Chapter 7 - Non-linear Models

Lab Solution

1 Problem 9

```r
set.seed(1)
library(MASS)
attach(Boston)

(a). Use the `poly()` function to fit a cubic polynomial regression to predict `nox` using `dis`. Report the regression output, and plot the resulting data and polynomial fits.

```r
lm.fit = lm(nox ~ poly(dis, 3), data = Boston)
summary(lm.fit)
```

```r
### Call:
### lm(formula = nox ~ poly(dis, 3), data = Boston)
### ### Residuals:
###     Min 1Q Median 3Q Max
###    -0.121130 -0.040619 -0.009738 0.023385 0.194904
### ### Coefficients:
### Estimate Std. Error t value Pr(>|t|)
### (Intercept) 0.554695 0.002759 201.021 < 2e-16 ***
### poly(dis, 3)1 -2.003096 0.062071 -32.271 < 2e-16 ***
### poly(dis, 3)2 0.856330 0.062071 13.796 < 2e-16 ***
### poly(dis, 3)3 -0.318049 0.062071 -5.124 4.27e-07 ***
### ---
### Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
### ### Residual standard error: 0.06207 on 502 degrees of freedom
### Multiple R-squared: 0.7148, Adjusted R-squared: 0.7131
### F-statistic: 419.3 on 3 and 502 DF, p-value: < 2.2e-16
```

```r
dislim = range(dis)
dis.grid = seq(from = dislim[1], to = dislim[2], by = 0.1)
lm.pred = predict(lm.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, lm.pred, col = "red", lwd = 2)
```
(b). Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```r
all.rss = rep(NA, 10)
for (i in 1:10) {
  lm.fit = lm(nox ~ poly(dis, i), data = Boston)
  all.rss[i] = sum(lm.fit$residuals^2)
}
all.rss
## [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630 1.833331 1.832171
```

(c). Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```r
library(splines)
sp.fit = lm(nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)
summary(sp.fit)
```

```
## Call:
```
## lm(formula = nox ~ bs(dis, df = 4, knots = c(4, 7, 11)), data = Boston)
##
## Residuals:
##       Min        1Q   Median        3Q       Max
## -0.124567 -0.040355 -0.008702  0.024740  0.192920
##
## Coefficients:
##                     Estimate      Std. Error  t value  Pr(>|t|)
## (Intercept)          0.73926       0.01331  55.537 < 2e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))1 -0.08861       0.02504   -3.539  0.00044 ***
## bs(dis, df = 4, knots = c(4, 7, 11))2 -0.31341       0.01680  -18.658 < 2e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))3 -0.26618       0.03147   -8.459  3.00e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))4 -0.39802       0.04647  -8.565  < 2e-16 ***
## bs(dis, df = 4, knots = c(4, 7, 11))5 -0.25681       0.09001   -2.853   0.00451 **
## bs(dis, df = 4, knots = c(4, 7, 11))6 -0.32926       0.06327  -5.204  2.85e-07 ***
##
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06185 on 499 degrees of freedom
## Multiple R-squared:  0.7185, Adjusted R-squared:  0.7151
## F-statistic: 212.3 on 6 and 499 DF,  p-value: < 2.2e-16

sp.pred = predict(sp.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, sp.pred, col = "red", lwd = 2)
(d). Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```r
all.cv = rep(NA, 16)
for (i in 3:16) {
  lm.fit = lm(nox ~ bs(dis, df = i), data = Boston)
  all.cv[i] = sum(lm.fit$residuals^2)
}
all.cv[-c(1, 2)]
## [1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535 1.796992
## [10] 1.788999 1.782350 1.781838 1.782798 1.783546

lm.fit = lm(nox ~ bs(dis, df = 14), data = Boston)
lm.pred = predict(lm.fit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston, col = "darkgrey")
lines(dis.grid, lm.pred, col = "red", lwd = 2)
```

```r
lm.fit = lm(nox ~ bs(dis, df = 5), data = Boston)
lm.pred = predict(lm.fit, list(dis = dis.grid))
lines(dis.grid, lm.pred, col = "blue", lwd = 2)
```
The lowest value appears to be with df=14, however I might choose df=5 or 6 since there isn’t much difference in the test MSE. Note that the red curve appears much more flexible than the blue, and is likely capturing noise.