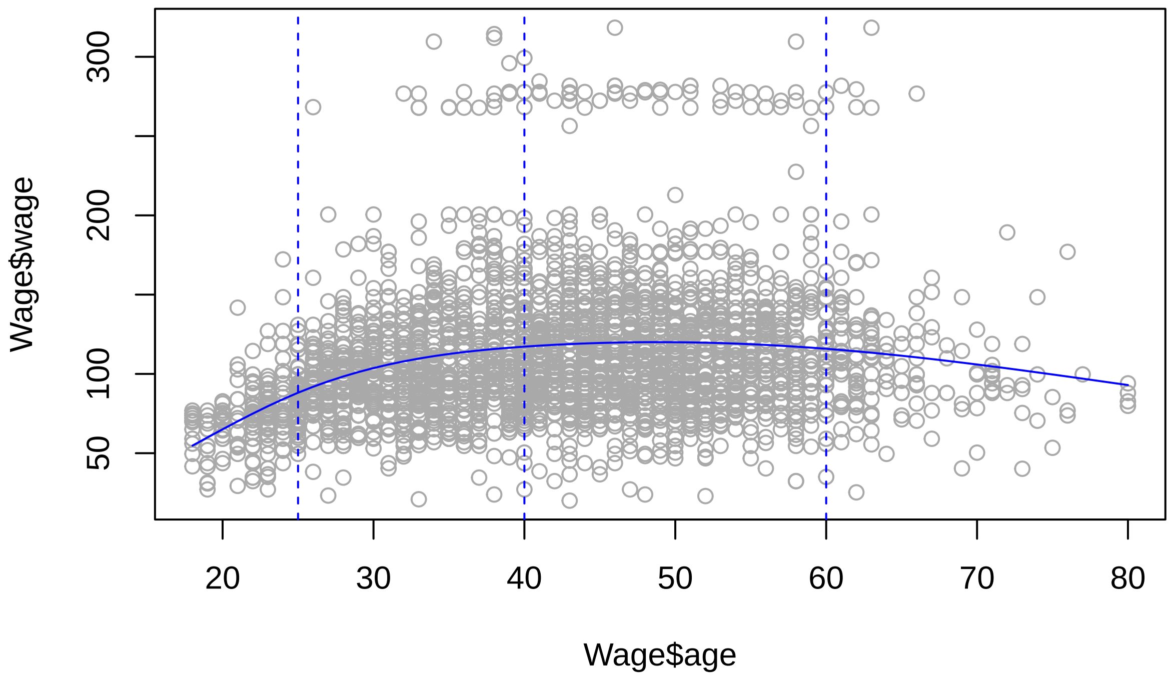
Splines are more flexible than polynomials, but the idea is rather similar.

```
library(splines)
fit = lm(wage ~ bs(age, df = 3, knots = c(25,40,60)), data = Wage)
plot(Wage$age, Wage$wage, col = "darkgrey")
lines(age.grid, predict(fit, list(age = age.grid)), col = "blue")
abline(v=c(25,40,60), lty = 2, col = "blue")
```



Natural Spline

```
library(splines)
fit = lm(wage ~ ns(age, df = 3, knots = c(25,40,60)), data = Wage)
plot(Wage$age, Wage$wage, col = "darkgrey")
lines(age.grid, predict(fit, list(age = age.grid)), col = "blue")
abline(v=c(25,40,60), lty = 2, col = "blue")
```

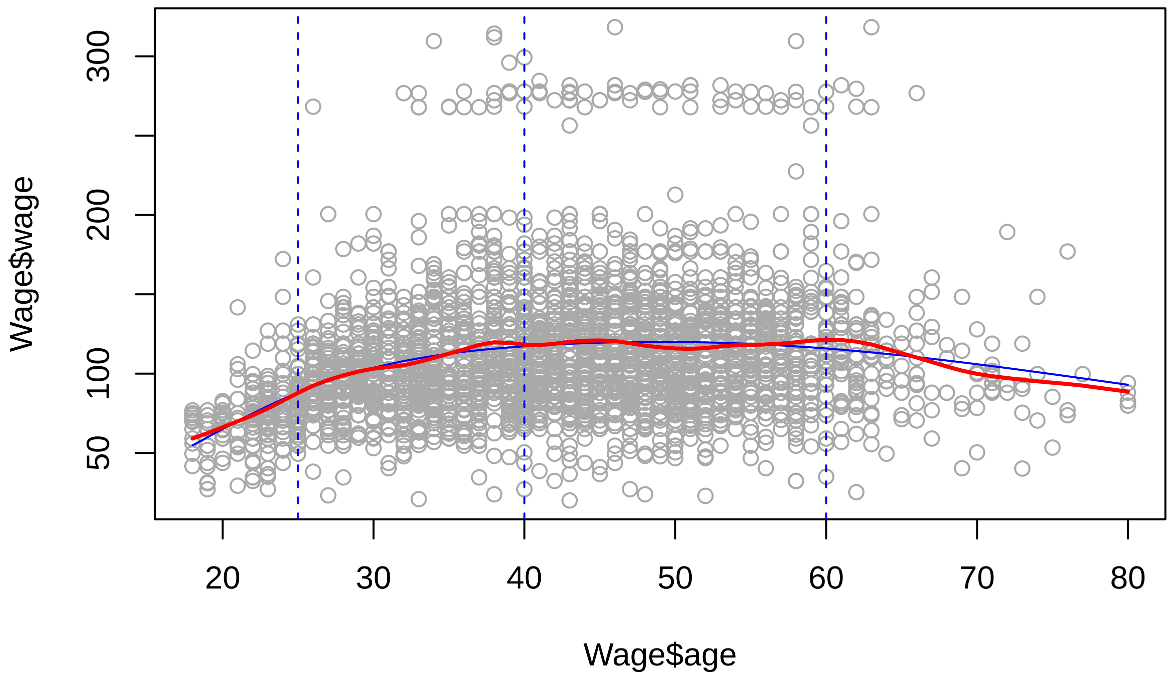


#

smoothing spline It doen't require knot selection, but it dose have a smoothing parameter, which can conviniently be secified via the effective degrees of freedom or "df"

```
plot(Wage$age, Wage$wage, col = "darkgrey")
lines(age.grid, predict(fit, list(age = age.grid)), col = "blue")
abline(v=c(25,40,60), lty = 2, col = "blue")

fit = smooth.spline(Wage$age, Wage$wage, df = 16)
lines(fit,col = "red", lwd = 2)
```



Or we can use LOO cross-validation to select the smoothing parameter for us automatically

```
fit = smooth.spline(Wage$age, Wage$wage, cv = TRUE)
```

```
## Warning in smooth.spline(Wage$age, Wage$wage, cv = TRUE): cross-validation with
## non-unique 'x' values seems doubtful
```

```
fit
```

```
## Call:
## smooth.spline(x = Wage$age, y = Wage$wage, cv = TRUE)
##
## Smoothing Parameter  spar= 0.6988943  lambda= 0.02792303 (12 iterations)
## Equivalent Degrees of Freedom (Df): 6.794596
## Penalized Criterion (RSS): 75215.9
## PRESS(l.o.o. CV): 1593.383
```

Generalized Additive Model

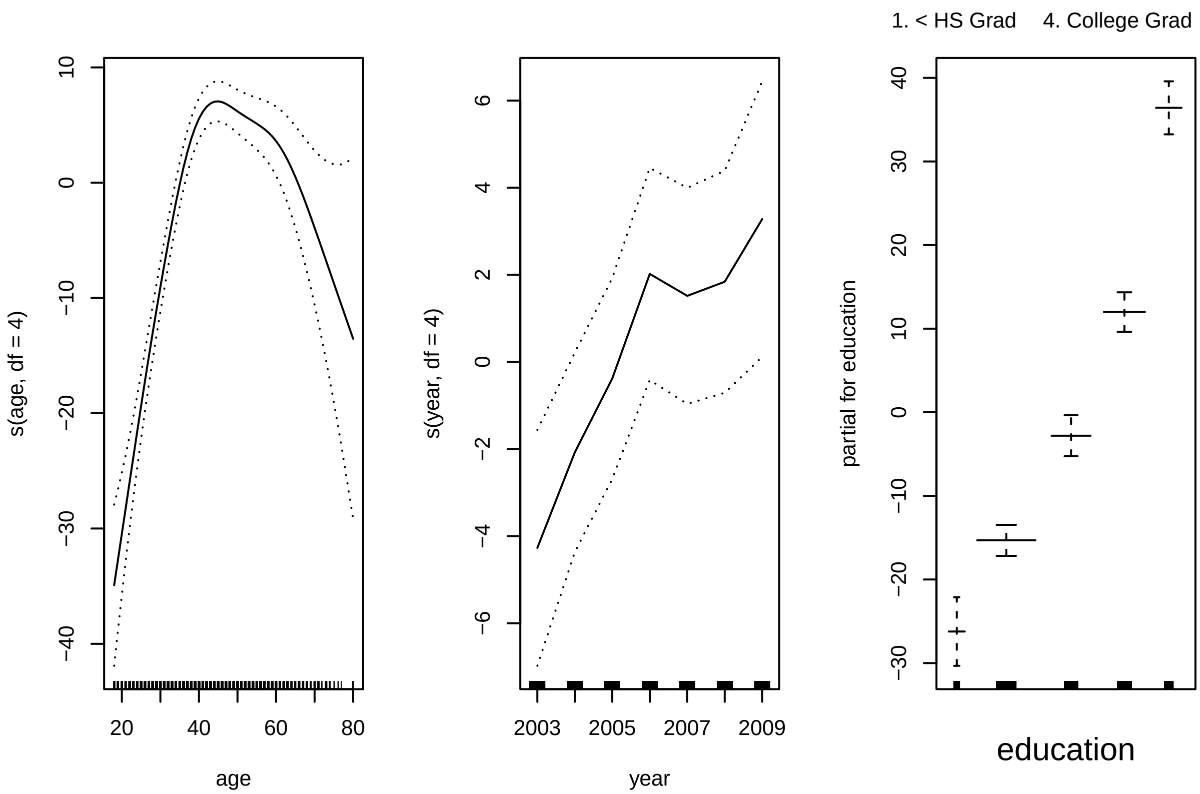
So for we have focused on fitting models with mostly single nonlinear terms. The "gam" package make it easier to work with multiple nonlinear terms. In addition, it knows how to plot these functions and their standard errors.

```
library(gam)
```

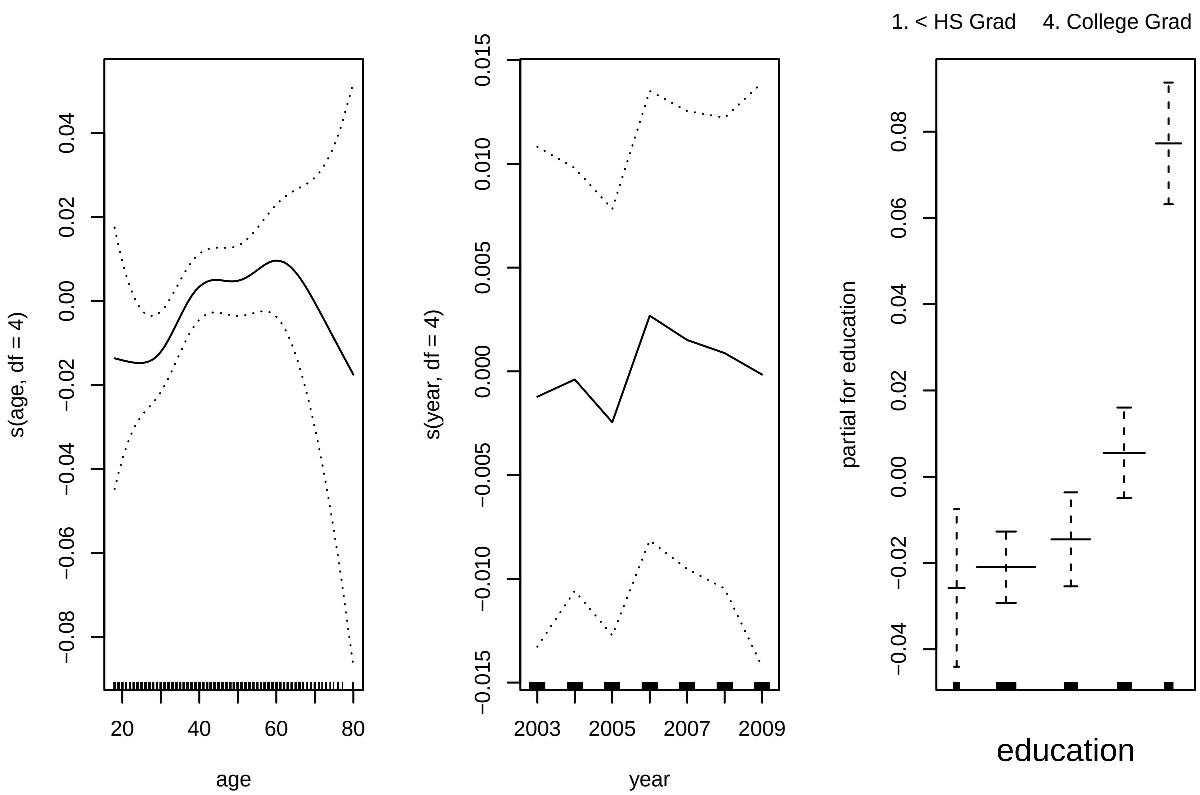
```
## Loading required package: foreach
```

```
## Loaded gam 1.22-1
```

```
gam1 = gam(wage ~ s(age, df =4) + s(year, df = 4)+education, data = Wage)
par(mfrow = c(1,3))
plot(gam1, se = T)
```




```
gam2 = gam(I(wage>250) ~ s(age, df =4) + s(year, df = 4)+education, data = Wage)
plot(gam2, se = T)
```



Let's see if we need a nonlinear term for year

```r
gam2a = gam(I(wage>250) ~ s(age, df =4) + year +education, data = Wage)
anova(gam2, gam2a)
```

```
## Analysis of Deviance Table
##
## Model 1: I(wage > 250) ~ s(age, df = 4) + s(year, df = 4) + education
## Model 2: I(wage > 250) ~ s(age, df = 4) + year + education
##   Resid. Df Resid. Dev Df  Deviance Pr(>Chi)
## 1      2987     73.148
## 2      2990     73.173 -3 -0.025323    0.793
```

```r
AIC(gam2)
```

```
## [1] -2600.023
```

```r
AIC(gam2a)
```

```
## [1] -2604.985
```

One nice feature if the “gam” package is that it knows how to plot the function nicely, even for modes fit by ‘lm’ and "glm’

```r
par(mfrow = c(1,3))
lm1 = lm(wage ~ ns(age, df = 4)+ ns(year, df = 4) + education, data = Wage)
plot.Gam(lm1)
```