

# Ch. 7 - Moving Beyond Linearity

Chris Hayduk

March 5, 2021

## 1 Polynomial Regression and Step Functions

Polynomial regression and step functions both exist to give a more flexible fit than standard linear regression will allow. Polynomial regression works by creating new variables out of powers of the initial predictor variable. For example, if the initial predictor is  $x$ , we may use  $x, x^2$ , and  $x^3$  as our three predictors if we wish to use a cubic fit. On the other than, step functions work by cutting the data up into bins and then fitting constant functions on each of these bins.

We start here by fitting a degree 4 polynomial with the `lm()` function.

```
data(Wage)
attach(Wage)
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

The `poly()` function used above allows us to avoid writing out a long equation by hand. However, by default it returns a matrix whose columns are a basis of orthogonal polynomials. That is, each column is a linear combination of the variables `age`, `age2`, `age3`, and `age4`. While the choice of basis will not affect the fitted values, we may want to use `age`, `age2`, `age3`, and `age4` directly. We can do this by setting `raw=TRUE` in the arguments for the function `poly()`.

```
fit2=lm(wage~poly(age,4,raw=T), data=Wage)
coef(summary(fit2))
```

##		Estimate	Std. Error	t value	Pr(> t )
##	(Intercept)	-1.841542e+02	6.004038e+01	-3.067172	0.0021802539

```
## poly(age, 4, raw = T)1  2.124552e+01 5.886748e+00  3.609042 0.0003123618
## poly(age, 4, raw = T)2 -5.638593e-01 2.061083e-01 -2.735743 0.0062606446
## poly(age, 4, raw = T)3  6.810688e-03 3.065931e-03  2.221409 0.0263977518
## poly(age, 4, raw = T)4 -3.203830e-05 1.641359e-05 -1.951938 0.0510386498
```

There are several other equivalent ways of fitting this model in R. For example,

```
fit2a = lm(wage~age+I(age^2)+I(age^3)+I(age^4), data=Wage)
coef(fit2a)

##      (Intercept)          age      I(age^2)      I(age^3)      I(age^4)
## -1.841542e+02  2.124552e+01 -5.638593e-01  6.810688e-03 -3.203830e-05
```

and,

```
fit2b = lm(wage~cbind(age, age^2, age^3, age^4), data = Wage)
```

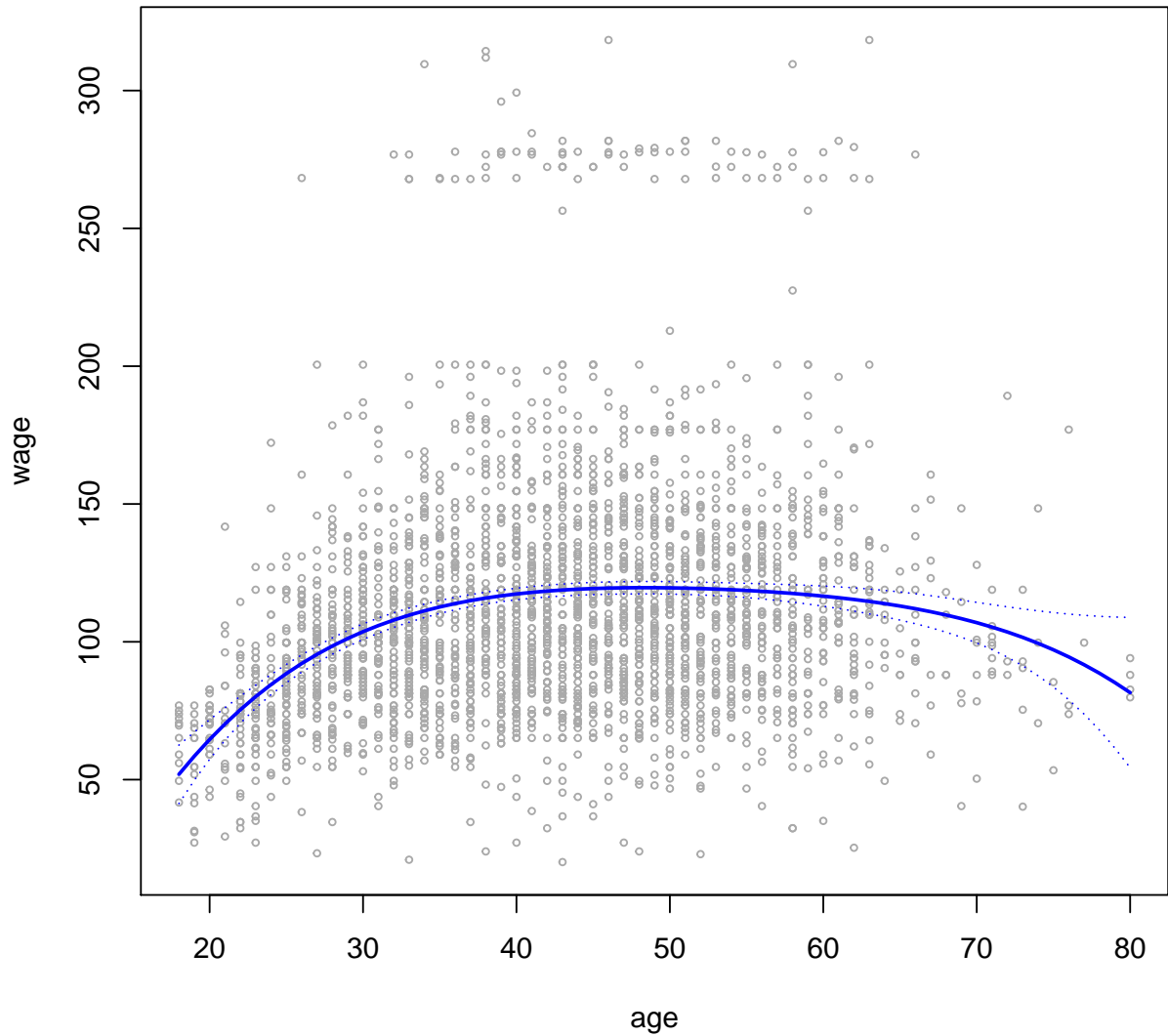
We now create a grid of values for `age` at which we want predictions, and then call the generic `predict()` function, specifying that we want standard errors as well.

```
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)
```

The above code creates our grid of age values, gets predictions for each value in the grid, and then sets the standard error bars at 95% confidence (two times the standard error plus/minus the prediction). Finally, let us plot the data and add the fit from the 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)
```

## Degree-4 Polynomial



As we mentioned above, the choice of basis will not affect the fitted values of the function. We can verify this by checking the predicted values on our age grid from both types of our fitted polynomial functions:

```
preds2 = predict(fit2, newdata=list(age=age.grid), se=TRUE)
max(abs(preds$fit-preds2$fit))

## [1] 7.81597e-11
```

As we can see, the maximum of the absolute value of the difference of these two fits is extremely close to 0.

In practice, we will want to check several possible polynomial degrees in order to find the simplest model that provides the best fit. We will now fit models ranging from linear to degree-5 polynomials and use the `anova()` function to perform an analysis of variance in order to determine the best model. Analysis of variance (ANOVA) works by testing the null hypothesis that a model  $M_1$  is sufficient to explain the data against the alternative hypothesis that a more complex model  $M_2$  is required. In order to use the `anova()` function,  $M_1$  and  $M_2$  must be nested models. In this case, we are fitting five different models and will 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 0, 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, so the quadratic fit is also insufficient. The p-value comparing the cubic model and the degree-4 polynomial is approximately 5%, while the degree-5 polynomial seems unnecessary because its p-value is 0.37. Hence, either a cubic or quartic polynomial appears to provide a reasonable fit to the data, but lower- or higher-order models are not justified.

We can actually avoid using the `anova()` function since `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
```

As you can see, the p-values are exactly 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:

```
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 ANOVA, we could also choose the polynomial degree using cross-validation.

Now let us consider the task of predicting whether an individual earns more than \$250,000 per year. We proceed much as before, except that first we create the appropriate response vector, and then apply the `glm()` function using `family="binomial"` in order to fit a polynomial logistic regression model.

```
fit = glm(I(wage>250) ~ poly(age,4), data = Wage, family = binomial)
```

We are using the wrapper `I()` to create the binary response variable on the fly. The expression `wage>250` evaluates to a logical variable containing `TRUE`s and `FALSE`s, which

`glm()` coerces to binary by setting **TRUEs** to 1 and **FALSEs** to 0.

We now use the `predict()` function to make our predictions,

```
preds=predict(fit, newdata=list(age=age.grid),se=T)
```

Confidence intervals in this scenario are a bit more involved than they were in the linear regression case. Our predictions are for the *logit*. That is, we have fit a model of the form

$$\log \left( \frac{\Pr(Y = 1|X)}{1 - \Pr(Y = 1|X)} \right) = X\beta$$

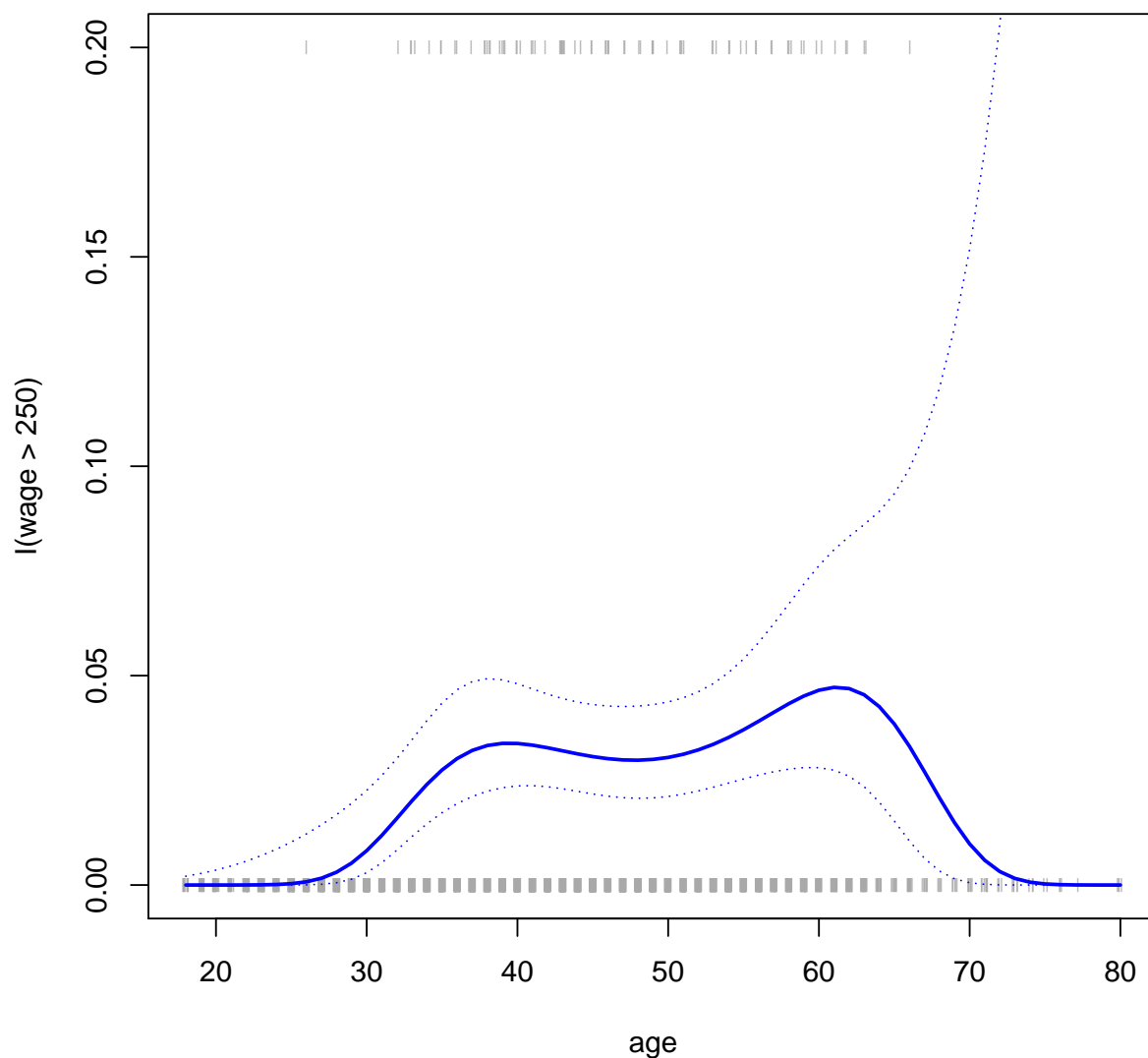
, and the predictions given are of the form  $X\hat{\beta}$ . The standard errors given are also of this form. In order to obtain confidence intervals for  $\Pr(Y = 1|X)$ , we use the transformation

$$\Pr(Y = 1|X) = \frac{\exp(X\beta)}{1 + \exp(X\beta)}$$

```
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))
```

However, the corresponding confidence intervals would not have been sensible because we would end up with negative probabilities! Finally, the right-hand plot from Figure 7.1 was made as follows:

```
plot(age,I(wage>250), xlim=agelims, type="n", ylim=c(0,0.2))
points(jitter(age), I((wage>250)/5), cex=.5, pch="|", col="darkgrey")
lines(age.grid, pfit, lwd=2, col="blue")
matlines(age.grid, se.bands, lwd=1, col="blue", lty=3)
```



In order to fit a step function, as discussed in Section 7.2, we use the `cut()` function.

```
table(cut(age,4))

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

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
```

Here `cut()` automatically picked the cutpoints at 33.5, 49, and 64.5 years of age. The function `cut()` returns an ordered categorical variable; the `lm()` function then creates a set of dummy variables for use in the regression. The `age<33.5` category is left out, so the intercept coefficient of \$94,160 can be interpreted as the average salary for those under 33.5 years of age, and the other coefficients can be interpreted as the average additional salary for those in the other age groups.

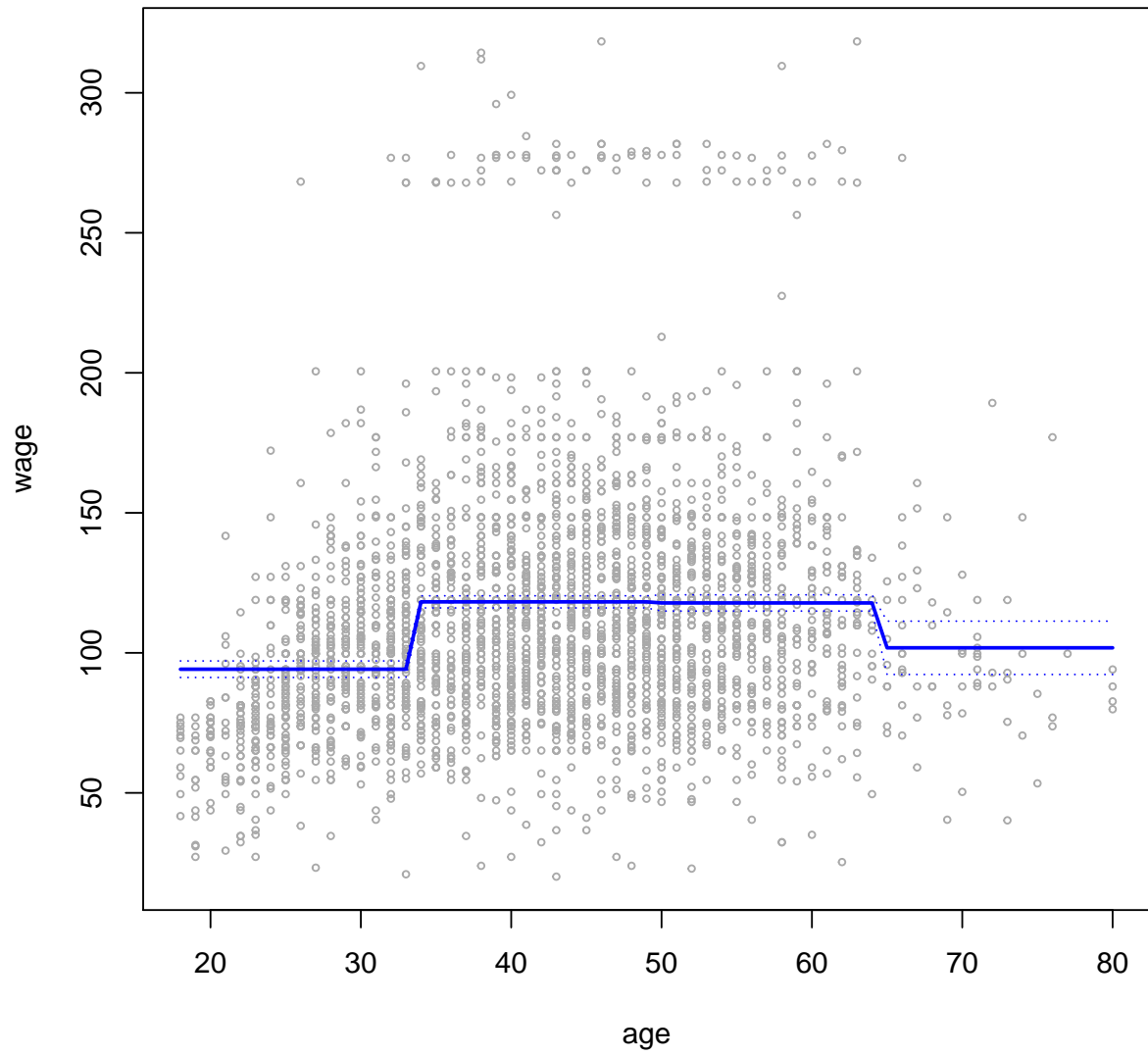
We can produce predictions and plots just as we did in the case of the polynomial fit:

```
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, xlim=agelims, cex=.5, col="darkgrey")
title("Step Function", outer=T)
lines(age.grid, preds$fit, lwd=2, col="blue")
matlines(age.grid, se.bands, lwd=1, col="blue", lty=3)
```



## Step Function



**2   Splines**

**3   GAMs**