

Nonlinear models

Explore nonlinear models using R tools.

```
require(ISLR)
```

```
## Loading required package: ISLR
```

```
attach(Wage)
```

Polynomials

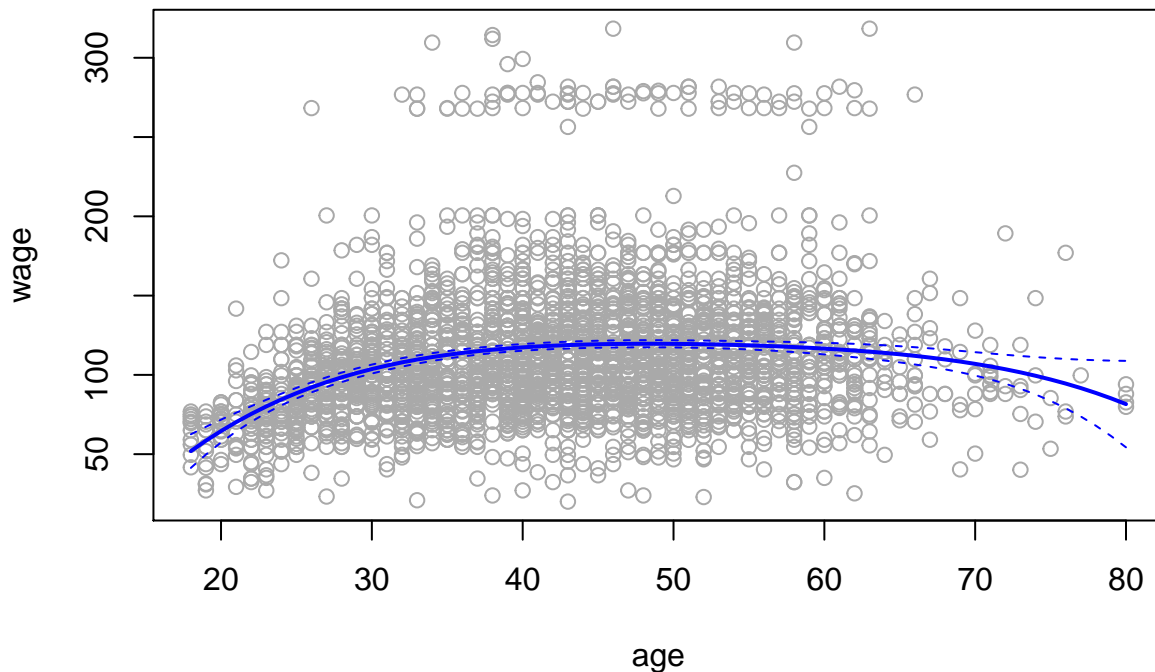
Focus on a single predictor, age:

```
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
```

The `poly()` function generates a basis of *orthogonal polynomials*. Let's make a plot of the fitted function, along with the standard errors of the fit.

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



There are other ways of fitting a polynomials in R. For example:

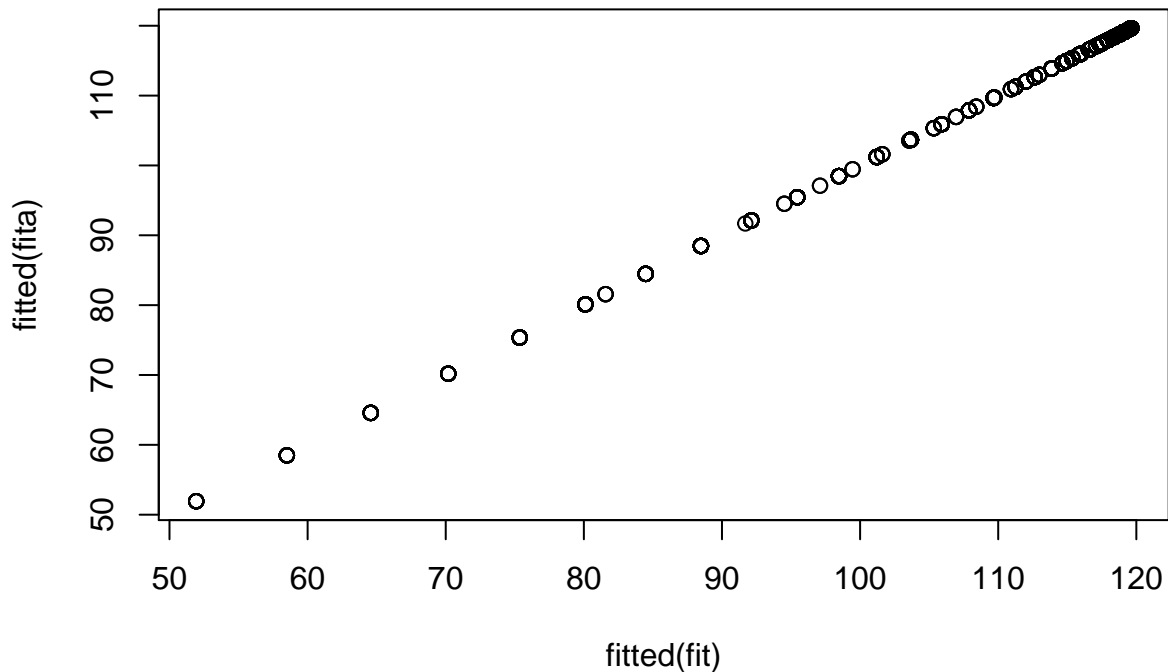
```
fita = lm(wage~age+I(age^2)+I(age^3)+I(age^4), data=Wage)
summary(fita)
```

```
##
## 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
```

Above `I()` wrapper is used because `age^2` means something to the formula language, while `I(age^2)` is protected.

The coefficients are different, but the fits are the same.

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



By using orthogonal polynomials in this simple way, it turns out that we can separately test for each coefficient. So we look at the summary again, we can see that the linear, quadratic, and cubic terms are significant, but not the quartic.

This only works with linear regression, and if there is a single predictor. In general, we would use `anova()` as this next example demonstrates. This is an example of nested sequence of complexity:

```
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, constructed from `wage`. We code the big earners (>250k) as 1, else 0

```
fita = glm(I(wage > 250) ~ poly(age,2), data=Wage, family=binomial)
fitb = glm(I(wage > 250) ~ poly(age,3), data=Wage, family=binomial)
anova(fita, fitb)
```

```
## Analysis of Deviance Table
##
## Model 1: I(wage > 250) ~ poly(age, 2)
## Model 2: I(wage > 250) ~ poly(age, 3)
##   Resid. Df Resid. Dev Df Deviance
## 1      2997      709.02
## 2      2996      707.92  1    1.1022
```

```
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
```

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

The above calculations are on the logit scale. We need to transform it back using the inverse logit

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

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(0,0.1))  
points(jitter(age), I(wage>250)/10, pch="|", cex=0.5)
```

