RLab 7

Jon Griffith

2025-04-24

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.4.3
```

```
library(fields)
```

```
## Loading required package: spam
```

```
## Spam version 2.10-0 (2023-10-23) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: viridisLite
```

```
##
## Try help(fields) to get started.
```

```
library(splines)
```

For this lab, we focus on Wage data contained in ISLR2 package.

```
head(Wage)
```

```
##        year age           maritl     race       education             region
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad 2. Middle Atlantic
## 86582  2004  24 1. Never Married 1. White 4. College Grad 2. Middle Atlantic
## 161300 2003  45       2. Married 1. White 3. Some College 2. Middle Atlantic
## 155159 2003  43       2. Married 3. Asian 4. College Grad 2. Middle Atlantic
## 11443  2005  50      4. Divorced 1. White      2. HS Grad 2. Middle Atlantic
## 376662 2008  54       2. Married 1. White 4. College Grad 2. Middle Atlantic
##              jobclass          health health_ins  logwage      wage
## 231655  1. Industrial      1. <=Good      2. No 4.318063  75.04315
## 86582  2. Information 2. >=Very Good      2. No 4.255273  70.47602
## 161300  1. Industrial      1. <=Good     1. Yes 4.875061 130.98218
## 155159 2. Information 2. >=Very Good     1. Yes 5.041393 154.68529
## 11443  2. Information      1. <=Good     1. Yes 4.318063  75.04315
## 376662 2. Information 2. >=Very Good     1. Yes 4.845098 127.11574
```

7.8.1 Polynomial Regression and Step Functions

We first fit a degree four polynomial using a few different methods. The first method uses the basis of the orthogonal polynomials while the other three use the raw data. They all produce similar results with just differing coefficient values.

Basis fit:

```
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
```

Raw fits with identical results:

```
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
```

```
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
```

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

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

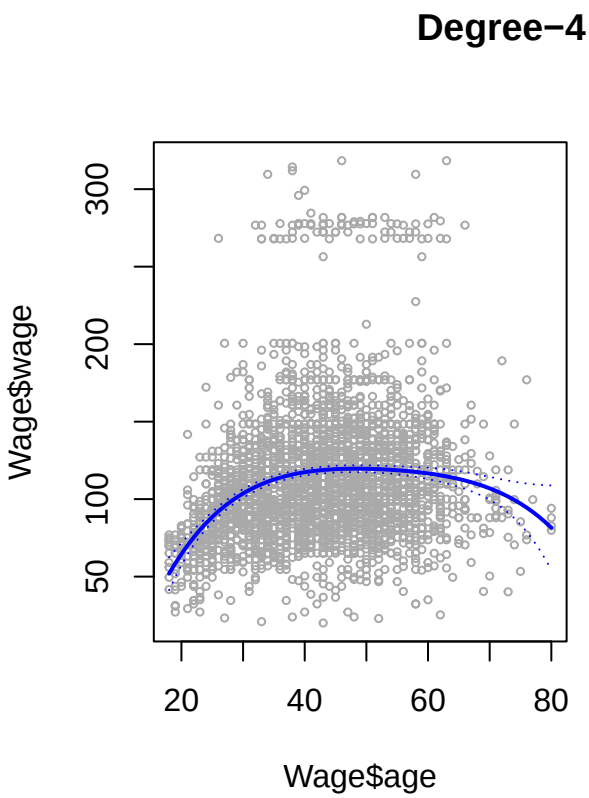
Now we move on to making predictions with our original model, along with corresponding standard errors.

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

# Calculate the 95% confidence interval for predictions
se.bands <- cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
```

Finally, we plot the data and add our degree four polynomial fit.

```
par(mfrow=c(1,2), mar = c(4.5, 4.5, 1, 1),
    oma = c(0, 0, 4, 0))
```

```
plot(Wage$age, Wage$wage, xlim=agelims, cex=0.5, 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)
```



Notice that the fitted values obtained from any of these models are identical.

```
preds2 <- predict(fit2, newdata=list(age=age.grid),
                  se=TRUE)
test.for.zero(preds$fit, preds2$fit)
```

```
## PASSED test at tolerance  1e-08
```

We'll now fit models ranging from degree 1 to degree 5 polynomials and look to select the simplest model. We'll test these models using the anova() function to perform analysis of variance between our models of interest. We see that the degree 3 and degree 4 both offer evidence of being good fits, but only those two.

```
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
```

Now we'll exploit the fact that the poly() function creates orthogonal polynomials and therefore can give the same results as our anova() function.The p-values are the same and the square of the t-statistics equal the F-statistics from anova().

```
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
```

ANOVA works whether or not the polynomials are orthogonal and should be used when there are other terms in the model as well. We'll demonstrate by comparing three models below. We can also select the degree polynomial we think is optimal using cross-validation.

```
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
```

We'll now fit a polynomial logistic regression model and predict whether an individual earns more than 250k per year. We create a mask for the 250k wage threshold and glm() will coerce this response vector to a binary vector. The predictions for a logistic regression model are given in terms of logit and so we need to transform these logits into probabilities using the sigmoid function. We need to use this transformation instead of the 'type="response"' option in the predict() function because the logits need to be transformed together for the fit and the standard error.

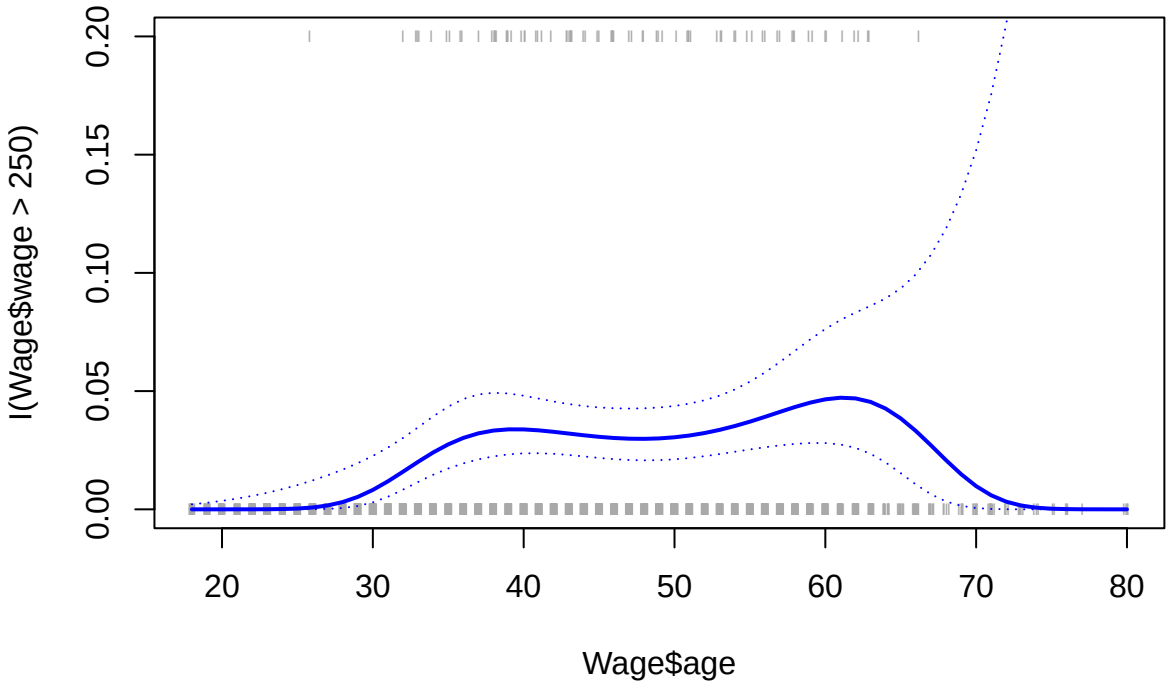
```
fit <- glm(I(wage > 250) ~ poly(age, 4), data=Wage, family=binomial)

preds <- predict(fit, newdata=list(age=age.grid), se=TRUE)
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))

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

```
##           [,1]        [,2]
## 1 4.560982e-14 0.002112586
## 2 2.067720e-12 0.002769461
## 3 6.283757e-11 0.003572811
## 4 1.315721e-09 0.004538493
## 5 1.949028e-08 0.005680373
```

Now we replicate the right-hand plot from Figure 7.1 in the text. Notice the ticks on the top correspond to ages with wage above 250k and the ticks on the bottom correspond to ages below 250k to give us a visual of the split. This is known as a 'rug plot'.

```
plot(Wage$age, I(Wage$wage > 250), xlim = agelims, type='n', ylim=c(0, 0.2))
points(jitter(Wage$age), I((Wage$wage>250) / 5), cex=0.5, pch='l', col='darkgrey')
lines(age.grid, pfit, lwd=2, col='blue')
matlines(age.grid, se.bands, lwd=1, col='blue', lty=3)
```



Next, we fit a step function using the cut() function to return an ordered categorical variable. We get four intervals ranging from the lowest to highest age.

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

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

```
cat('\n')
```

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

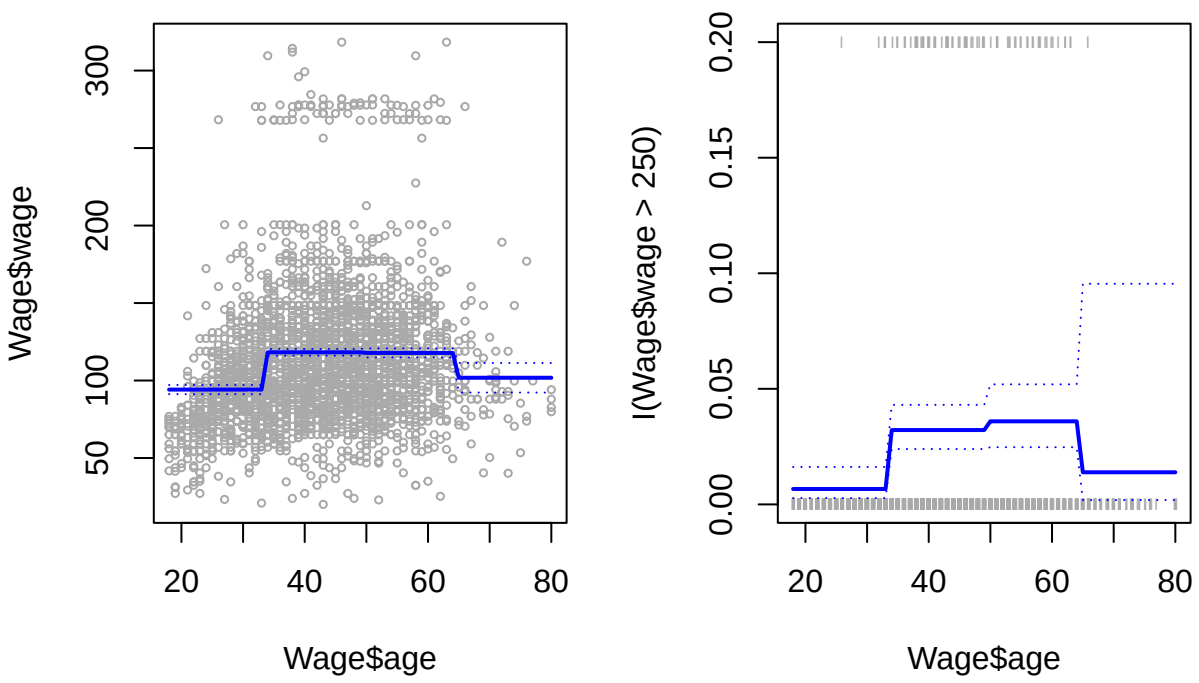
```
##                         Estimate Std. Error   t value     Pr(>|t|)
```

```
## (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
```

Now we can replicate the Figure 7.2 in the book using our step function fit.

```
preds <- predict(fit, newdata=list(age=age.grid), se=TRUE)

# Calculate the 95% confidence interval for predictions
se.bands <- cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
```

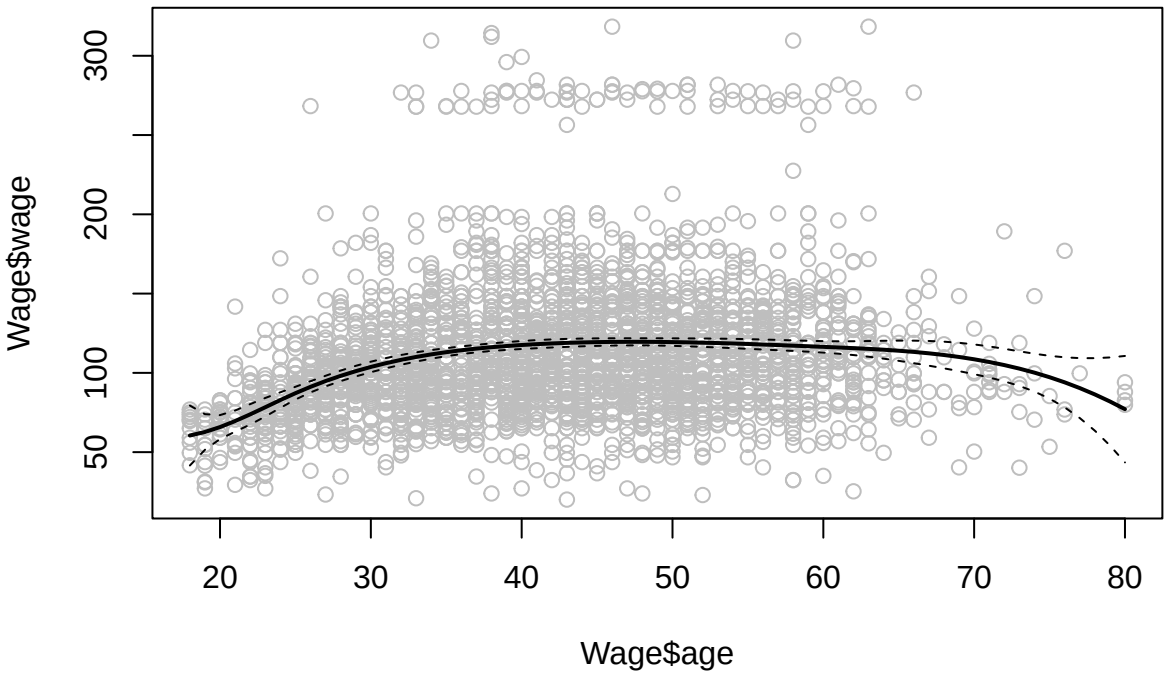
```
par(mfrow=c(1,2), mar = c(4.5, 4.5, 1, 1),
    oma = c(0, 0, 4, 0))

plot(Wage$age, Wage$wage, xlim=agelims, cex=0.5, col = 'darkgrey')
title('Piecewise Constant', outer=T)
lines(age.grid, preds$fit, lwd=2, col='blue')
matlines(age.grid, se.bands, lwd=1, col='blue', lty=3)


fit <- glm(I(wage > 250) ~ cut(age, 4), data=Wage, family=binomial)

preds <- predict(fit, newdata=list(age=age.grid), se=TRUE)
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))

plot(Wage$age, I(Wage$wage > 250), xlim = agelims, type='n', ylim=c(0, 0.2))
points(jitter(Wage$age), I((Wage$wage>250) / 5), cex=0.5, pch='l', col='darkgrey')
lines(age.grid, pfit, lwd=2, col='blue')
matlines(age.grid, se.bands, lwd=1, col='blue', lty=3)
```



7.8.2 Splines

Now we move onto splines and use the 'splines' library to fit our models. We first fit wage to age using a regression spline and use the bs() function to generate a matrix of basis functions with specified knots. We'll use three knots to produce a spline with six basis functions.

```
fit <- lm(wage ~ bs(age, knots = c(25, 40, 60)), data=Wage)
pred <- predict(fit, newdata = list(age = age.grid), se=TRUE)
plot(Wage$age, Wage$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')
```



We verify below that we have six basis functions and also show how we can use the df argument to produce three knots at uniform quantiles of the data.

```
dim(bs(Wage$age, knots = c(25, 40, 60)))
```

```
## [1] 3000    6
```

```
dim(bs(Wage$age, df=6))
```

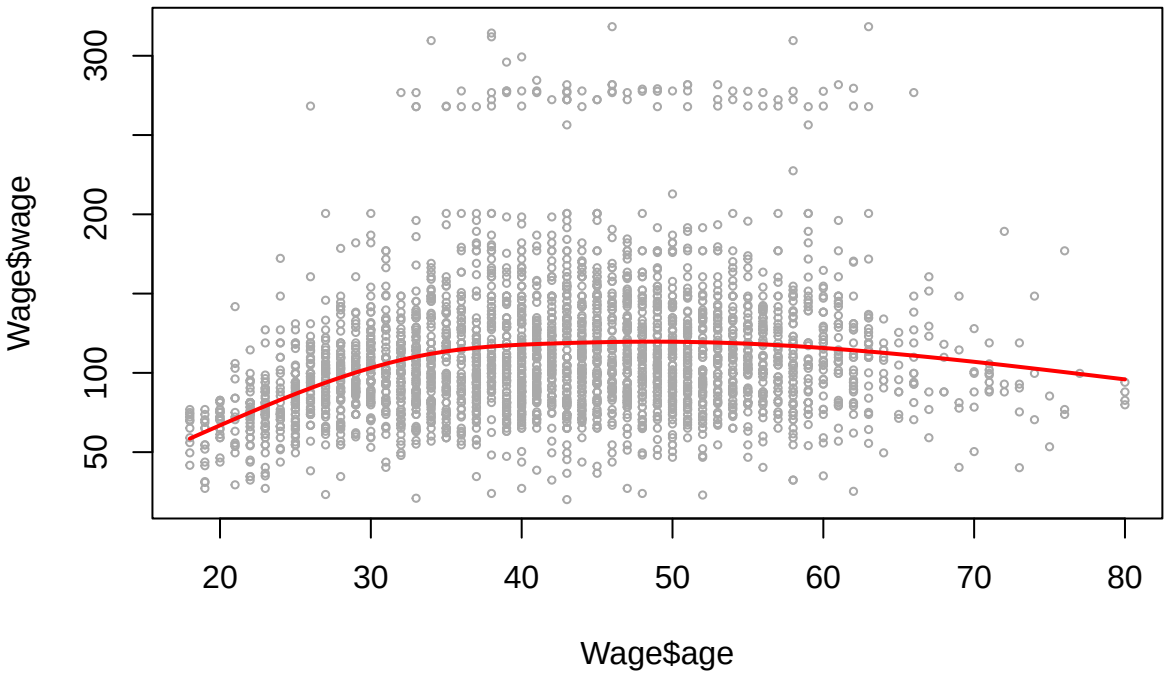
```
## [1] 3000    6
```

```
attr(bs(Wage$age, df=6), 'knots')
```

```
## [1] 33.75 42.00 51.00
```

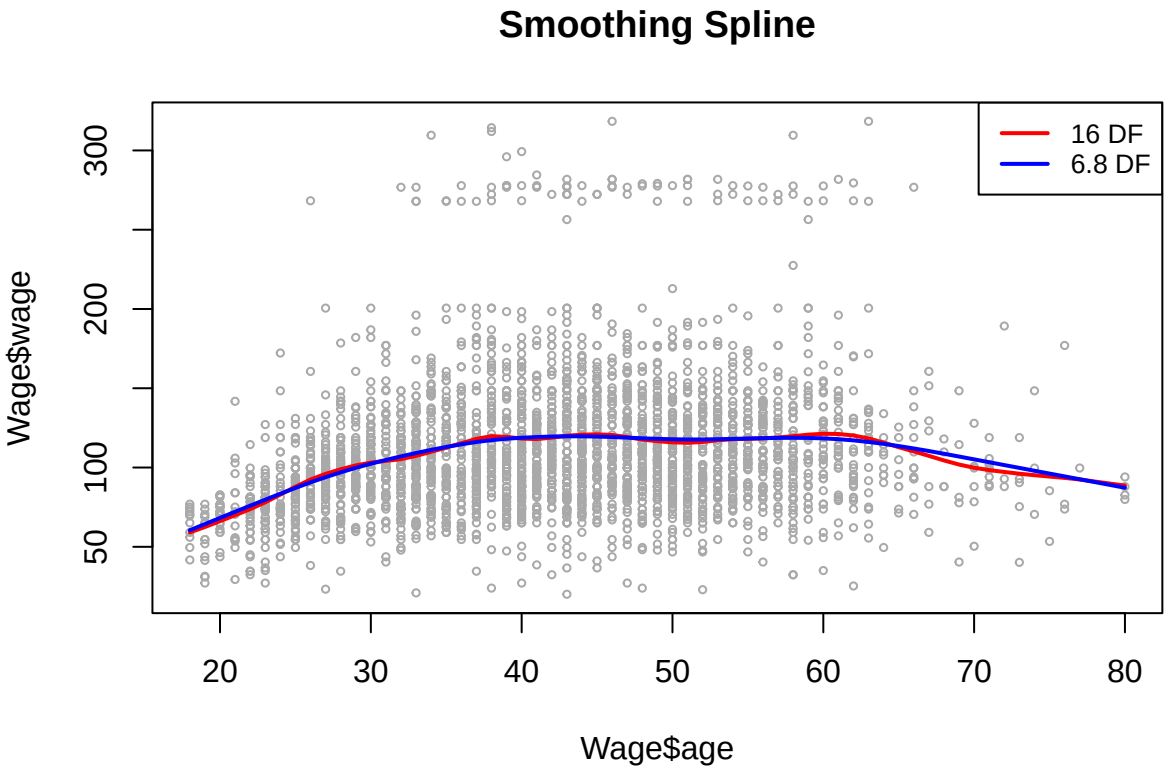
Now we'll fit a natural spline that has four degrees of freedom using the ns() function.

```
fit2 <- lm(wage ~ ns(age, df=4), data=Wage)
pred2 <- predict(fit2, newdata=list(age=age.grid), se=TRUE)

plot(Wage$age, Wage$wage, xlim=agelims, cex=0.5, col='darkgrey')
lines(age.grid, pred2$fit, col = 'red', lwd=2)
```



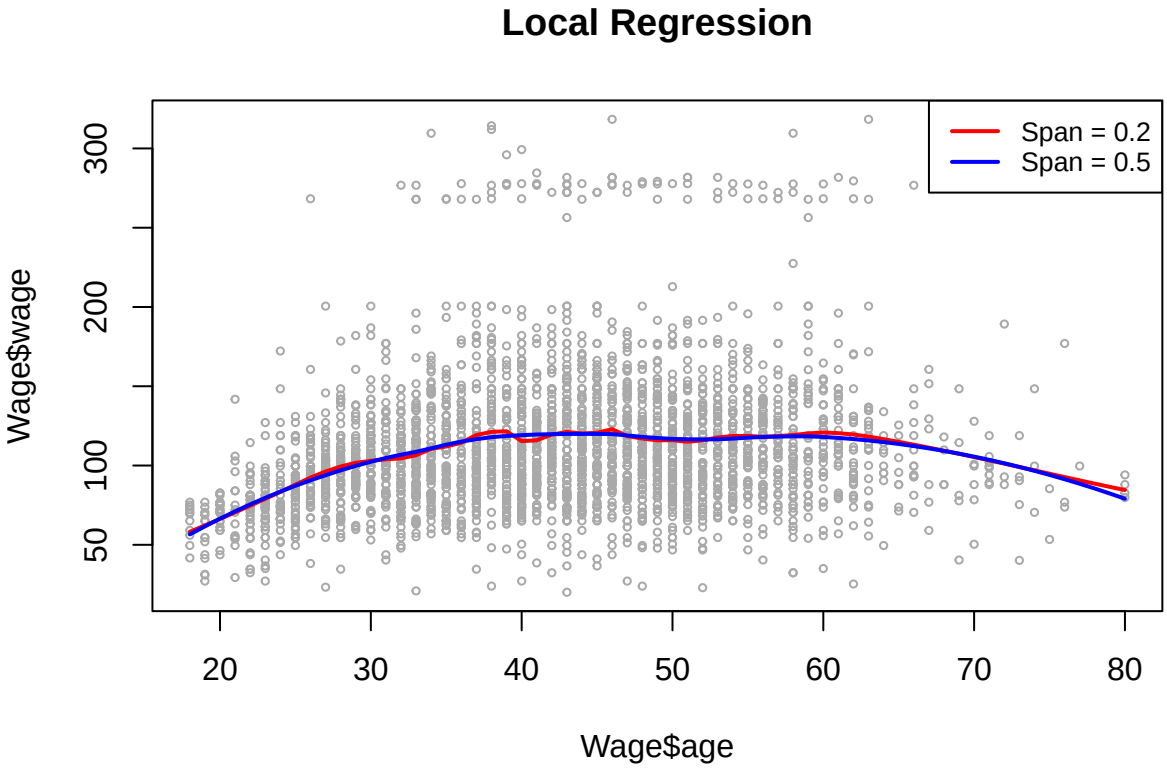
Now we'll fit a smoothing spline using the smooth.spline() function and produce the Figure 7.8 from the text. The first spline we specify 16 DF and a lambda is determined that meets that number while the second uses cross-validation to find a lambda that produces 6.8 DF.

```
fit <- smooth.spline(Wage$age, Wage$wage, df=16)
fit2 <- smooth.spline(Wage$age, Wage$wage, cv=TRUE)

plot(Wage$age, Wage$wage, xlim=agelims, cex=0.5, col='darkgrey')
title('Smoothing Spline')
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=0.8)
```



Finally, we'll fit a local regression using the loess() function. We'll specify spans of 0.2 and 0.5 for two different fits that will make each neighborhood consist of 20 percent and 50 percent of the observations, respectively. As we'll see, the larger we make this span the smoother the fit will be.

```
fit <- loess(wage ~ age, span=0.2, data=Wage)
fit2 <- loess(wage ~ age, span=0.5, data=Wage)

plot(Wage$age, Wage$wage, xlim=agelims, cex=0.5, col='darkgrey')
title('Local Regression')
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=0.8)
```


Local Regression
Span = 0.2
Span = 0.5
Wage$wage
Wage$age
50
100
200
300
20
30
40
50
60
70
80