statLearn 7.8 R Lab: NON-LINEAR MODELS

Load ISLR library and attach Wage dataset

```
library(ISLR)
attach(Wage)
```

7.8.1 POLYNOMIAL REGRESSION

First, use polynomials, focusing on single predictor: age

```
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
```

Usually, we are not interested in the coefficients, but the fitted functions that they produce.

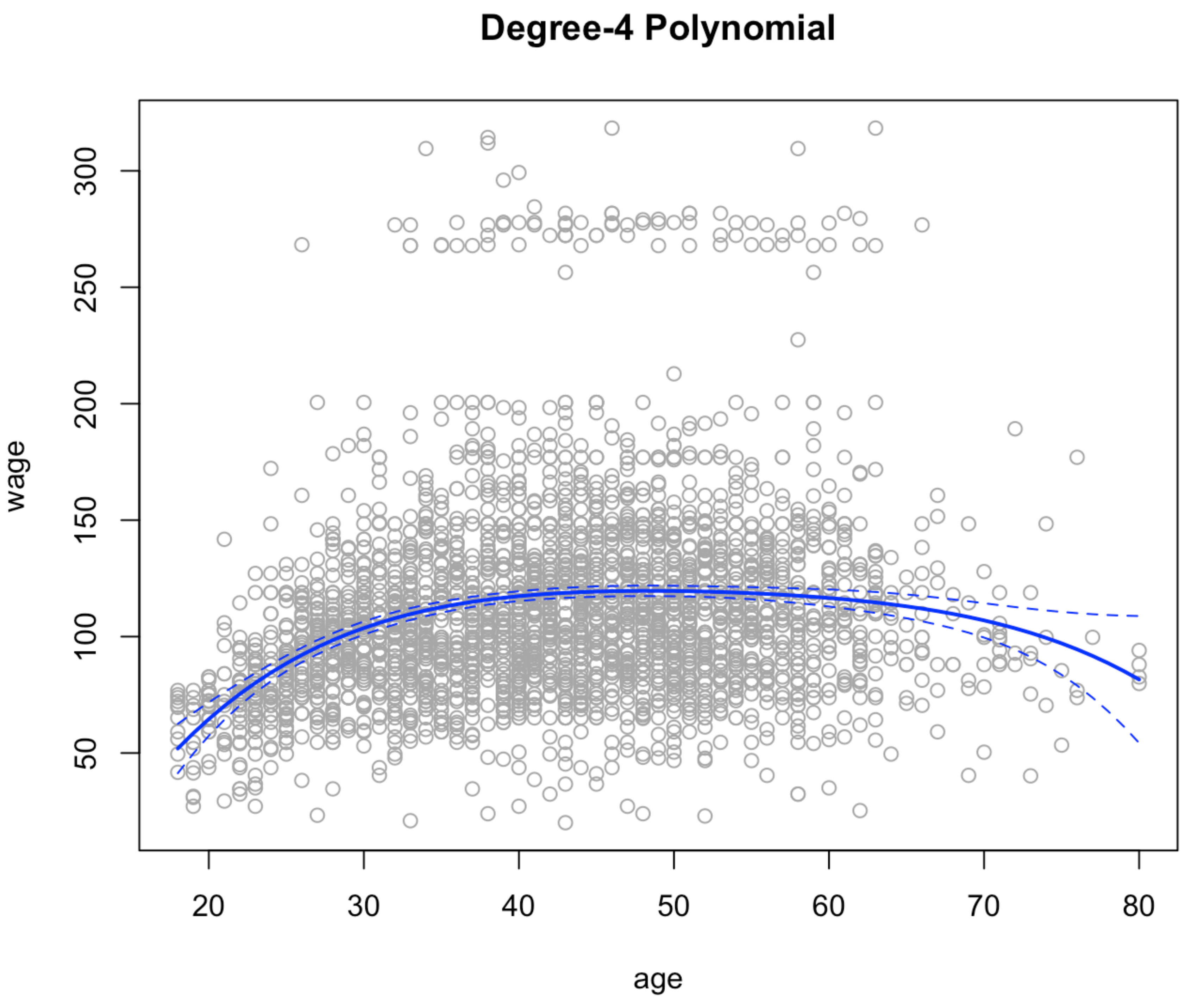
Next, construct plot of fitted functions, along with SEs of fit. Set plot frame parameters for R

Get range of ages; use seq to make grid of values for age. Generate predictions for fit, create se bands, generate plot. Then add linesadd lines for fit model and SEs above, below.

```
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)

plot(age, wage, col="darkgrey")
lines(age.grid, preds$fit, lwd=2, col="blue")
matlines(age.grid, se.bands, col="blue", lty=2)
title("Degree-4 Polynomial")
```

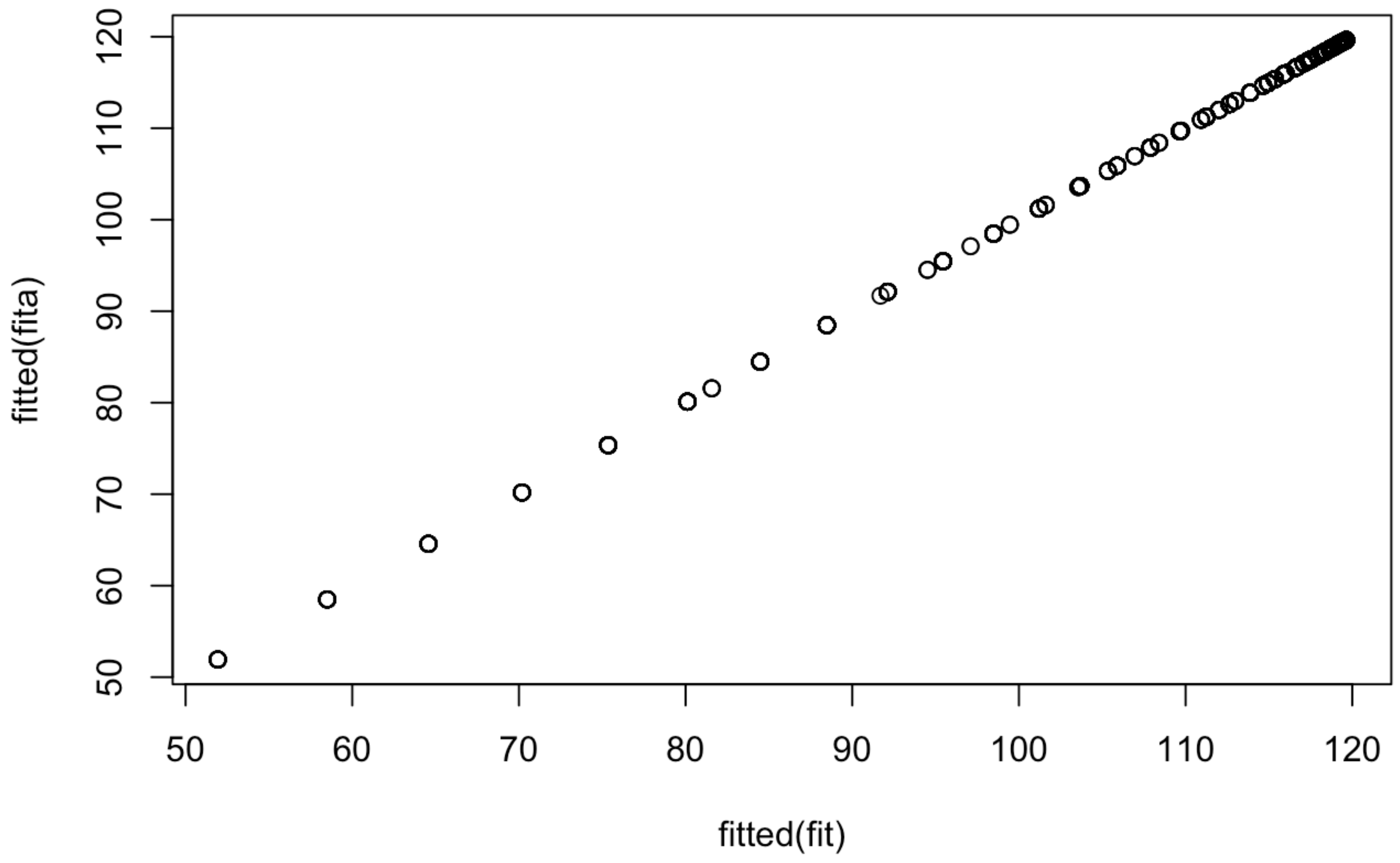


More direct way to fit polynomials in R, using different basis: Here, “I()” is ‘wrapper’ identity function to protect ‘I(age^2)’ Coefficients are different than before, but the fit is same.

```
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
```

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



Using orthogonal polynomials this way, we can separately test for each coefficient. Looking at summary, we see the linear, quadratic, cubic terms are significant, but not the quartic.

This only works with linear regression, and a single predictor. In general, we test differences between models using 'anova()' with models that are nested one within the other

```
fit.a = lm(wage~age,data=Wage)
fit.b = lm(wage~poly(age,2),data=Wage)
fit.c = lm(wage~poly(age,3),data=Wage)
fit.d = lm(wage~poly(age,4),data=Wage)
anova(fit.a,fit.b,fit.c,fit.d)
```

```
## 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)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.6025 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8894  0.001679 **
## 4   2995 4771604  1      6070   3.8101  0.051039 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

POLYNOMIAL LOGISTIC REGRESSION

Fit logistic regression model to binary response variable; constructed from ‘wage’; code big earners (‘>250’) as 1, else 0. Fit model, make prediction, create matrix of fit, upper, lower SE

```
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.fit)
se.bands[1:5,]
```

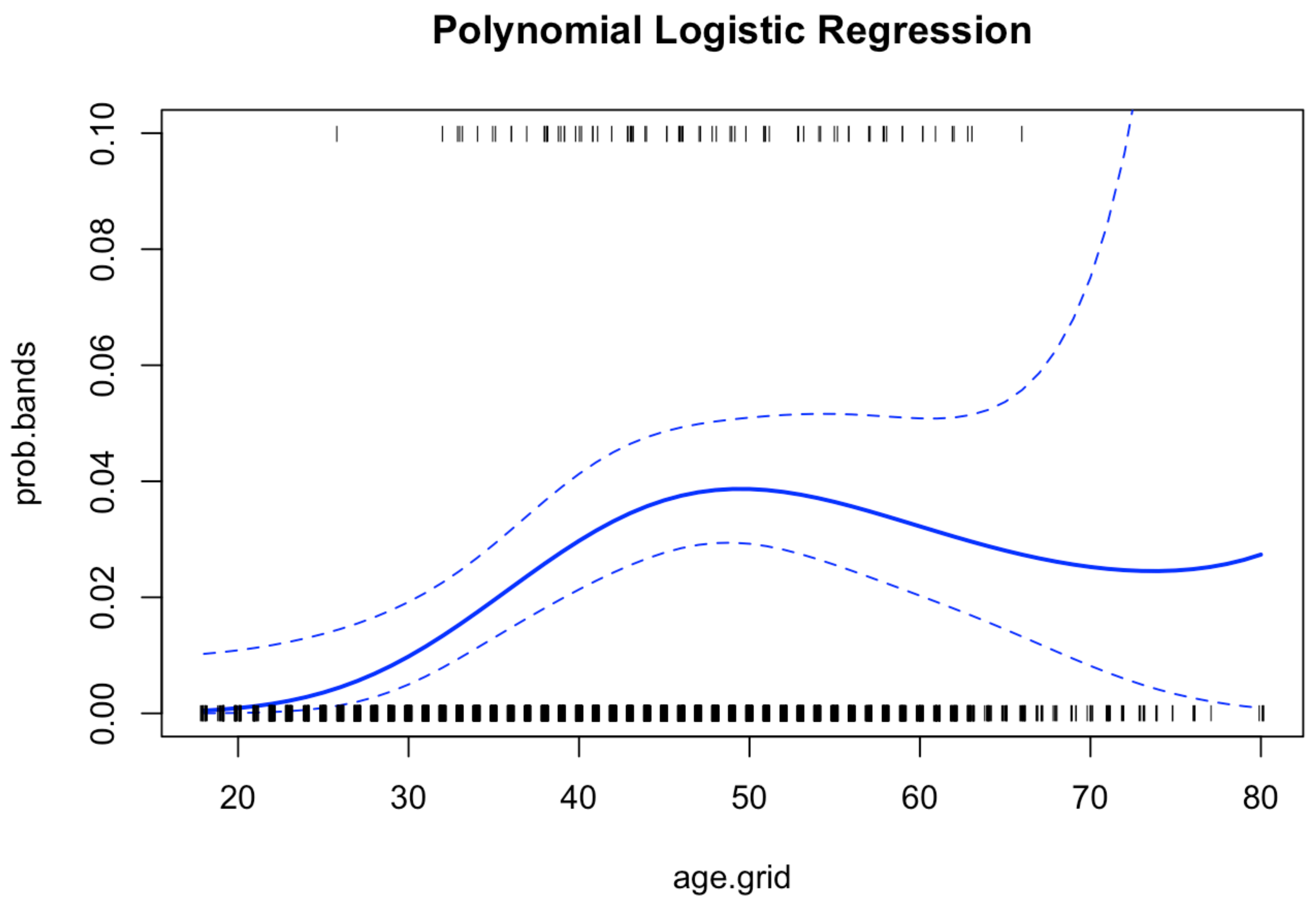
```
##         fit      lower     upper
## 1 -7.664756 -10.759826 -4.569686
## 2 -7.324776 -10.106699 -4.542852
## 3 -7.001732  -9.492821 -4.510643
## 4 -6.695229  -8.917158 -4.473300
## 5 -6.404868  -8.378691 -4.431045
```

Predict fits on logit scale, but we are usually interested in predictions on probability scale; therefore, to transform we apply inverse logit mapping: Markdown interprets TeX expression

$$p = \frac{e^\eta}{1 + e^\eta}.$$

We can do this simultaneously for all three columns of se.bands

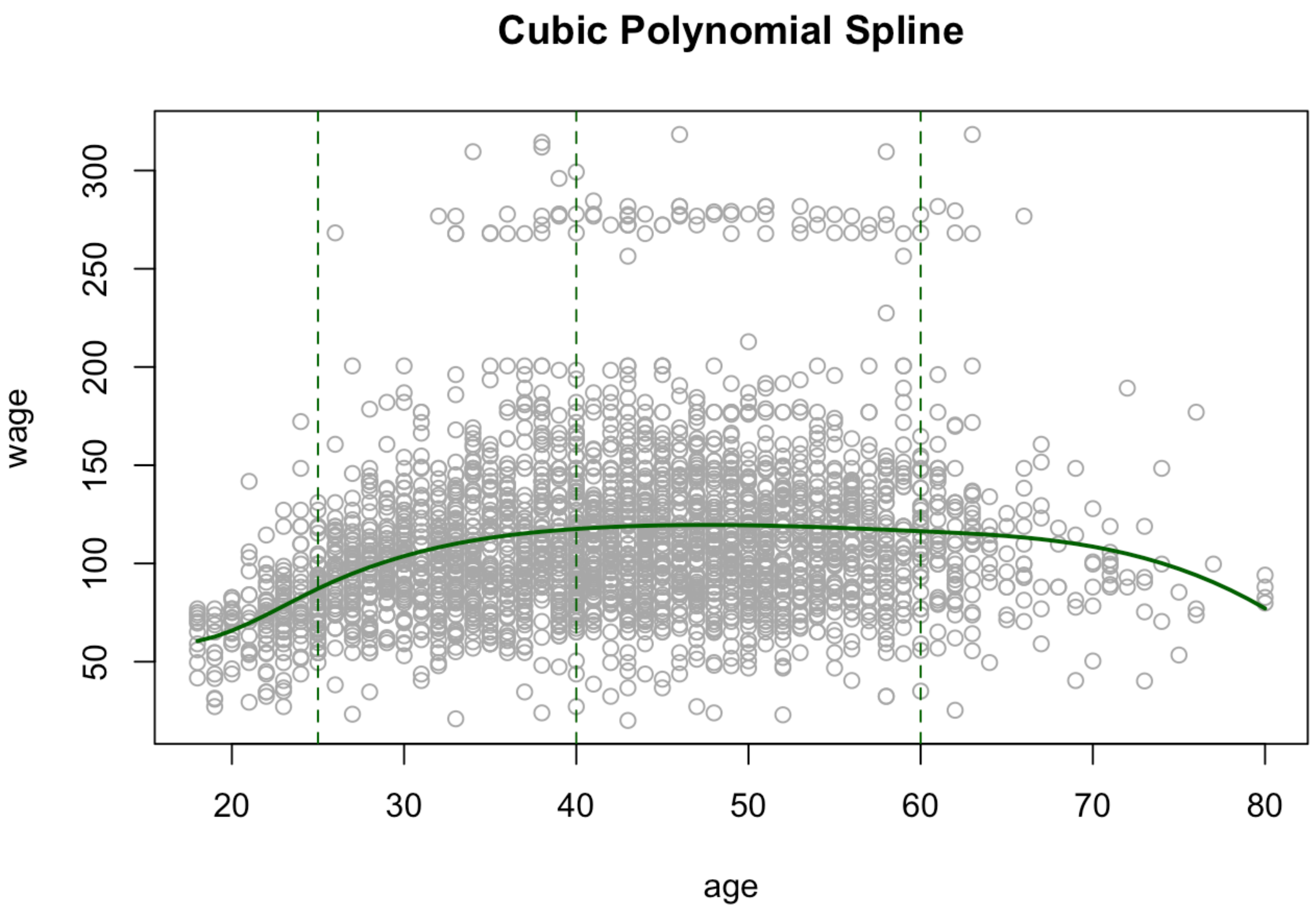
```
prob.bands = exp(se.bands) / (1+exp(se.bands))
matplot(age.grid, prob.bands, col="blue", lwd=c(2,1,1), lty=c(1,2,2), type="l", ylim=
c(0,.1))
points(jitter(age), I(wage>250)/10, pch="|", cex=0.5)
title("Polynomial Logistic Regression")
```



R.Lab 7.8.2 SPLINES

Splines are more flexible than polynomials, but similar idea. Explore cubic splines: setting the number of knots, joints

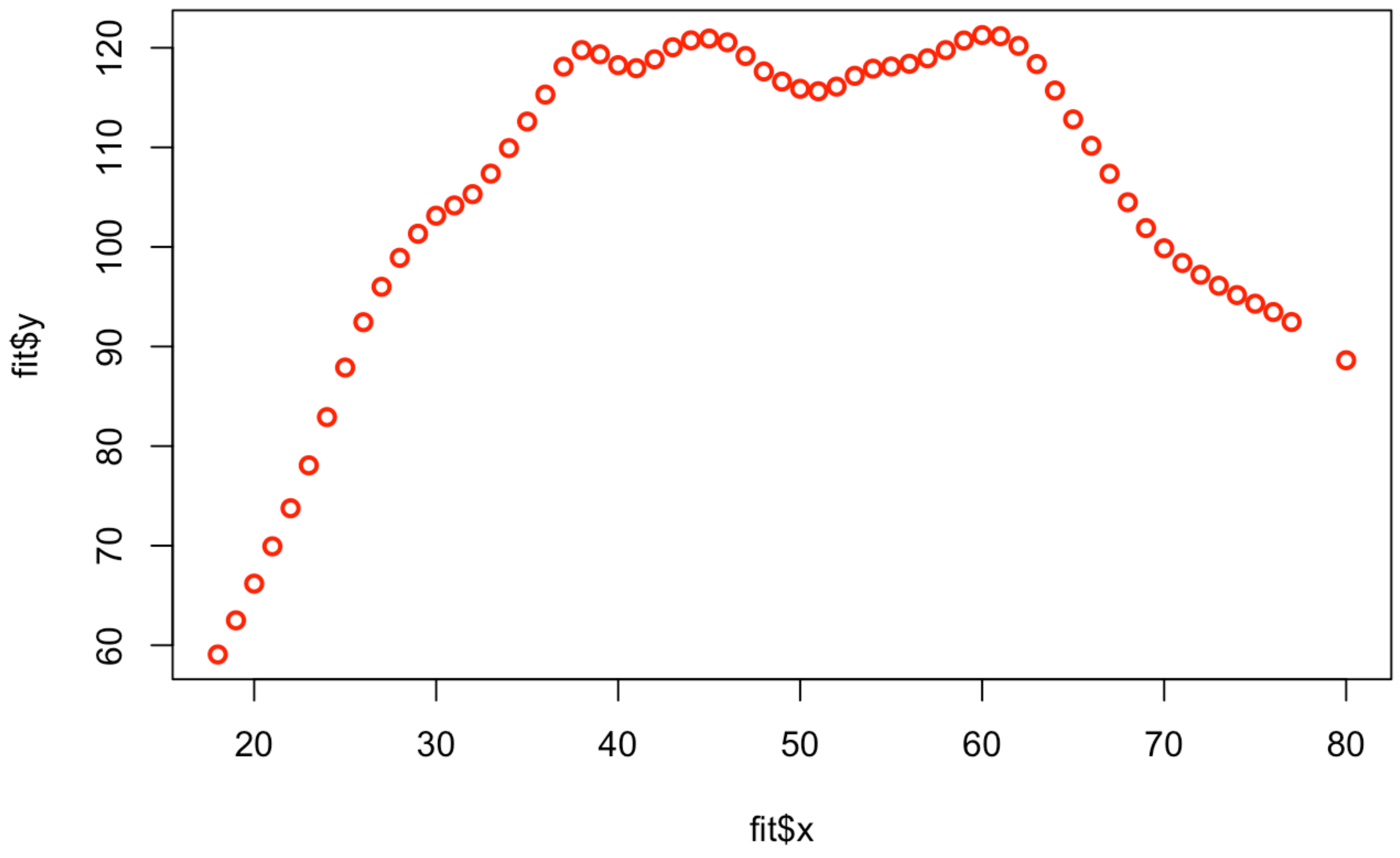
```
library(splines)
fit = lm(wage~bs(age, knots=c(25, 40, 60)), data=Wage)
plot(age, wage, col="darkgrey")
lines(age.grid, predict(fit, list(age=age.grid)), col="darkgreen", lwd=2)
abline(v=c(25, 40, 60), lty=2, col="darkgreen")
title("Cubic Polynomial Spline")
```



SMOOTHING SPLINES

Does not require knot selection, smoothing parameter (lambda), can be specified via the effective degrees of freedom ‘df’. Here, smoothing spline has df=16, which is reather large; and is a bit wiggly, maybe a little too much in this case

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

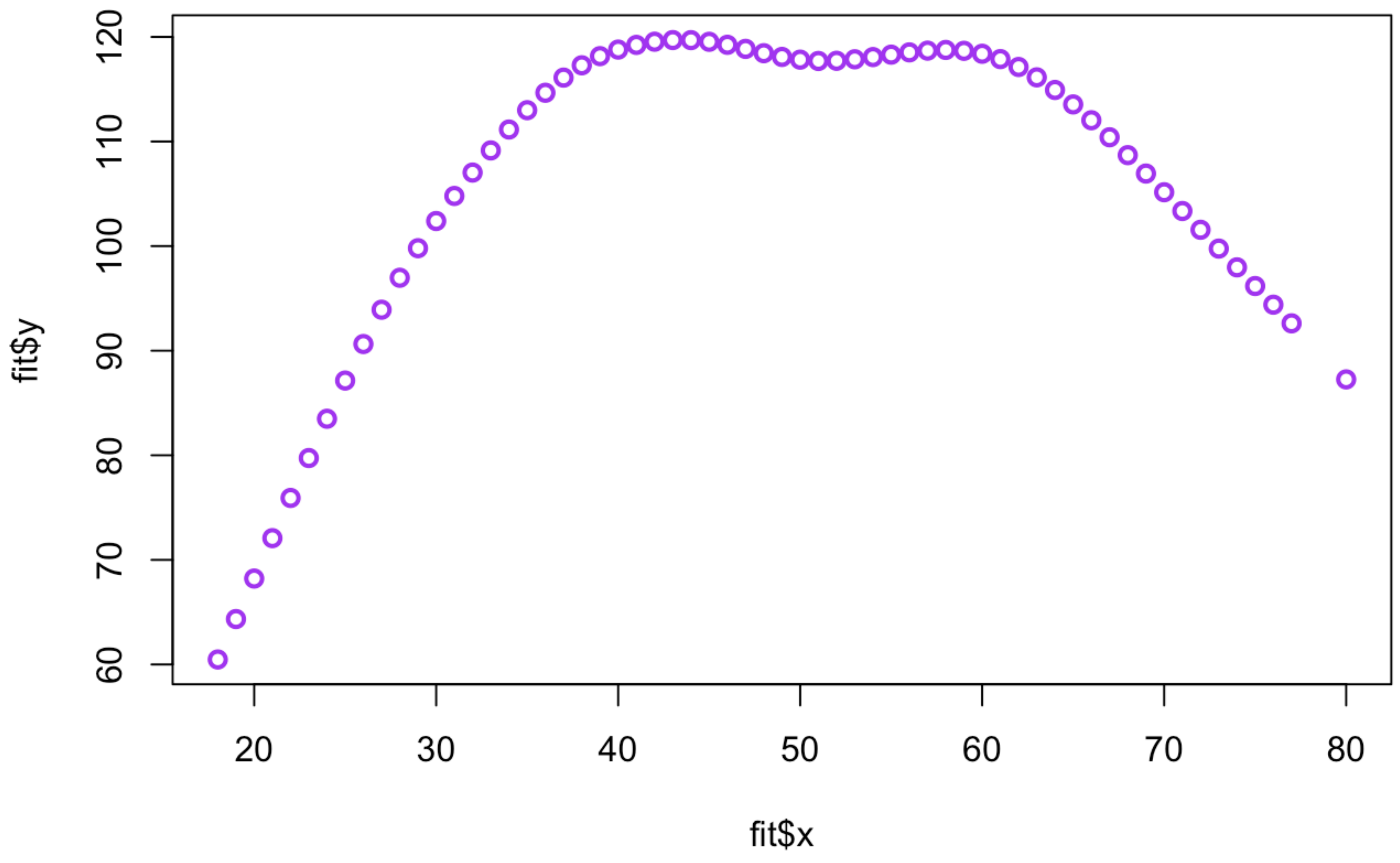


Alternatively, we can use CROSS-VALIDATION (LOOC) to select smoothing parameter automatically:

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

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

```
plot(fit, col="purple", lwd=2)
```



```
fit
```

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

So far, we focused on fitting models with mostly single nonlinear items.
===

GENERALIZED ADDITIVE MODELS (GAM)

‘gam’ package makes it easier to work with multiple nonlinear terms, knows how to plot these functions and their std errors. Enter wage as response, smoothing splines for age, year, df=4 Education is qualitative predictor, dummary variables

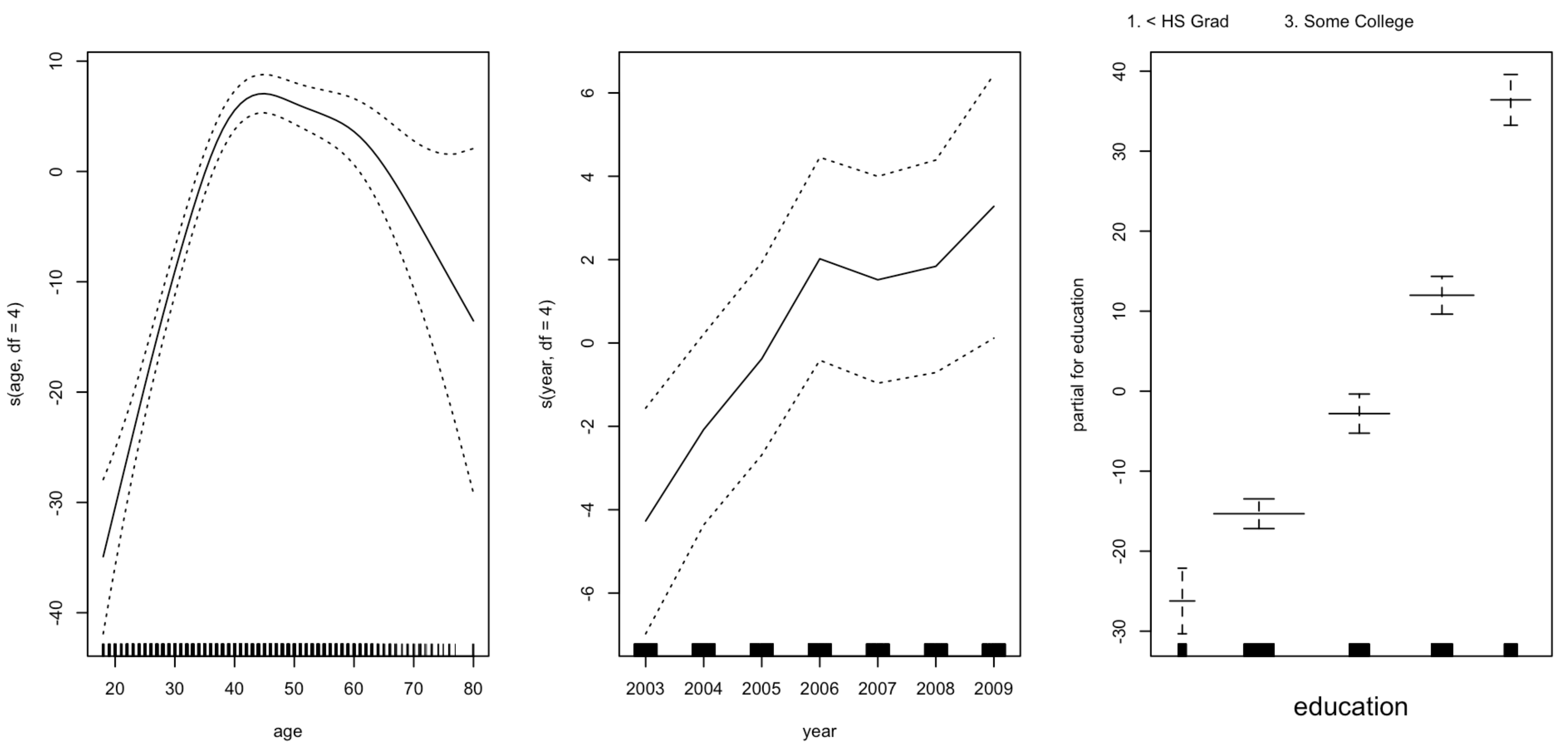
```
library(gam)
```

```
## Warning: package 'gam' was built under R version 3.3.2
```

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

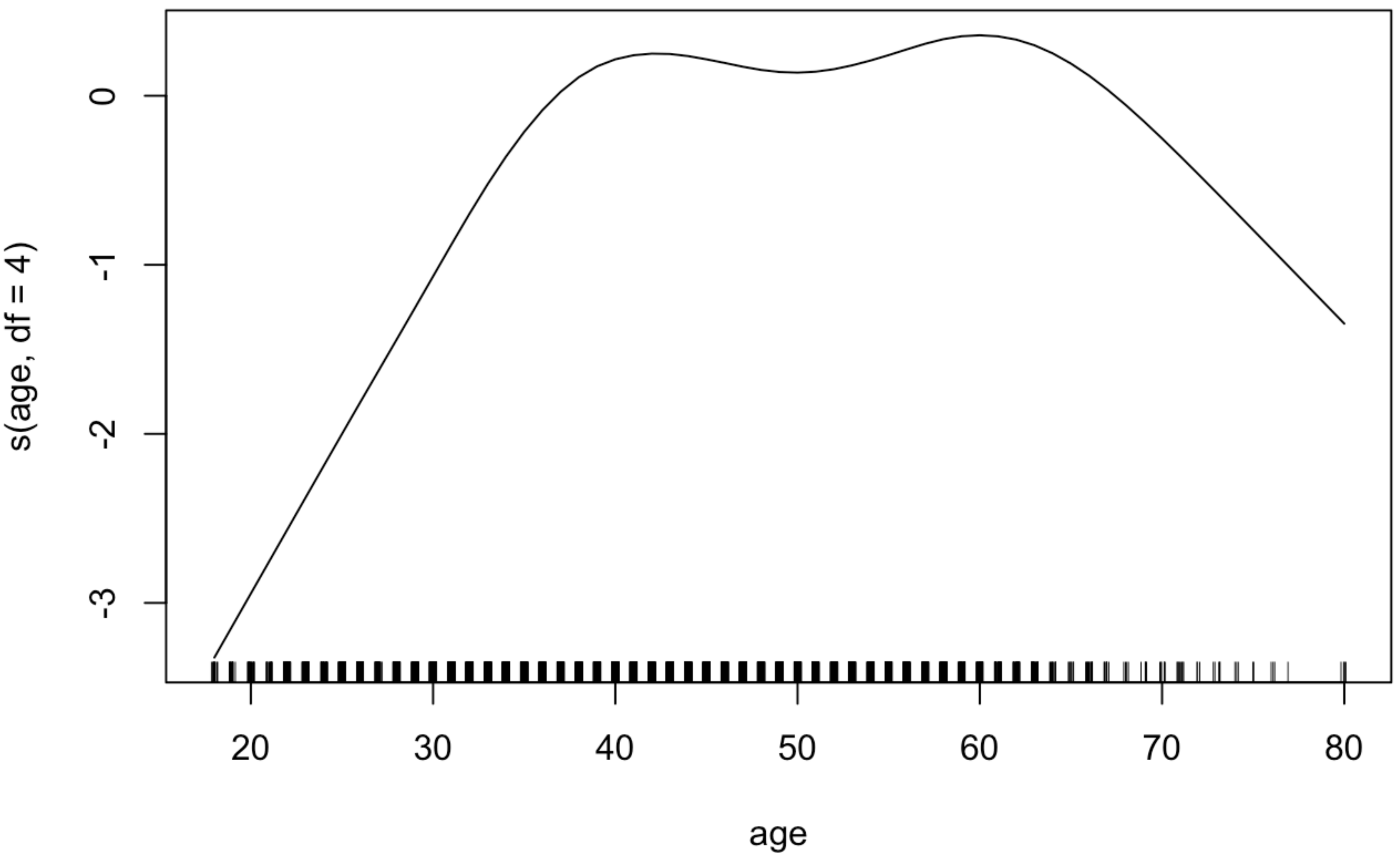
```
## Loaded gam 1.14-4
```

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

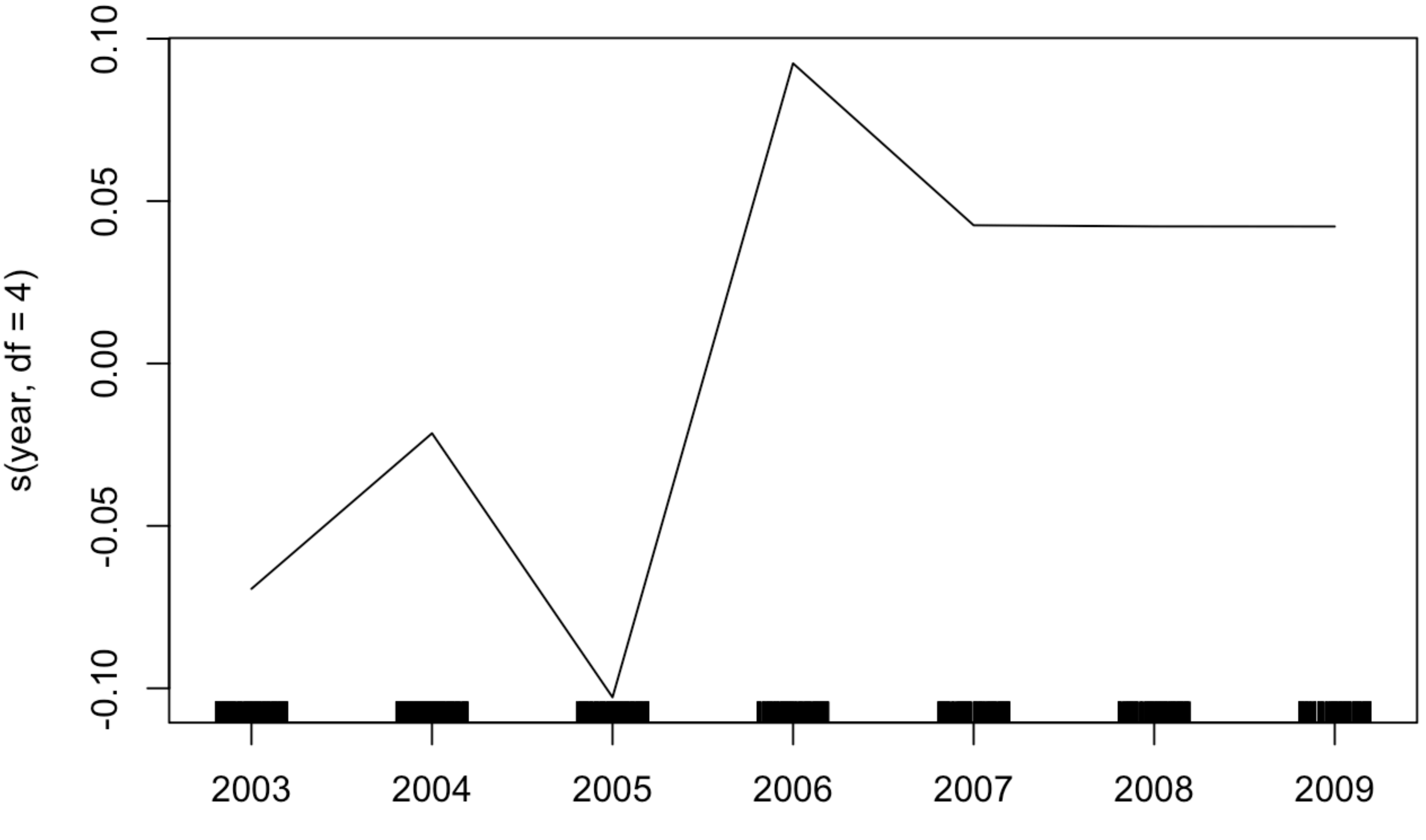


Same procedure with Logistic Regression for qualitative response

```
gam.2 = gam(I(wage>250)~s(age,df=4)+s(year,df=4)+education, data=Wage, family=binomia
l)
plot(gam.2)
```

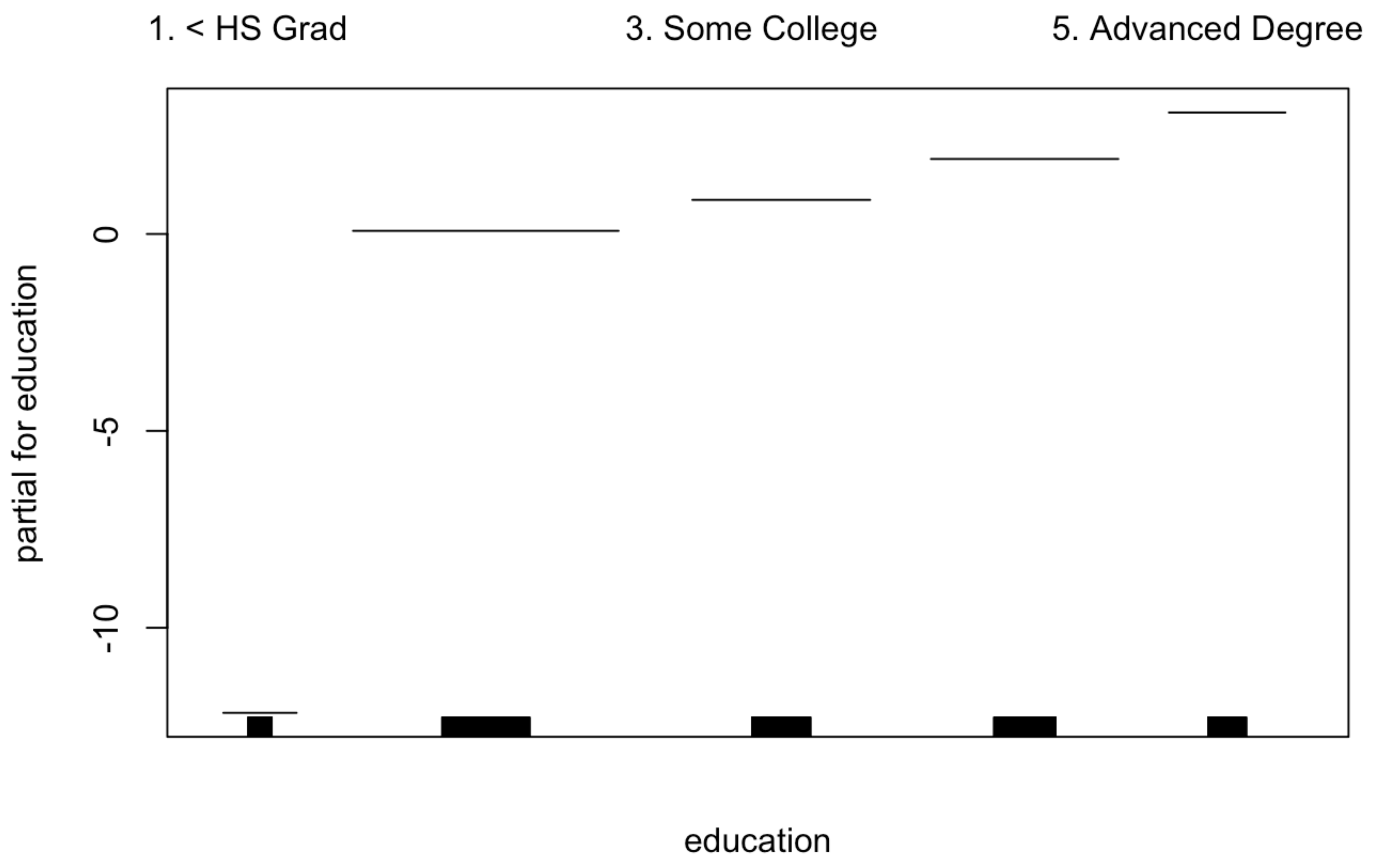


s(age, df = 4)
0
-1
-2
-3
20
30
40
50
60
70
80
age



s(year, df = 4)
0.10
0.05
0.00
-0.05
-0.10
2003
2004
2005
2006
2007
2008
2009

year



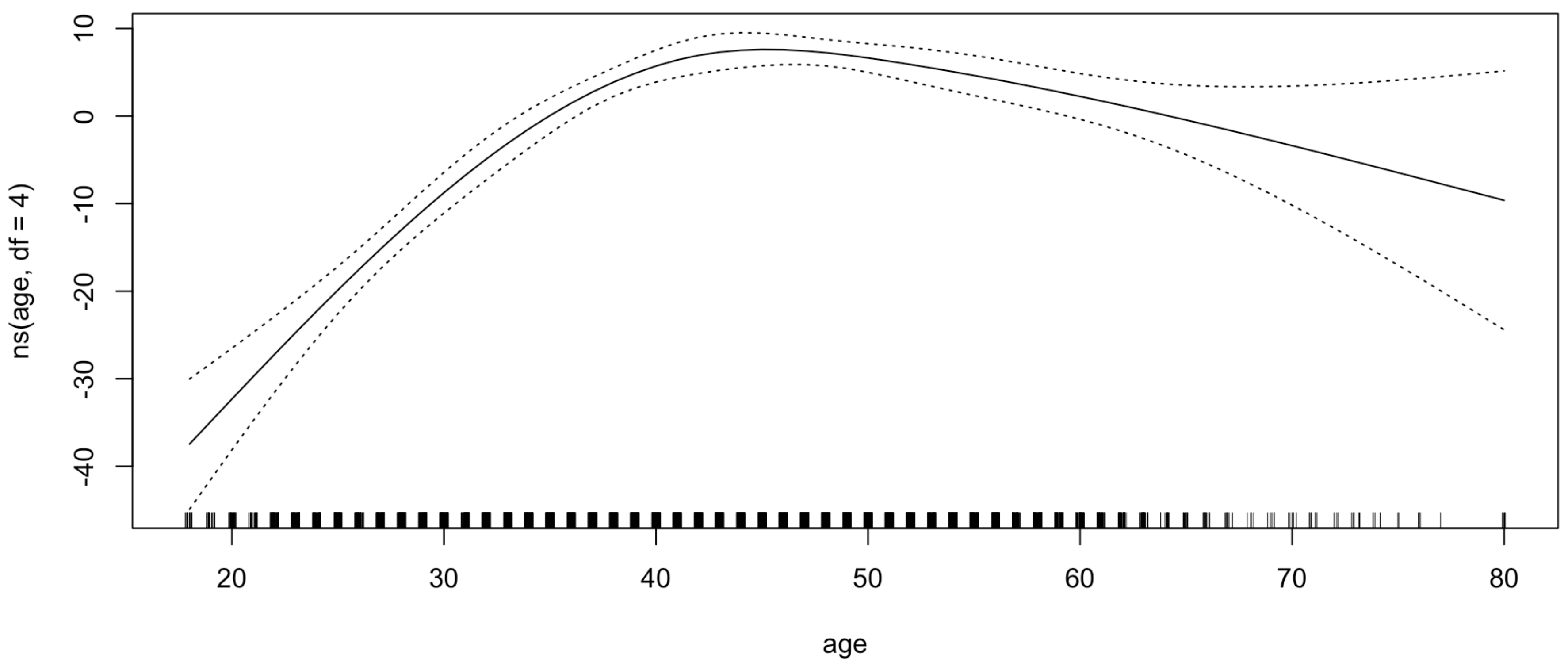
See if we need a nonlinear term for year by comparing two models using anova () (non-paramatric)

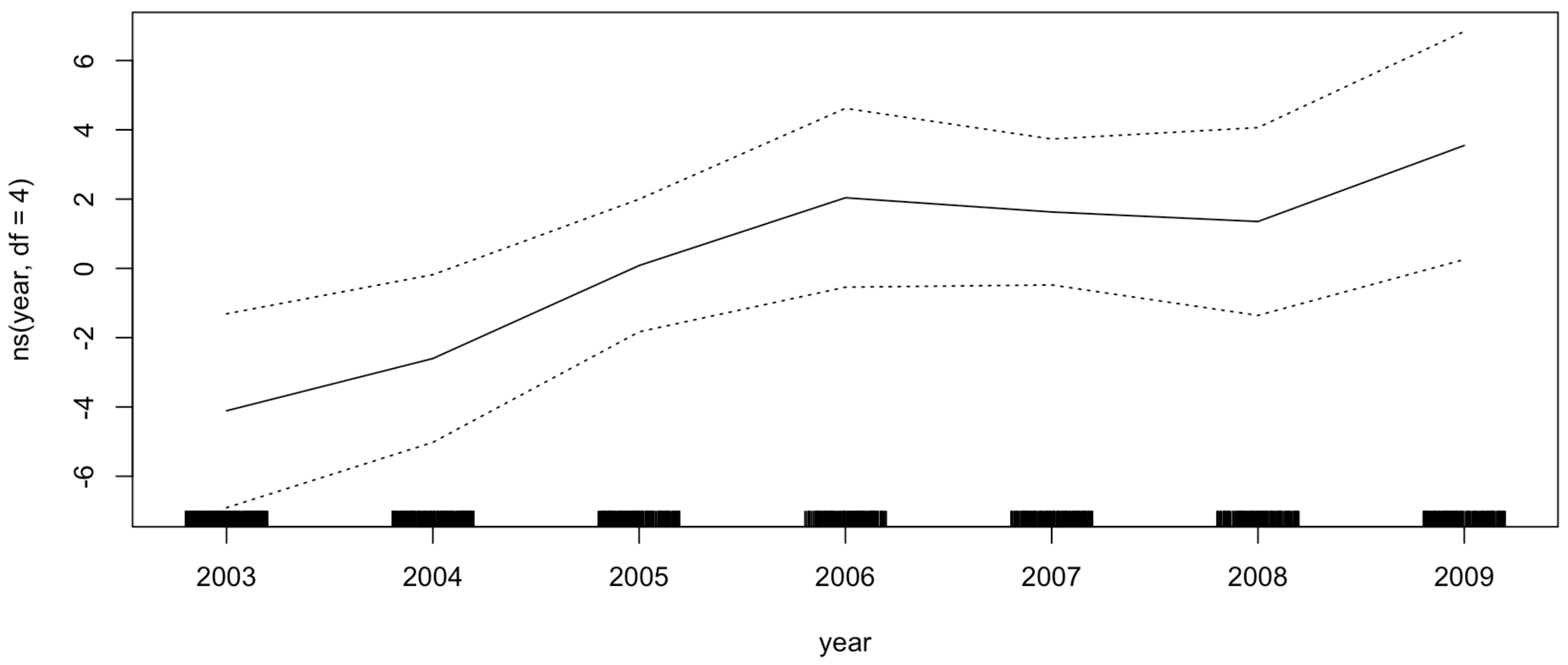
```
gam.3 = gam(I(wage>250)~s(age,df=4)+year+education, data=Wage, family=binomial)
anova(gam.2,gam.3,test="Chisq")
```

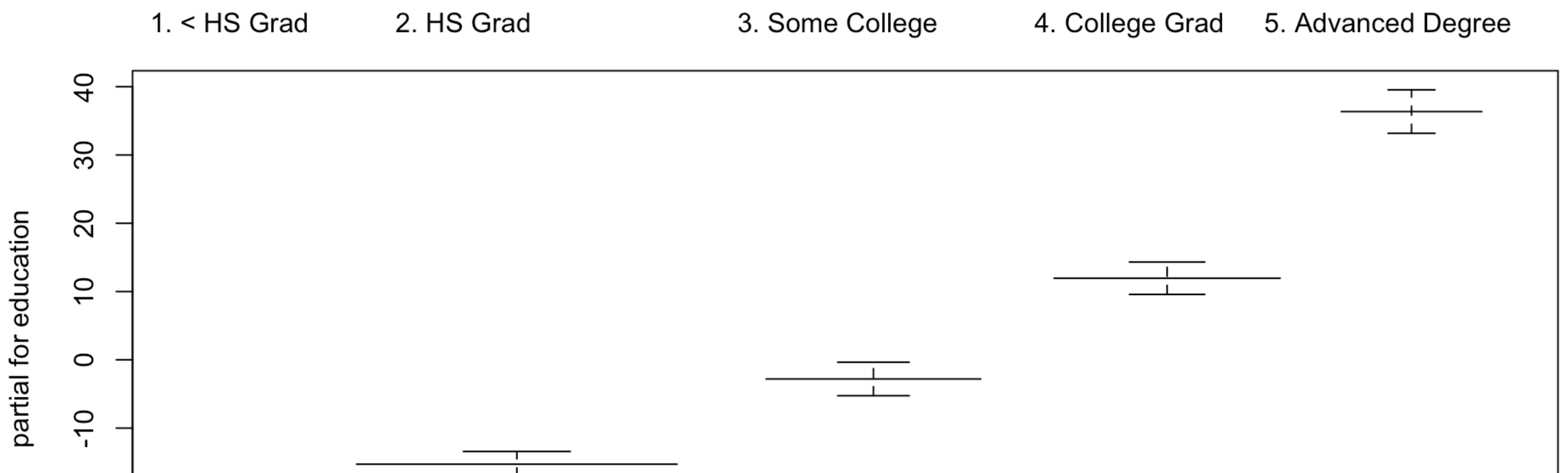
```
## Analysis of Deviance Table
##
## Model 1: I(wage > 250) ~ s(age, df = 4) + s(year, df = 4) + education
## Model 2: I(wage > 250) ~ s(age, df = 4) + year + education
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      2987     602.87
## 2      2990     603.78 -3 -0.90498   0.8242
```

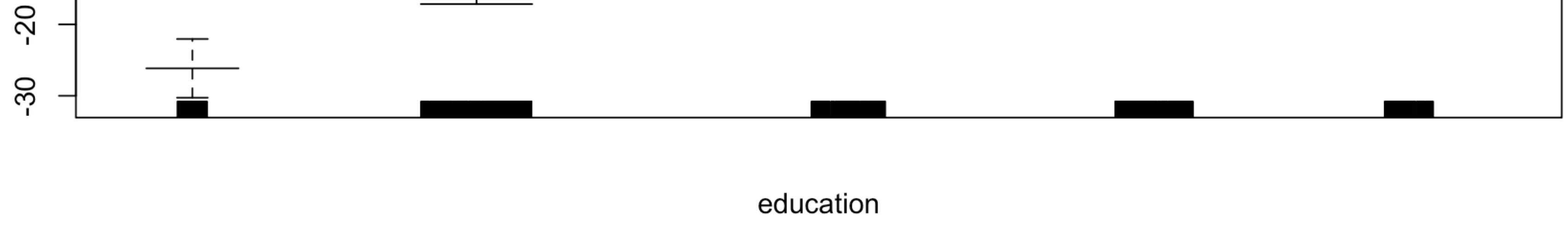
Nice feature of ‘gam’ package: it knows how to plot functions even for models fit by ‘lm’ and ‘glm’ Here, using linear model with natural splines on age, year

```
lm.1 = lm(wage~ns(age,df=4)+ns(year,df=4)+education,data=Wage)
plot.gam(lm.1, se=TRUE)
```


ns(age, df = 4)
10
0
-10
-20
-30
-40
20
30
40
50
60
70
80
age


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


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

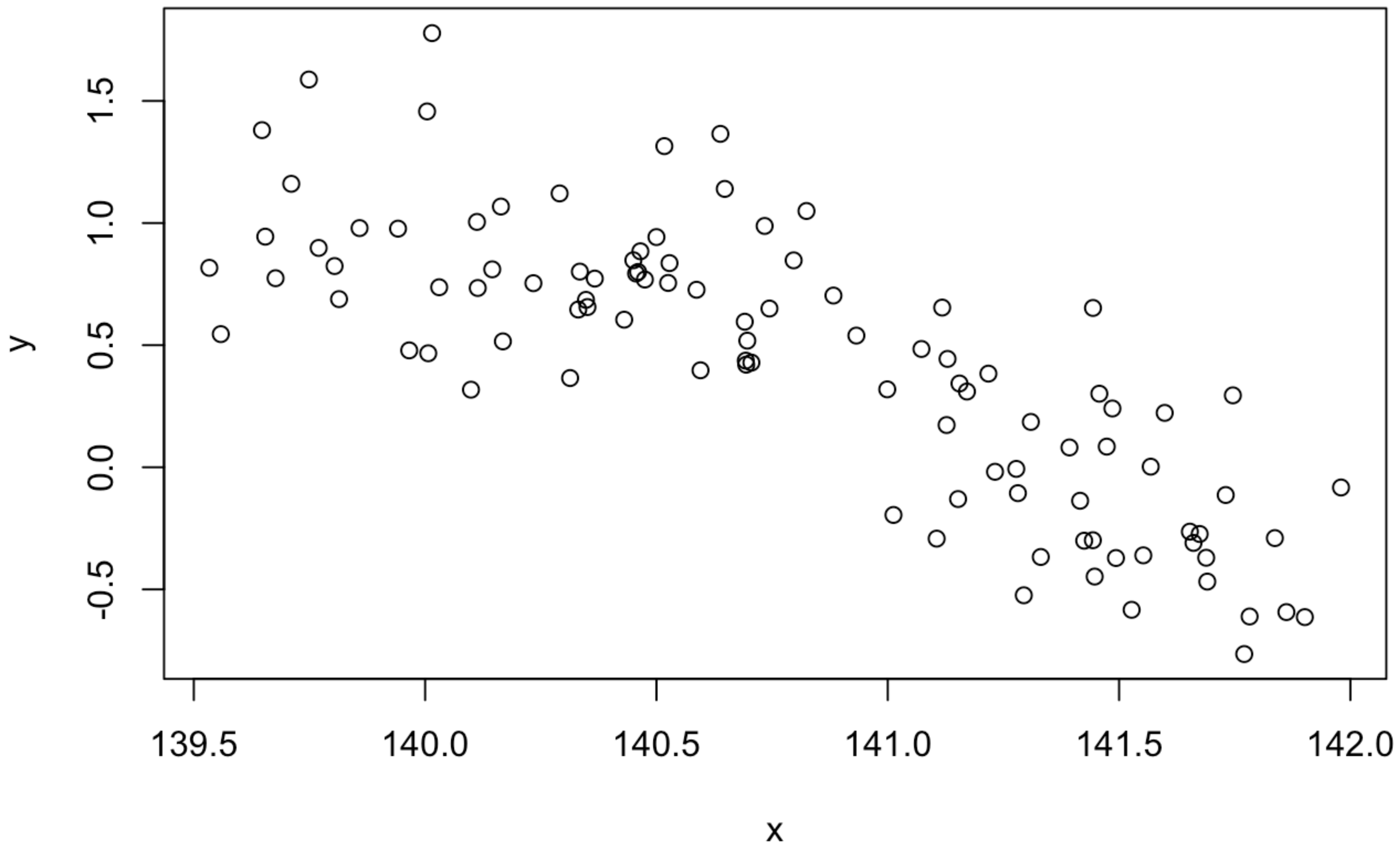


7.7 RData Quiz Question

```
par(mfrow=c(1,1))
load(file='7.R.RData')
attach('7.R.RData')
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     x, y
```

```
plot(x,y)
```



```
lm.fit = lm(y~x)
lm.fit
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##     95.4363      -0.6748
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71289 -0.26943 -0.02448  0.21068  0.83582
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 95.43627    7.14200   13.36   <2e-16 ***
## x           -0.67483    0.05073  -13.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3376 on 98 degrees of freedom
## Multiple R-squared:  0.6436, Adjusted R-squared:   0.64
## F-statistic:   177 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
lm.fit2 = lm(y~I(1+x)+I(x^2))
lm.fit2
```

```
##
## Call:
## lm(formula = y ~ I(1 + x) + I(x^2))
##
## Coefficients:
## (Intercept)     I(1 + x)       I(x^2)
##  -5498.9542      77.7075      -0.2784
```

```
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = y ~ I(1 + x) + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65698 -0.18190 -0.01938  0.16355  0.86149
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.499e+03  1.569e+03  -3.506 0.000691 ***
## I(1 + x)     7.771e+01  2.197e+01   3.536 0.000624 ***
## I(x^2)      -2.784e-01  7.805e-02  -3.567 0.000563 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3191 on 97 degrees of freedom
## Multiple R-squared:  0.6849, Adjusted R-squared:  0.6784
## F-statistic: 105.4 on 2 and 97 DF,  p-value: < 2.2e-16
```