Nonlinear Models

Contents

1 Getting Set Up

1.1 Setting chunk options

```
knitr::opts_chunk$set(warning=FALSE, message=FALSE)
knitr::purl("nonlinear-models.Rmd")
```

1.2 Installing Packages

We begin by loading the ISLR library, which contains the data.

```
install.packages('ISLR')
install.packages('gam')
library(ISLR)
attach(Wage)
```

2 Polynomial Regression and Step Functions

We first fit the model using the following command:

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

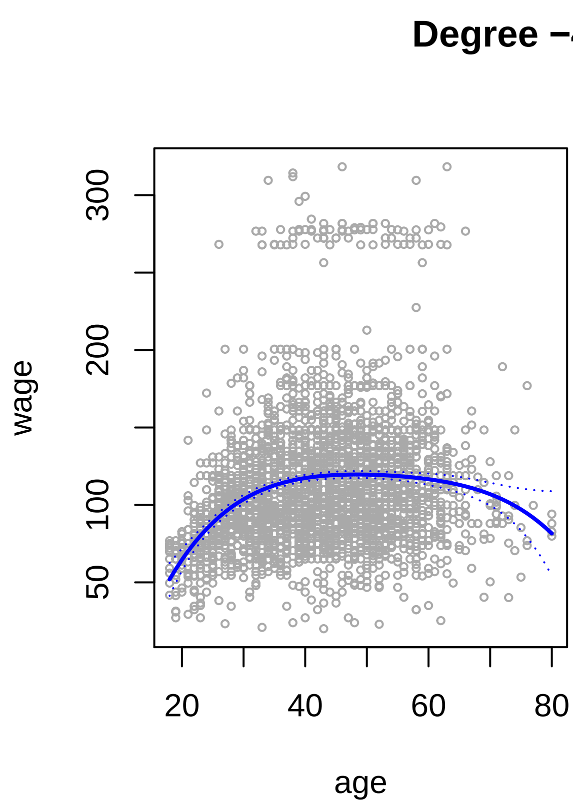
```
##                   Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)    111.70361  0.7287409 153.283015 0.000000e+00
## poly(age, 4)1  447.06785 39.9147851  11.200558 1.484604e-28
## poly(age, 4)2 -478.31581 39.9147851 -11.983424 2.355831e-32
## poly(age, 4)3  125.52169 39.9147851   3.144742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

This syntax fits a linear model, using the lm() function, in order to predict wage using a fourth-degree polynomial in age: poly(age,4). We now create a grid of values for age at which we want predictions, and then call the generic predict() function, specifying that we want standard errors as well.

```
agelims = range(age)
age.grid = seq(from=agelims[1], to=agelims[2])
preds = predict(fit, newdata=list(age=age.grid), se=TRUE)
se.bands = cbind(preds$fit+2*preds$se.fit, preds$fit-2*preds$se.fit)
```

Finally, we plot the data and add the fit from the degree-4 polynomial.

```
par(mfrow=c(1,2), 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)
```



In performing a polynomial regression we must decide on the degree of the polynomial to use. One way to do this is by using hypothesis tests. We now fit models ranging from linear to a degree-5 polynomial and seek to determine the simplest model which is sufficient to explain the relationship between wage and age. We use the anova() function, which performs an analysis of variance (ANOVA, using an F-test) in order to test the null hypothesis that a model M1 is sufficient to explain the data against the variance alternative hypothesis that a more complex model M2 is required. In order to use the anova() function, M1 and M2 must be nested models: the predictors in M1 must be a subset of the predictors in M2. In this case, we fit five different models and sequentially compare the simpler model to the more complex model.

```
fit.1 = lm(wage~age, data=Wage)
fit.2 = lm(wage~poly(age,2), data=Wage)
fit.3 = lm(wage~poly(age,3), data=Wage)
fit.4 = lm(wage~poly(age,4), data=Wage)
fit.5 = lm(wage~poly(age,5), data=Wage)
anova(fit.1, fit.2, fit.3, fit.4, fit.5)
```

```
## 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
```

The p-value comparing the linear Model 1 to the quadratic Model 2 is essentially zero (<10−15), indicating that a linear fit is not sufficient. Similarly the p-value comparing the quadratic Model 2 to the cubic Model 3 is very low (0.0017), so the quadratic fit is also insufficient. The p-value comparing the cubic and degree-4 polynomials, Model 3 and Model 4, is approximately 5 % while the degree-5 polynomial Model 5 seems unnecessary because its p-value is 0.37. Hence, either a cubic or a quartic polynomial appear to provide a reasonable fit to the data, but lower- or higher-order models are not justified.

In this case, instead of using the anova() function, we could have obtained these p-values more succinctly by exploiting the fact that poly() creates orthogonal polynomials.

```
coef(summary(fit.5))
```

```
##                  Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)     111.70361  0.7287647 153.2780243 0.000000e+00
## poly(age, 5)1   447.06785 39.9160847  11.2001930 1.491111e-28
## poly(age, 5)2  -478.31581 39.9160847 -11.9830341 2.367734e-32
## poly(age, 5)3   125.52169 39.9160847   3.1446392 1.679213e-03
## poly(age, 5)4   -77.91118 39.9160847  -1.9518743 5.104623e-02
## poly(age, 5)5   -35.81289 39.9160847  -0.8972045 3.696820e-01
```

Notice that the p-values are the same, and in fact the square of the t-statistics are equal to the F-statistics from the anova() function.

However, the ANOVA method works whether or not we used orthogonal polynomials; it also works when we have other terms in the model as well. For example, we can use anova() to compare these three models:

```r
fit.1 = lm(wage~education + age, data=Wage)
fit.2 = lm(wage~education + poly(age,2), data=Wage)
fit.3 = lm(wage~education + poly(age,3), data=Wage)
anova(fit.1, fit.2, fit.3)
```

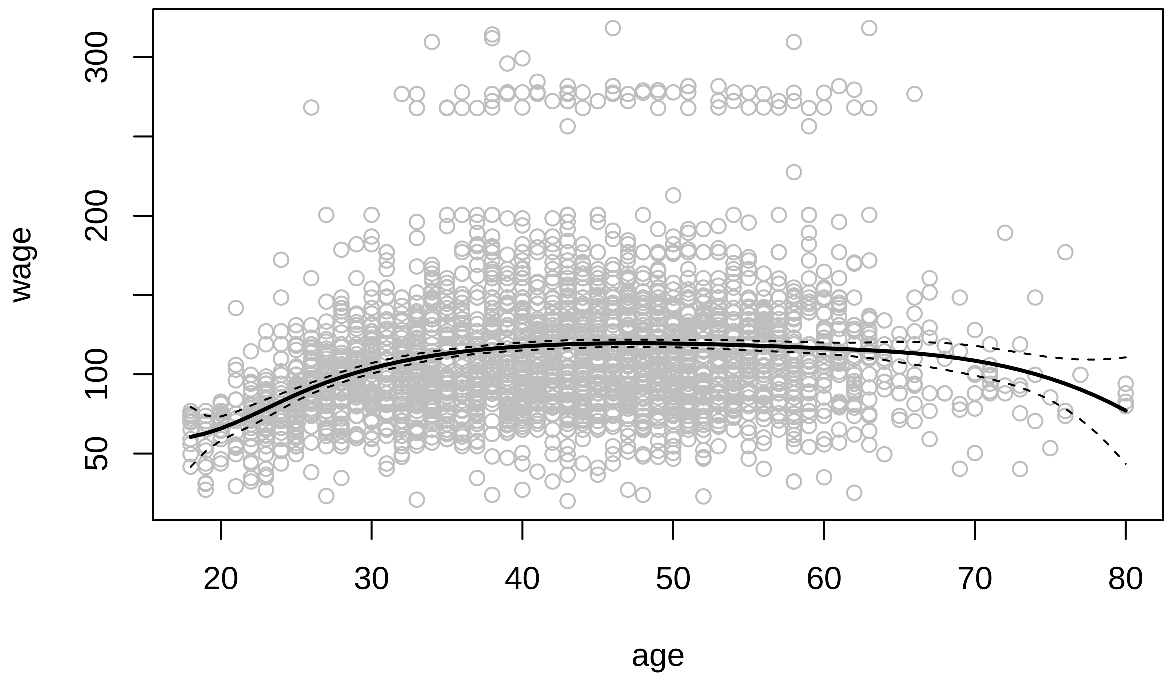
```
## Analysis of Variance Table
##
## Model 1: wage ~ education + age
## Model 2: wage ~ education + poly(age, 2)
## Model 3: wage ~ education + poly(age, 3)
##   Res.Df     RSS Df Sum of Sq        F Pr(>F)
## 1   2994 3867992
## 2   2993 3725395  1    142597 114.6969 <2e-16 ***
## 3   2992 3719809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

As an alternative to using hypothesis tests and ANOVA, we could choose the polynomial degree using cross-validation, as discussed in Chapter 5.

3 Splines

In order to fit regression splines in R, we use the splines library. The bs() function generates the entire matrix of basis functions for splines with the specified set of knots. By default, cubic splines are produced. Fitting wage to age using a regression spline is simple:

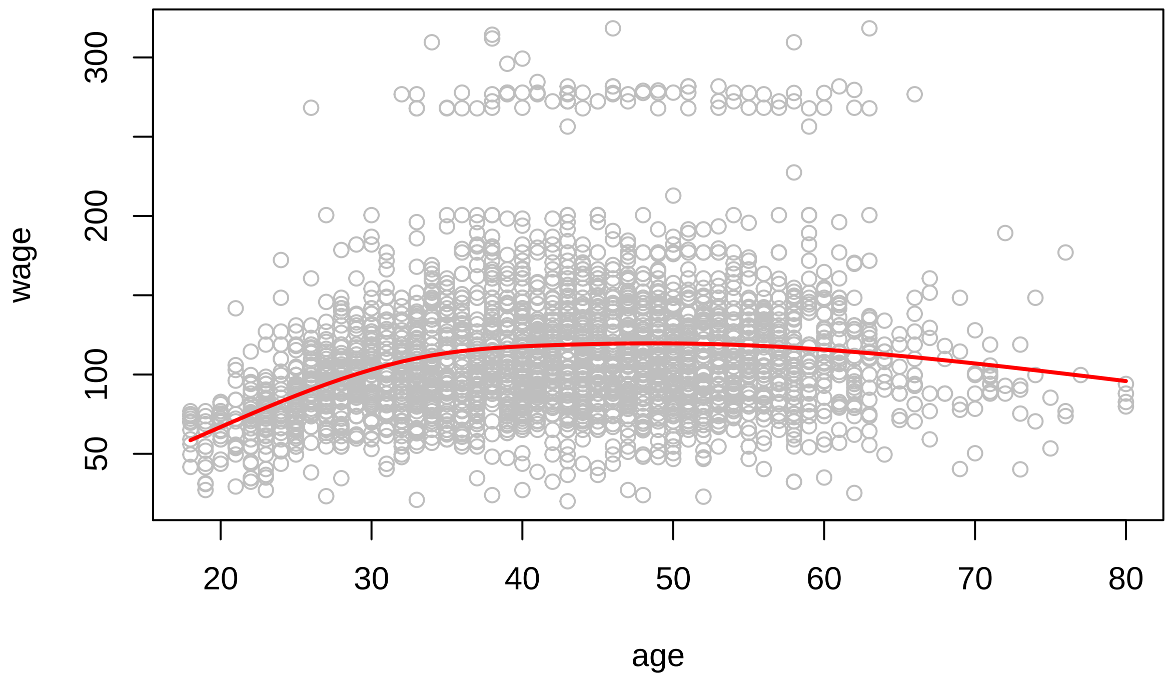
```r
library(splines)
fit = lm(wage~bs(age, knots=c(25,40,60)), data=Wage)
pred = predict(fit, newdata=list(age=age.grid), se=T)
plot(age, wage, col="gray")
lines(age.grid,pred$fit, lwd=2)
lines(age.grid,pred$fit+2*pred$se, lty="dashed")
lines(age.grid,pred$fit-2*pred$se, lty="dashed")
```



Here we have prespecified knots at ages 25, 40, and 60. This produces a spline with six basis functions. (Recall that a cubic spline with three knots has seven degrees of freedom; these degrees of freedom are used up by an intercept, plus six basis functions.) We could also use the df option to produce a spline with knots at uniform quantiles of the data.

In order to instead fit a natural spline, we use the ns() function. Here we fit a natural spline with 4 degrees of freedom.

```
fit2 = lm(wage~ns(age, df=4), data=Wage)
pred2 = predict(fit2, newdata=list(age=age.grid), se=T)
plot(age, wage, col="gray")
lines(age.grid, pred2$fit, col="red", lwd=2)
```

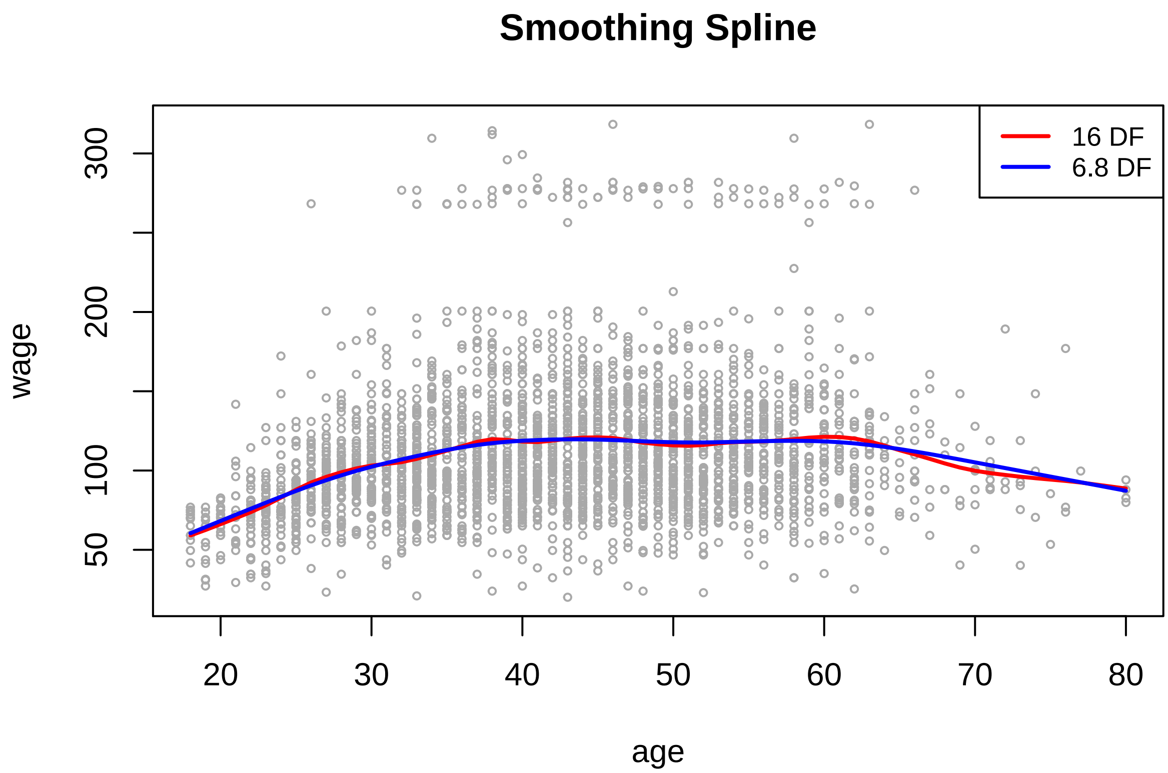


In order to fit a smoothing spline, we use the smooth.spline() function.

```
plot(age, wage, xlim=agelims, cex=.5, col="darkgrey")
title("Smoothing Spline")
fit = smooth.spline(age, wage, df=16)
fit2 = smooth.spline(age, wage, cv=TRUE)
fit2$df
```

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

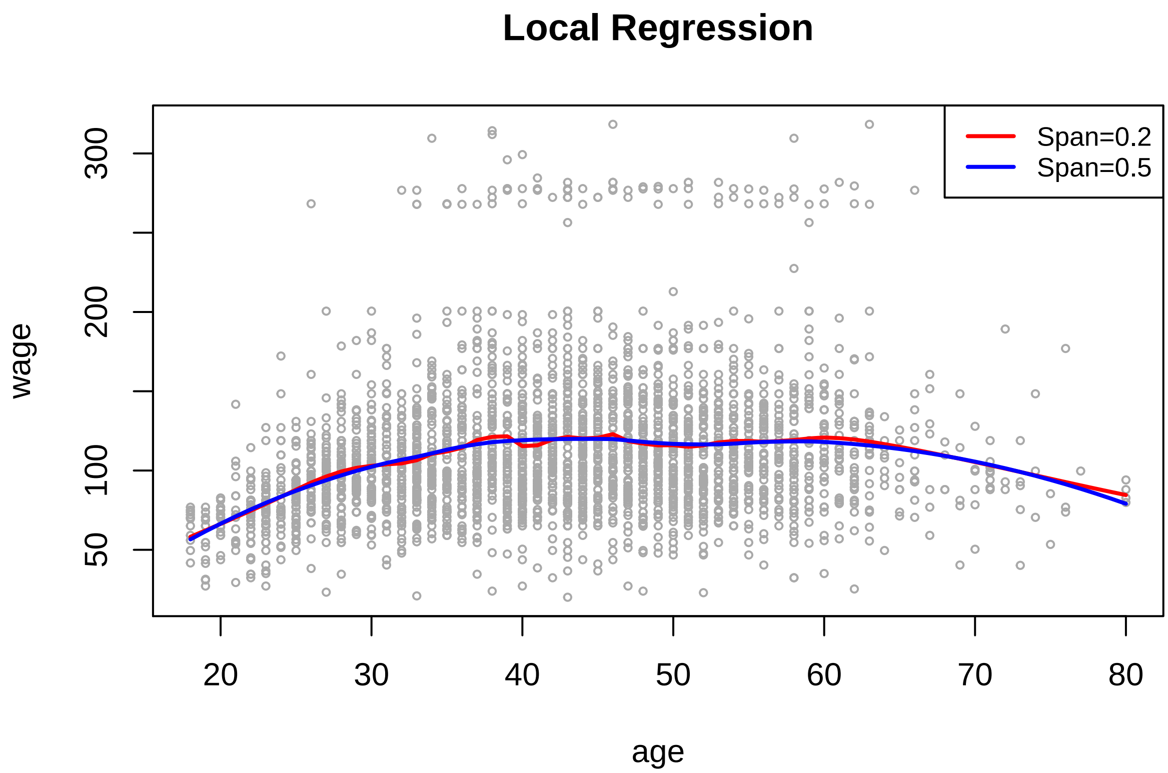
```
lines(fit, col="red", lwd=2)
lines(fit2, col="blue", lwd=2)
legend("topright", legend=c("16 DF","6.8 DF"),
col=c("red","blue"), lty=1, lwd=2, cex=.8)
```



Notice that in the first call to smooth.spline(), we specified df=16. The function then determines which value of leads to 16 degrees of freedom. In the second call to smooth.spline(), we select the smoothness level by cross validation; this results in a value of that yields 6.8 degrees of freedom.

In order to perform local regression, we use the loess() function.

```
plot(age, wage, xlim=agelims, cex=.5, col="darkgrey")
title("Local Regression")
fit = loess(wage~age, span=.2, data=Wage)
fit2 = loess(wage~age, span=.5, data=Wage)
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)
```



Here we have performed local linear regression using spans of 0.2 and 0.5: that is, each neighborhood consists of 20 % or 50 % of the observations. The larger the span, the smoother the fit.

4 GAMs

We now fit a GAM to predict wage using natural spline functions of year and age, treating education as a qualitative predictor. Since this is just a big linear regression model using an appropriate choice of basis functions, we can simply do this using the lm() function.

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

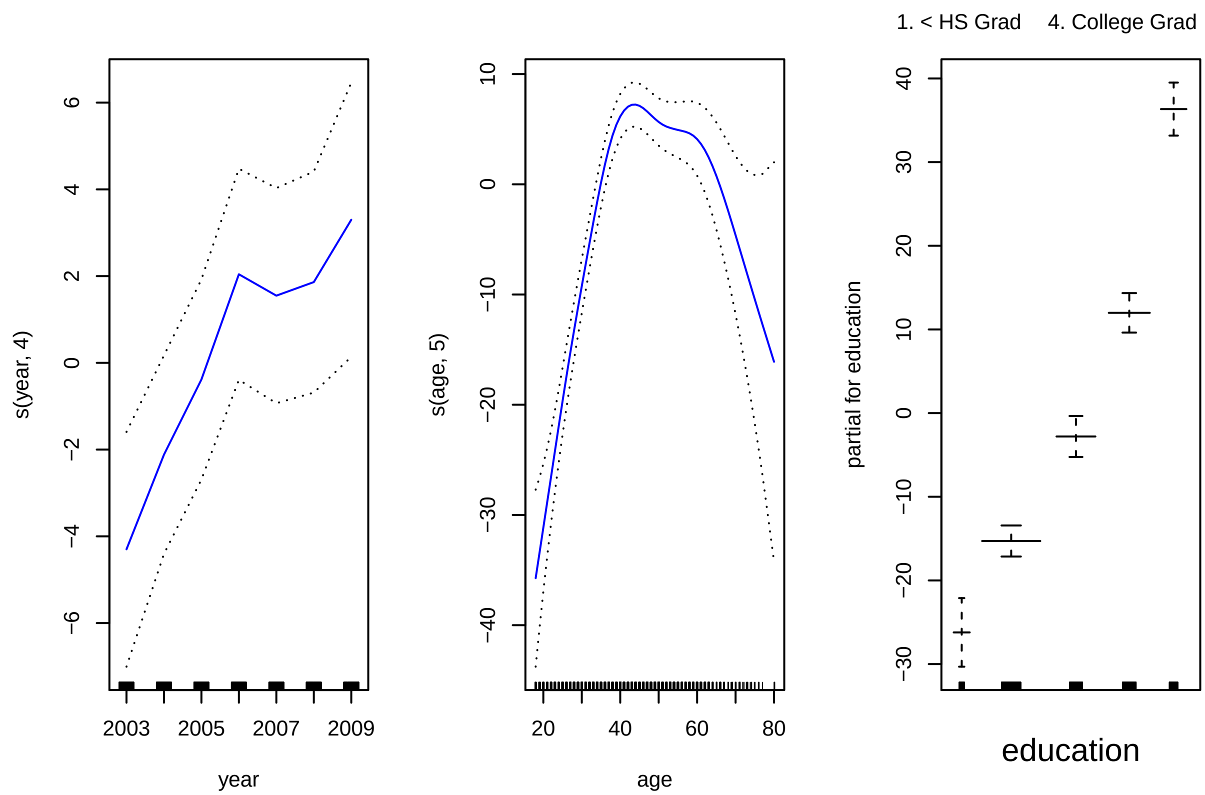
We now fit the model using smoothing splines rather than natural splines. In order to fit more general sorts of GAMs, using smoothing splines or other components that cannot be expressed in terms of basis functions and then fit using least squares regression, we will need to use the gam library in R.

The s() function, which is part of the gam library, is used to indicate that we would like to use a smoothing spline. We specify that the function of year should have 4 degrees of freedom, and that the function of age will have 5 degrees of freedom. Since education is qualitative, we leave it as is, and it is converted into four dummy variables. We use the gam() function in order to fit a GAM using these components. All of the terms in are fit simultaneously, taking each other into account to explain the response.

```
library(gam)
gam.m3 = gam(wage~s(year, 4) + s(age, 5) + education, data=Wage)
```

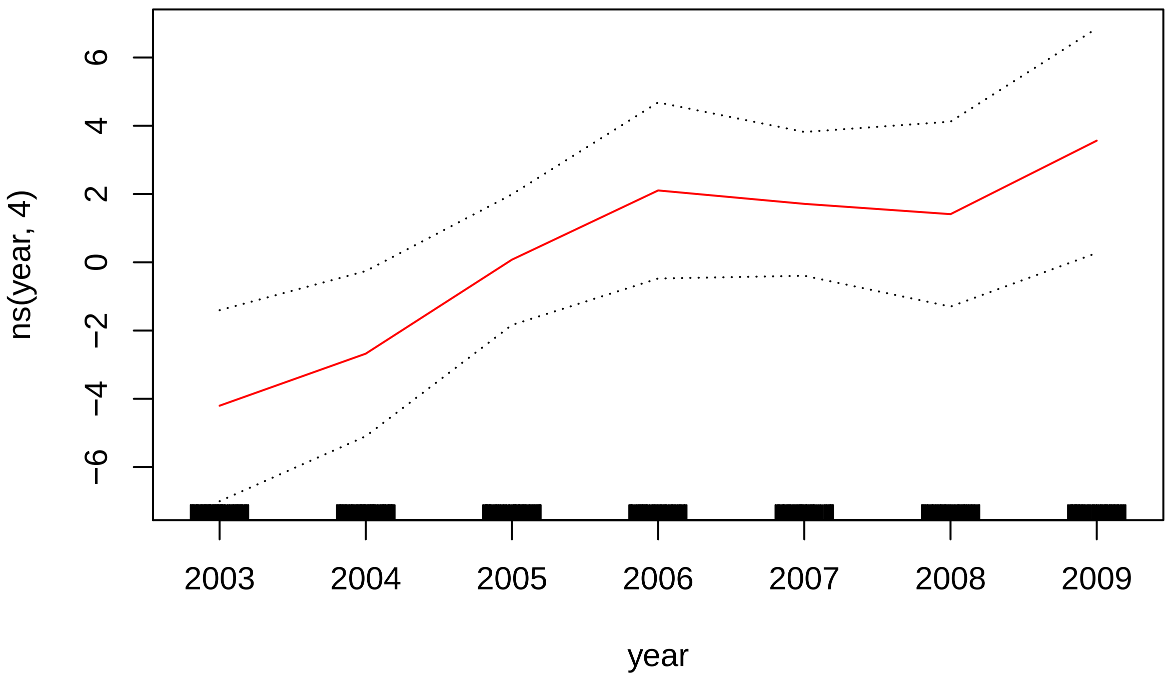
We then call the plot function.

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

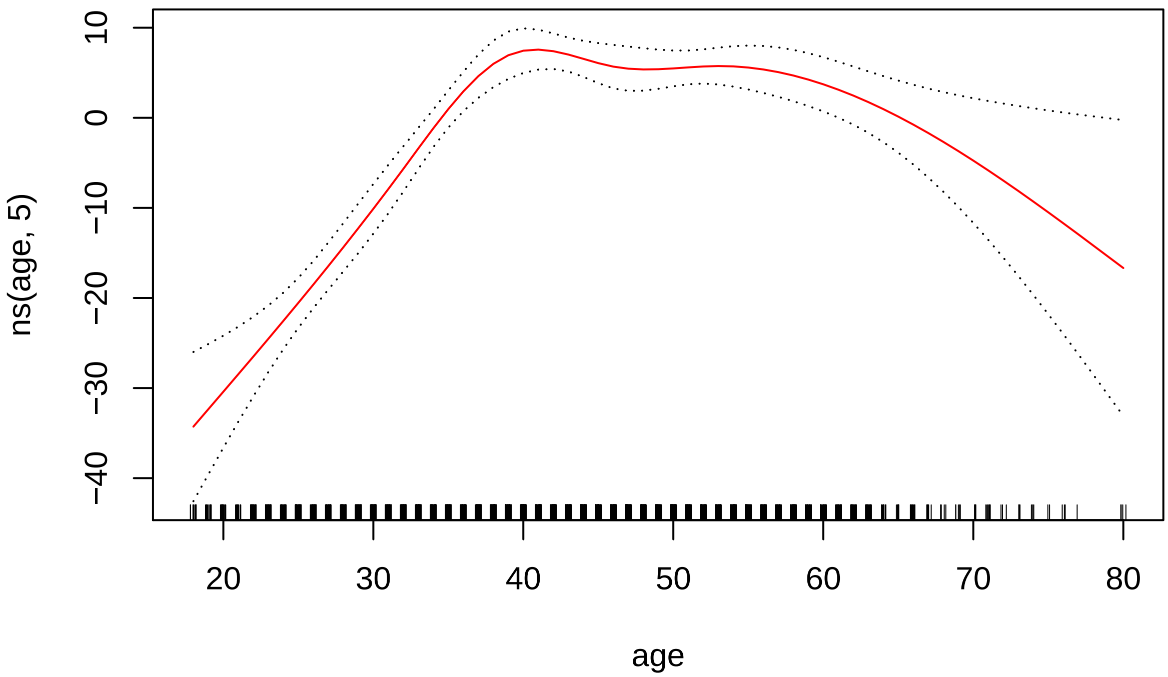


The generic plot() function recognizes that gam.m3 is an object of class gam, and invokes the appropriate plot.gam() method. Conveniently, even though gam1 is not of class gam but of class lm, we can still use plot.gam() on it.

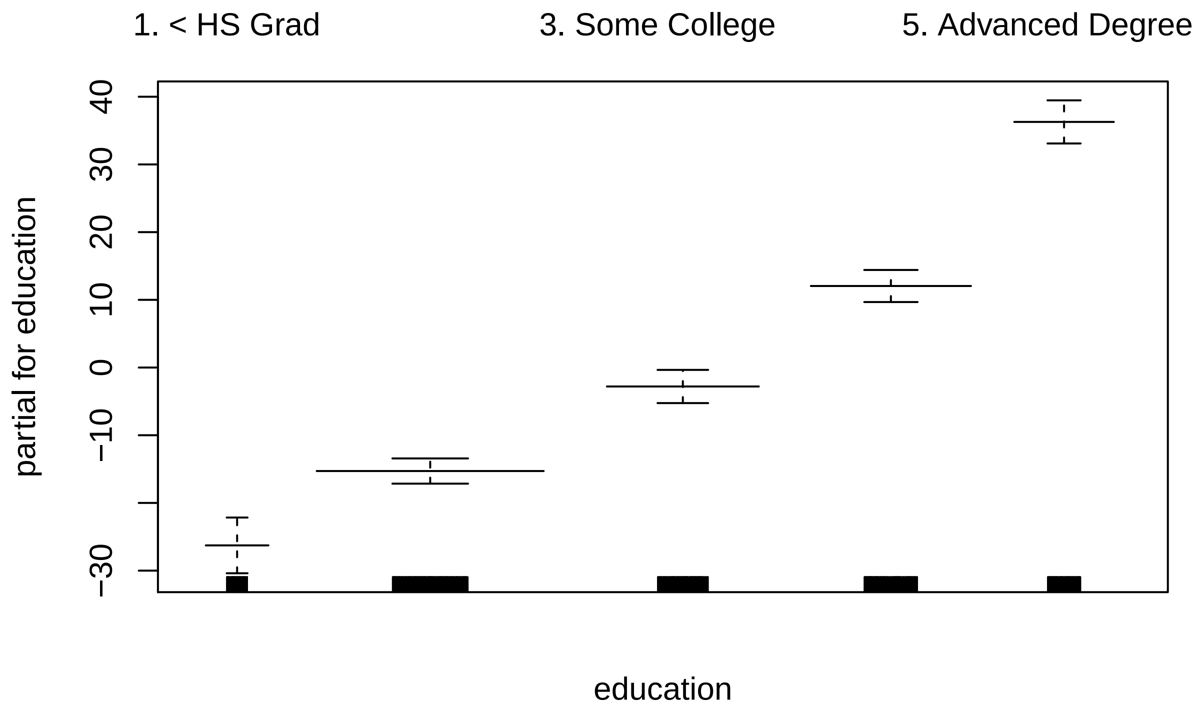
```r
plot.Gam(gam1, se=TRUE, col="red")
```



6
4
2
0
−2
−4
−6
ns(year, 4)
2003
2004
2005
2006
2007
2008
2009
year



10
0
−10
−20
−30
−40
ns(age, 5)
20
30
40
50
60
70
80
age



In these plots, the function of year looks rather linear. We can perform a series of ANOVA tests in order to determine which of these three models is best: a GAM that excludes year (M1), a GAM that uses a linear function of year (M2), or a GAM that uses a spline function of year (M3).

```
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
```

We find that there is compelling evidence that a GAM with a linear function of year is better than a GAM that does not include year at all (p-value = 0.00014). However, there is no evidence that a non-linear function of year is needed (p-value = 0.349). In other words, based on the results of this ANOVA, M2 is preferred.

The summary() function produces a summary of the gam fit.

```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: 2
##
## 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
```

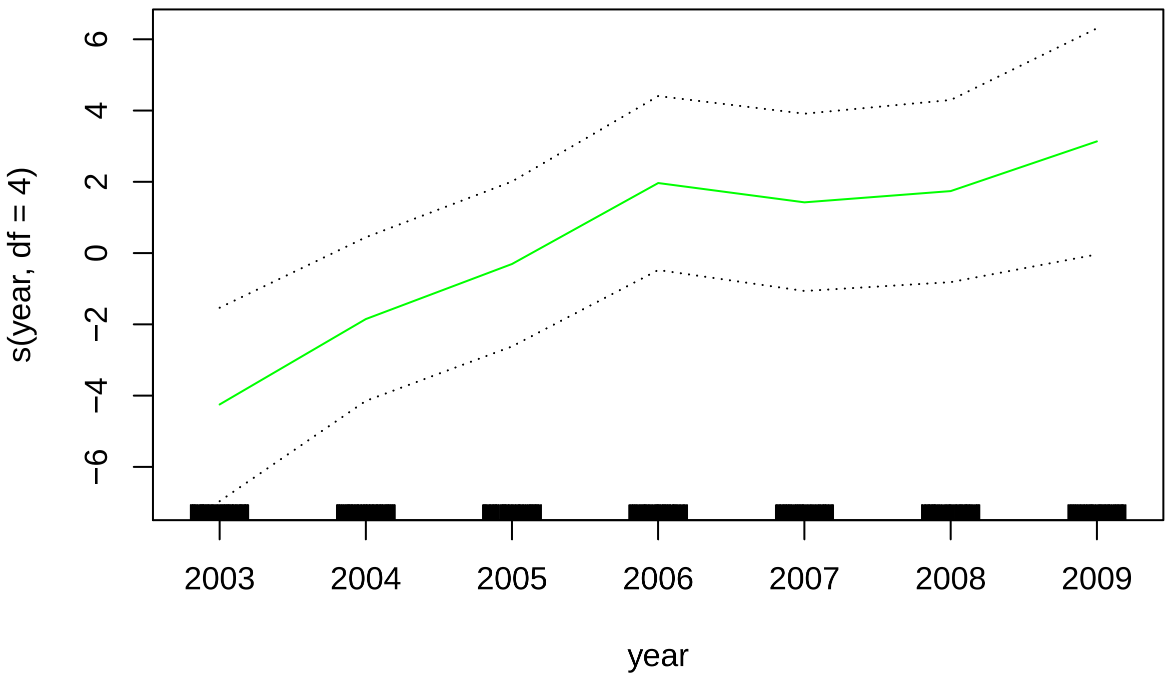
The p-values for year and age correspond to a null hypothesis of a linear relationship versus the alternative of a non-linear relationship. The large p-value for year reinforces our conclusion from the ANOVA test that a linear function is adequate for this term. However, there is very clear evidence that a non-linear term is required for age.

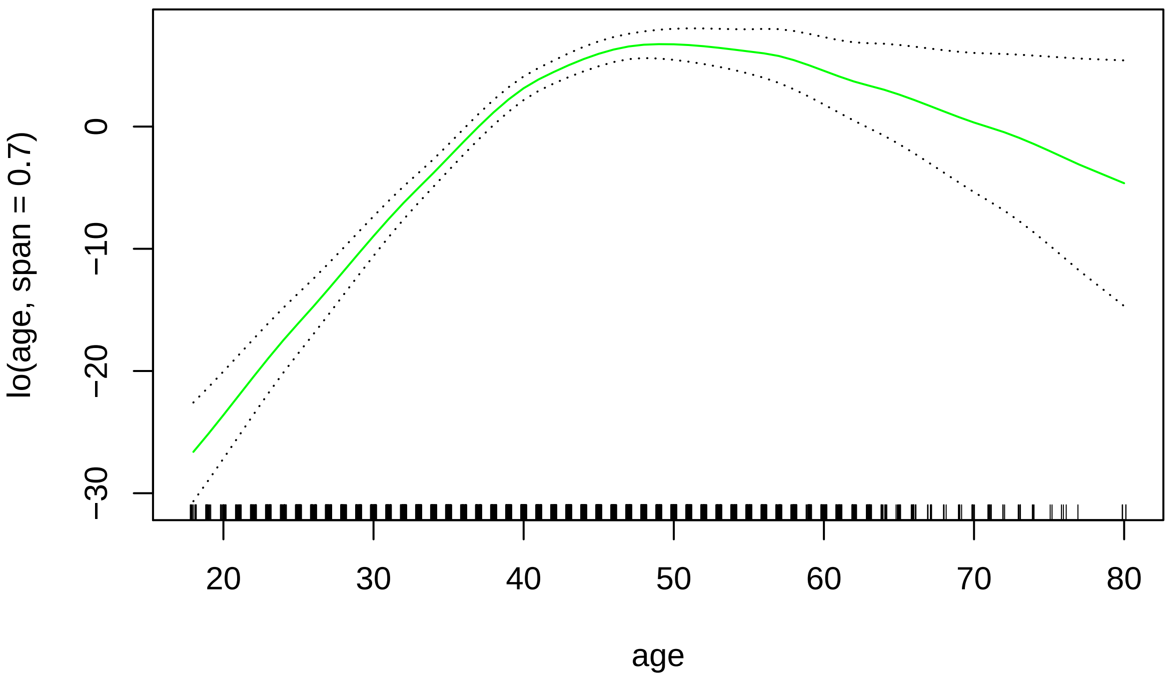
We can make predictions from gam objects, just like from lm objects, using the predict() method for the class gam. Here we make predictions on the training set.

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

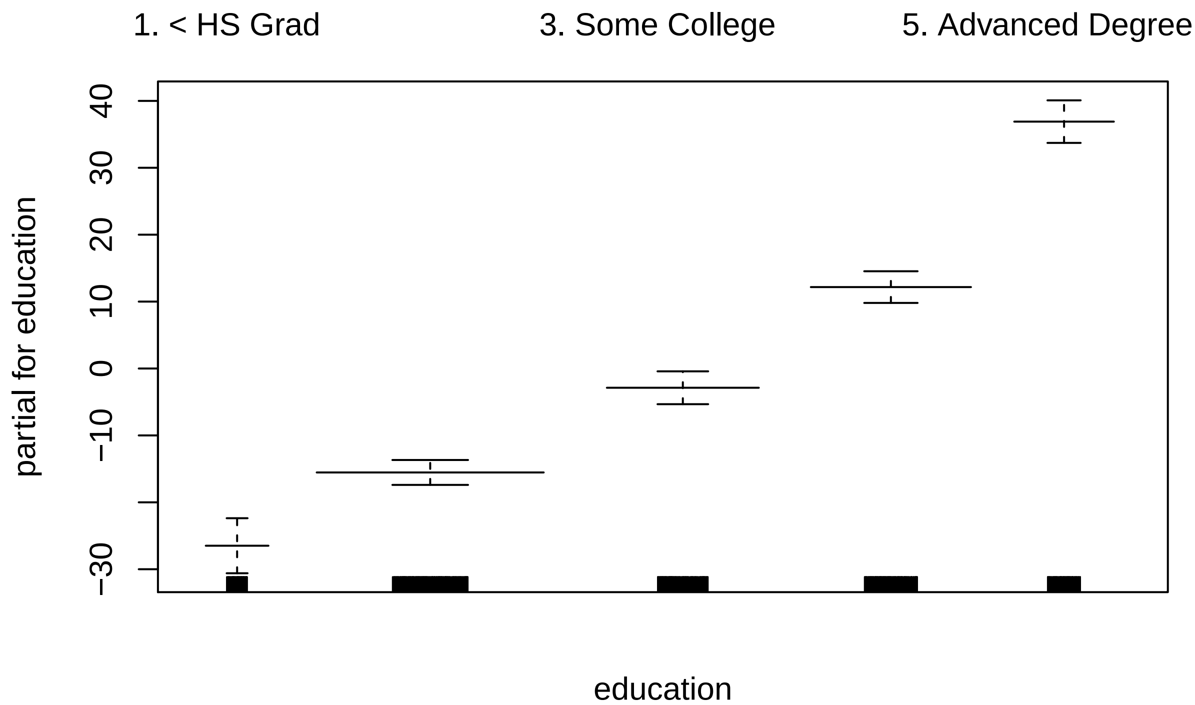
We can also use local regression fits as building blocks in a GAM, using the lo() function.

```r
gam.lo = gam(wage~s(year, df=4) + lo(age, span=0.7) + education, data=Wage)
plot.Gam(gam.lo, se=TRUE, col ="green")
```


s(year, df = 4)
6
4
2
0
−2
−4
−6
2003
2004
2005
2006
2007
2008
2009
year



lo(age, span = 0.7)
0
−10
−20
−30
20
30
40
50
60
70
80
age


1. < HS Grad
3. Some College
5. Advanced Degree
partial for education
40
30
20
10
0
−10
−30
education