Lab 7 - Non-Linear Modeling

An Introduction to Statistical Learning

We will be using the `Wage` dataset

```
pacman::p_load(ISLR)
attach(Wage)
```

1. Polynomial Regression and Step Functions

Polynomial Regression

Fitting the model

Our goal is to produce two plots, one showing `wage` vs `age` and other showing `wage>250` vs `age`.

We start by fitting a linear model with powers of `age`:

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

Making predictions

We create a grid for `age` in which we will make predictions:

```
agelims = range(age)
age.grid = seq(from=agelims[1], to=agelims[2])
```

Now we make the predictions and compute standard errors:

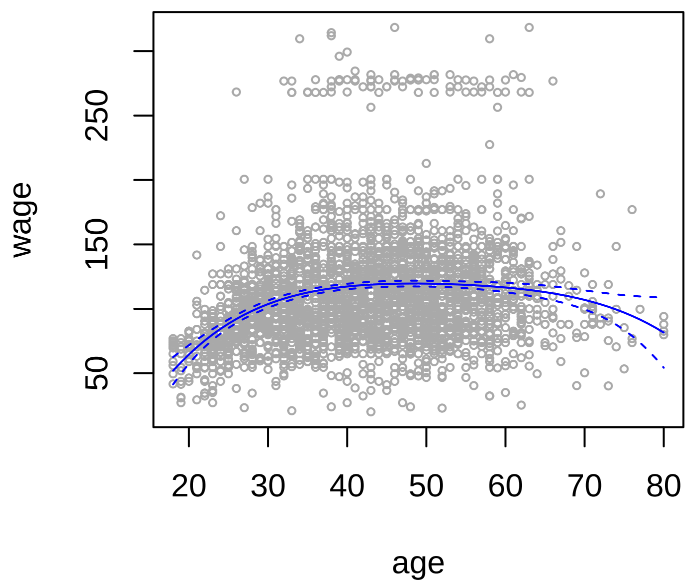
```
poly.pred = predict(poly.fit, newdata = list(age = age.grid), se.fit = TRUE)
```

We create a matrix containing the standard error intervals:

```
se.bands = cbind(poly.pred$fit - 2*poly.pred$se.fit,
                 poly.pred$fit + 2*poly.pred$se.fit)
```

We now produce the first plot, showing `wage` vs `age` with a confidence interval of 95%:

```
plot(age, wage, xlim = agelims, cex=.5, col='darkgrey')
lines(age.grid, poly.pred$fit, lwd = 1, col = 'blue')
matlines(age.grid, se.bands, lwd = 1, col = 'blue', lty = 2)
```



Creating the second plot requires a little more work. We start by selecting the degree for the polynomial on `age`. To do that we will use `anova()`:

```
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 output of the ANOVA analysis shows that the p-value comparing `fit.1` and `fit.2` is essentially 0, which means that considering both models are equivalent (H_0), the probability of obtaining the performance difference between them that ANOVA found is almost 0, so `fit.2` is better than `fit.1`. The same happens for `fit.3` and arguibly for `fit.4`.

We will use a 4-degree polynomial to fit the data to a *qualitative* model with `glm()`, in which the target is the probability of `wage>250`, so we need `family=binomial`:

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

Now we make predictions for the age grid:

```
poly.4.pred = predict(poly.4.fit, newdata = list(age = age.grid), se.fit = TRUE)
```

The default prediction type is `link`, which for a default `binomial` model are *log-odds*, probabilities on *logit* scale:

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

Using `type="response"`, which gives the actual predicted probabilities, seems the correct choice here, but the confidence intervals we obtained this way would have negative values.

We need to convert the *logit* probabilities to actual probabilities:

$$p(Y=1\mid X) = \frac{\exp{(X\beta)}}{1+\exp{(X\beta)}}$$

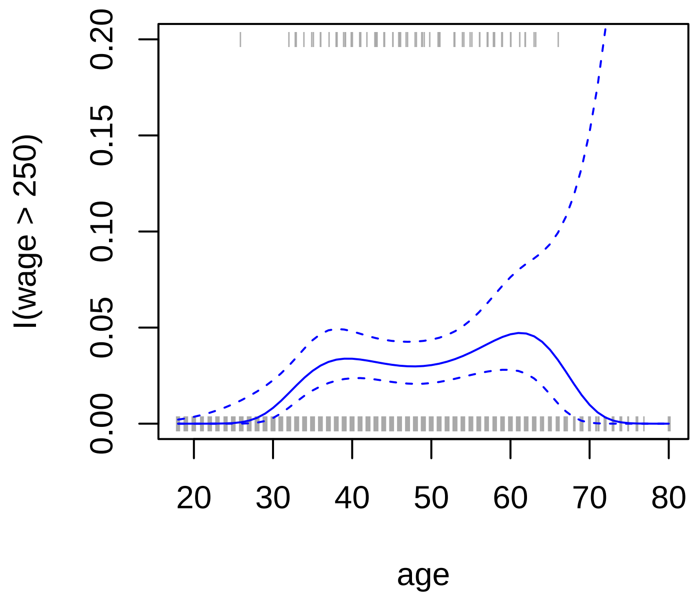
```
pfit = exp(poly.4.pred$fit) / (1 + exp(poly.4.pred$fit))
```

And for the confidence interval:

```
se.bands.logit = cbind(poly.4.pred$fit - 2 * poly.4.pred$se.fit,
                       poly.4.pred$fit + 2 * poly.4.pred$se.fit)
se.bands = exp(se.bands.logit) / (1 + exp(se.bands.logit))
```

We can now create the second plot:

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



Step Functions

The `cut()` function divides the range of the data into intervals and assigns each data point to one of those intervals, returning an ordered *categorical* variable:

```
cut(age, 4)[1:5]
```

```
## [1] (17.9,33.5] (17.9,33.5] (33.5,49]   (33.5,49]   (49,64.5]
## Levels: (17.9,33.5] (33.5,49] (49,64.5] (64.5,80.1]
```

Labels can be passed to `cut()` to name the different levels or intervals.

To get the total count of observations for each interval `table()` is used:

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

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

We can fit a linear model using these levels that creates *dummy* variables:

```
step.fit = lm(wage ~ cut(age, 4), data=Wage)
coef(summary(step.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
```