Stats 202 Homework 6 Solutions

Problem 1

Solution Credit: Weimu Lei

Suppose that a curve $\hat{g}$ is computed to smoothly fit a set of n points using the following formula:

$$\hat{g} = \arg\min_{g} \left(\sum_{i=1}^{n} \left(y_i - g\left(x_i\right) \right)^2 + \lambda \int \left[g^{(m)}\left(x\right) \right]^2 dx \right)$$

where $g^{(m)}$ represents the m^{th} derivative of g (and $g^{(0)} = g$). Provide example sketches of $\hat{g}$ in each of the following scenarios.

(a) $\lambda = \infty$, $m = 0$.

(b) $\lambda = \infty$, $m = 1$.

(c) $\lambda = \infty$, $m = 2$.

(d) $\lambda = \infty$, $m = 3$.

(e) $\lambda = 0$, $m = 3$.

(a) When $\lambda = \infty$ and $m = 0$, the penalty term completely dominates the function identifying process. The resulting function $\hat{g}$ minimizes squared value of $g\left(x_i\right)$. This means the value of this line is 0 everywhere. Essentially such function $\hat{g}$ maps x to 0, which gives a horizontal line that passes through 0.

(b) With $\lambda = \infty$ and $m = 1$, the resulting $\hat{g}$ minimizes squared value of $g'\left(x_i\right)$. This means the slope of the tangent line at every testing points is 0. Therefore such function gives a horizontal line. The loss function amounts to minimizing RSS, the value of the horizontal line is the response mean $\overline{y}$.

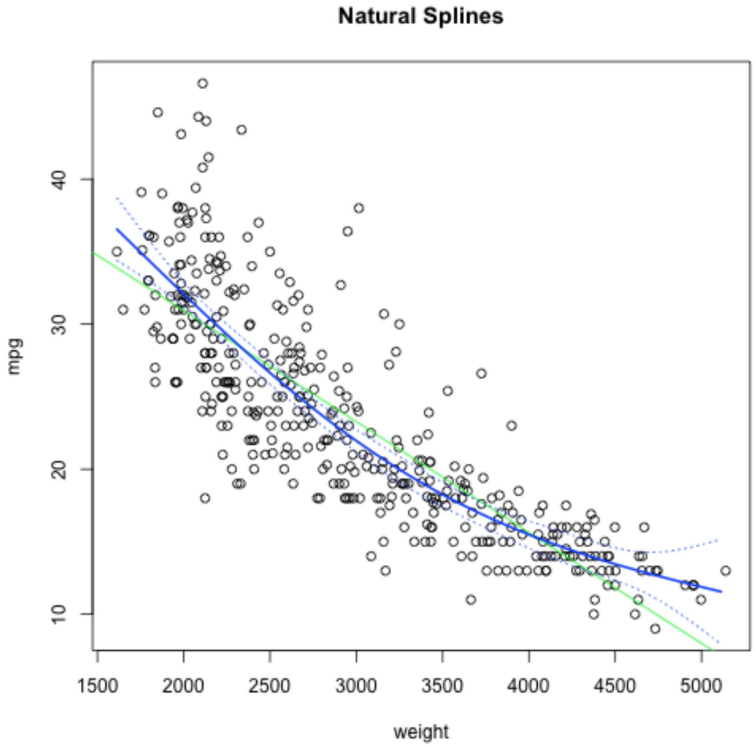
(c) With $\lambda = \infty$ and $m = 2$, the resulting $\hat{g}$ minimizes squared value of $g''\left(x_i\right)$. This means the smoothness of this line is 0 at training points. Therefore such function gives a straight line $(ax + b)$ that passes as closely as possible to the training points. The restriction from the loss function makes it the linear least squares line.

(d) With $\lambda = \infty$ and $m = 3$, the resulting $\hat{g}$ minimizes squared value of $g^{(3)}\left(x_i\right)$. This means the third derivative of the function is 0 at training points. Therefore such function gives a quadratic line $(ax^2 + bx + c)$ that passes as closely as possible to the training points. The restriction from the loss function makes it equivalent to fitting a degree-2 polynomial using least squares.

(e) When $\lambda = 0$, the penalty term would not affect the resulting function any more. The loss function part compeltely determines the function. Therefore such function will exactly interpolate the training observations. That is, $\hat{g}\left(x_i\right) = y_i$.

Problem 2

```
library(ISLR)
library(splines)
auto.lm.fit = lm(mpg ~ weight, data = Auto)
# Natural Splines
auto.ns.fit = lm(mpg ~ ns(weight, knots = c(2400, 3200, 4500)), data = Auto)
wt.grid = seq(min(Auto$weight), max(Auto$weight), 50)
plot(mpg ~ weight, data = Auto, main = "Natural Splines")
abline(auto.lm.fit, col = "green")
ns.pred = predict(auto.ns.fit, newdata = list(weight = wt.grid), se = T)
se.bands = cbind(ns.pred$fit + 2 * ns.pred$se.fit, ns.pred$fit - 2 * ns.pred$se.fit)
lines(wt.grid, ns.pred$fit, lwd = 2, col = "blue")
matlines(wt.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```

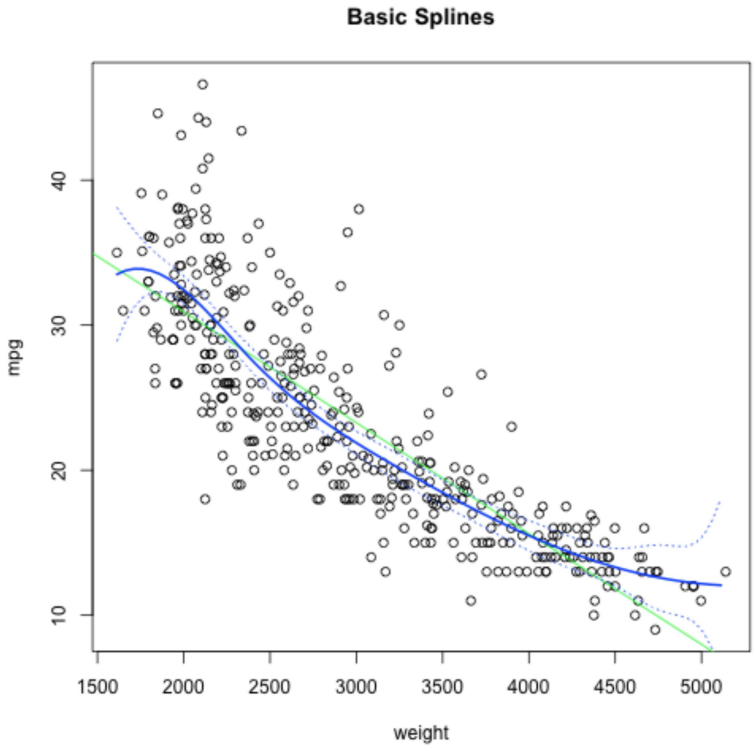



```
anova(auto.lm.fit, auto.ns.fit)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ weight
## Model 2: mpg ~ ns(weight, knots = c(2400, 3200, 4500))
##   Res.Df  RSS Df Sum of Sq    F  Pr(>F)
## 1    390 7321
## 2    387 6779  3       542 10.3 1.5e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value comparing the linear model to the natural splines model is $\approx 10^{-6}$ indicating that a linear model is not justified in this case. The non-linear relation is also evident from the plot.

```
# Basic Splines
auto.bs.fit = lm(mpg ~ bs(weight, knots = c(2400, 3200, 4500)), data = Auto)
plot(mpg ~ weight, data = Auto, main = "Basic Splines")
abline(auto.lm.fit, col = "green")
bs.pred = predict(auto.bs.fit, newdata = list(weight = wt.grid), se = T)
se.bands = cbind(bs.pred$fit + 2 * bs.pred$se.fit, bs.pred$fit - 2 * bs.pred$se.fit)
lines(wt.grid, bs.pred$fit, lwd = 2, col = "blue")
matlines(wt.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```




```
anova(auto.lm.fit, auto.bs.fit)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ weight
## Model 2: mpg ~ bs(weight, knots = c(2400, 3200, 4500))
##   Res.Df  RSS Df Sum of Sq    F  Pr(>F)
## 1    390 7321
## 2    385 6739  5       582 6.65 5.8e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value comparing the linear model to the basic splines model is $< 10^{-5}$. This is not as high as in the case of a natural splines model. Also in comparison, we notice that the confidence in the fit at the end points is not as good. So Natural splines is the better choice, as expected.

Problem 3

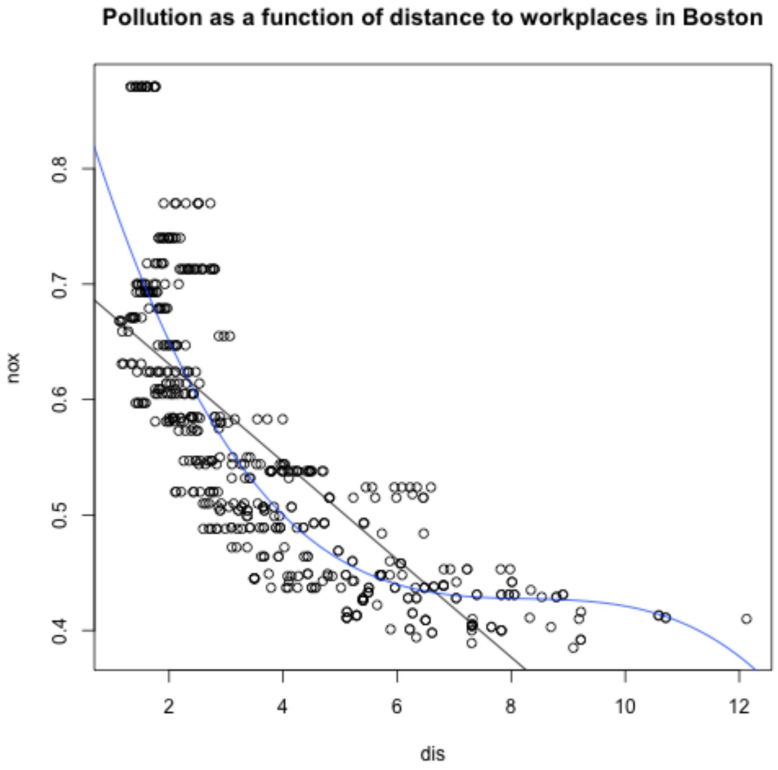
a)

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

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12113 -0.04062 -0.00974  0.02338  0.19490
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.55470    0.00276  201.02  < 2e-16 ***
## poly(dis, 3)1 -2.00310    0.06207  -32.27  < 2e-16 ***
## poly(dis, 3)2  0.85633    0.06207   13.80  < 2e-16 ***
```

```
## poly(dis, 3)3 -0.31805    0.06207   -5.12  4.3e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0621 on 502 degrees of freedom
## Multiple R-squared:  0.715,  Adjusted R-squared:  0.713
## F-statistic:  419 on 3 and 502 DF,  p-value: <2e-16
```

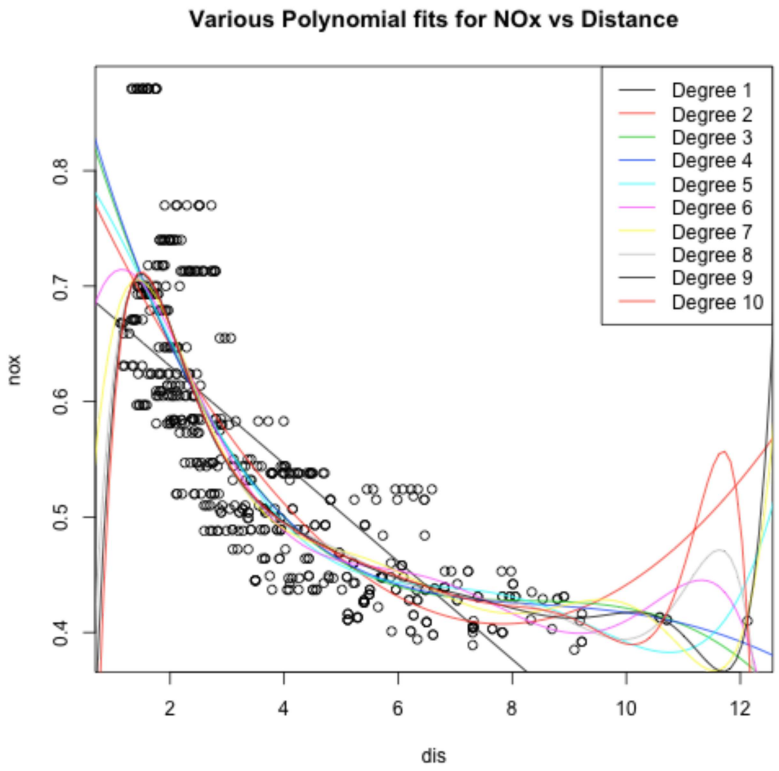
```
plot(nox ~ dis, data = Boston, main = "Pollution as a function of distance to workplaces in Boston")
dis.grid = seq(min(Boston$dis) - 1, max(Boston$dis) + 1, 0.1)
lines(dis.grid, predict(bos.poly.fit, newdata = list(dis = dis.grid)), col = "blue")
abline(lm(nox ~ dis, data = Boston))
```



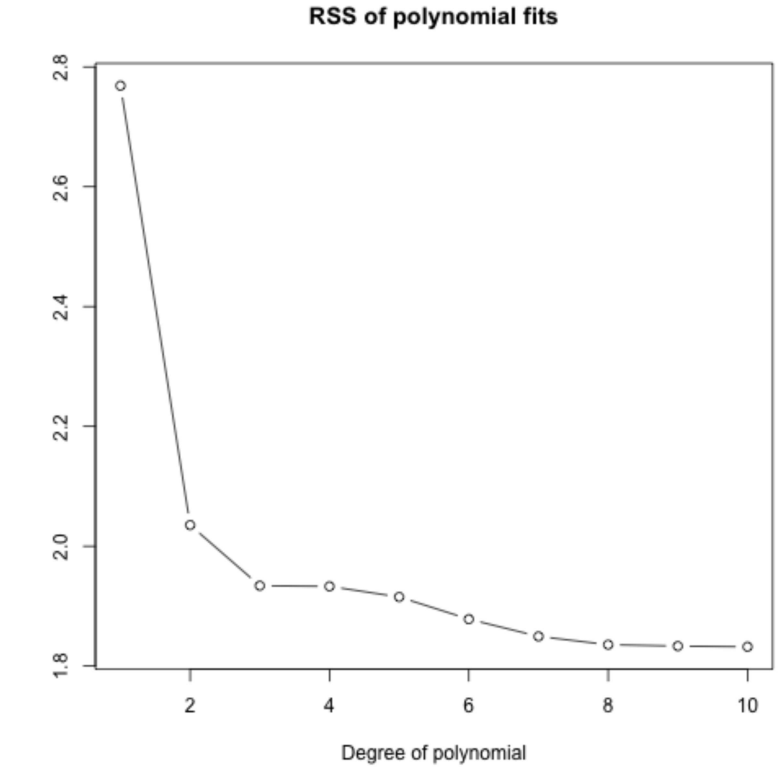
As we can see the cubic polynomial fits the data much better.

b)

```
rss = 0 * (1:10)
plot(nox ~ dis, data = Boston, main = "Various Polynomial fits for NOx vs Distance")
fits = list()
for (i in 1:10) {
    fits[[i]] = lm(nox ~ poly(dis, i), data = Boston)
    rss[i] = sum(fits[[i]]$residuals^2)
    lines(dis.grid, predict(fits[[i]], newdata = list(dis = dis.grid)), col = i)
}
legend("topright", legend = sapply(1:10, sprintf, fmt = "Degree %d"), col = 1:10,
    lty = 1)
```




```
plot(1:10, rss, typ = "b", main = "RSS of polynomial fits", xlab = "Degree of polynomial",
    ylab = "")
```



c) We could select a model using cross-validation. Here, we apply ANOVA (analysis of variance), a method that is less familiar but also useful. The function `anova` tests each model in a sequence to the preceding model and outputs a *p*-value.

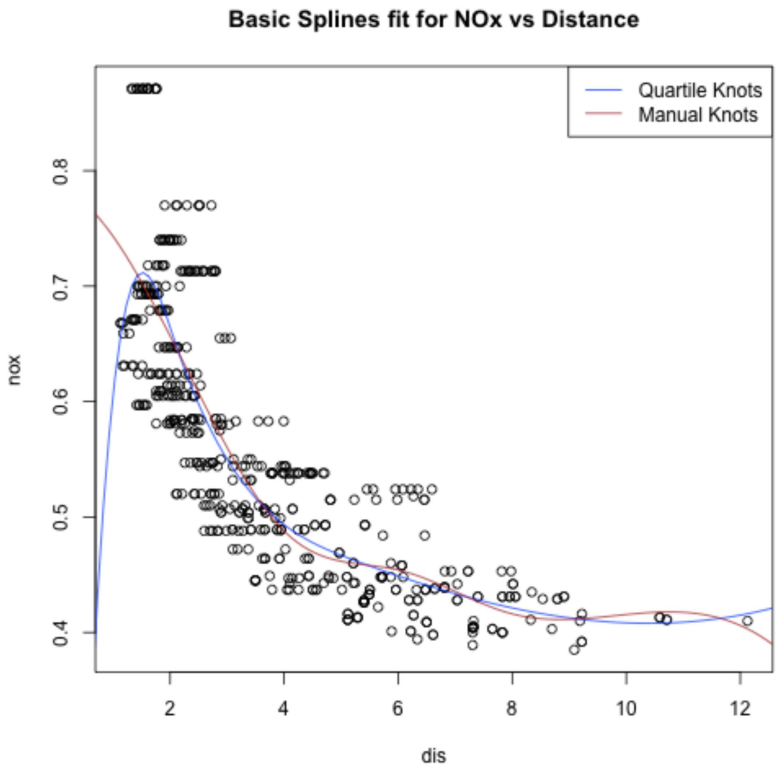
```
anova(fits[[1]], fits[[2]], fits[[3]], fits[[4]], fits[[5]], fits[[6]], fits[[7]],
    fits[[8]], fits[[9]], fits[[10]])
```

```
## Analysis of Variance Table
##
## Model  1: nox ~ poly(dis, i)
## Model  2: nox ~ poly(dis, i)
## Model  3: nox ~ poly(dis, i)
## Model  4: nox ~ poly(dis, i)
## Model  5: nox ~ poly(dis, i)
## Model  6: nox ~ poly(dis, i)
## Model  7: nox ~ poly(dis, i)
## Model  8: nox ~ poly(dis, i)
## Model  9: nox ~ poly(dis, i)
## Model 10: nox ~ poly(dis, i)
##    Res.Df  RSS Df Sum of Sq      F  Pr(>F)
## 1     504 2.77
## 2     503 2.04  1     0.733 198.12 < 2e-16 ***
## 3     502 1.93  1     0.101  27.33 2.5e-07 ***
## 4     501 1.93  1     0.001   0.30  0.5816
## 5     500 1.92  1     0.018   4.78  0.0293 *
## 6     499 1.88  1     0.037  10.01  0.0017 **
## 7     498 1.85  1     0.029   7.77  0.0055 **
## 8     497 1.84  1     0.014   3.74  0.0536 .
## 9     496 1.83  1     0.002   0.62  0.4310
## 10    495 1.83  1     0.001   0.31  0.5759
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It can be seen that from the linear model to the quadratic and from the quadratic to the cubic the p-value is very close to zero. Hence the cubic polynomial model is justified where as the linear and quadratic are not. But from the cubic to the fourth-degree a p-value of .58 is anything but significant. Hence we stop at the cubic model. This makes sense from the Plot too, where the third and fourth degree polynomials give a much better representation of the data than the degree two or degree five polynomials.

d)

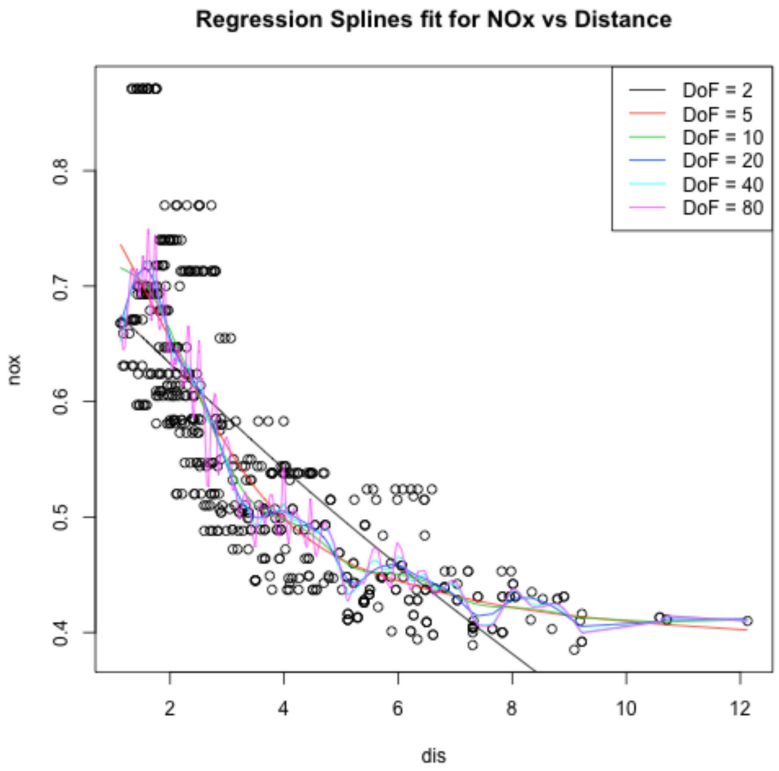
```
quartiles = summary(Boston$dis)[c(2, 3, 5)]
bos.bs.fit1 = lm(nox ~ bs(dis, knots = quartiles), data = Boston)
plot(nox ~ dis, data = Boston, main = "Basic Splines fit for NOx vs Distance")
lines(dis.grid, predict(bos.bs.fit1, newdata = list(dis = dis.grid)), col = "blue")
bos.bs.fit2 = lm(nox ~ bs(dis, knots = c(4, 6, 8)), data = Boston)
lines(dis.grid, predict(bos.bs.fit2, newdata = list(dis = dis.grid)), col = "brown")
legend("topright", leg = c("Quartile Knots", "Manual Knots"), col = c("blue",
    "brown"), lty = 1)
```



We fit a spline with 3 knots, or 7 degrees of freedom. The knots were chosen in two ways: at the quartiles of the distribution, as well as chosen manually to match the inflection points of the scatter plot. The latter method seems to produce a better fit, because we are incurring in data snooping.

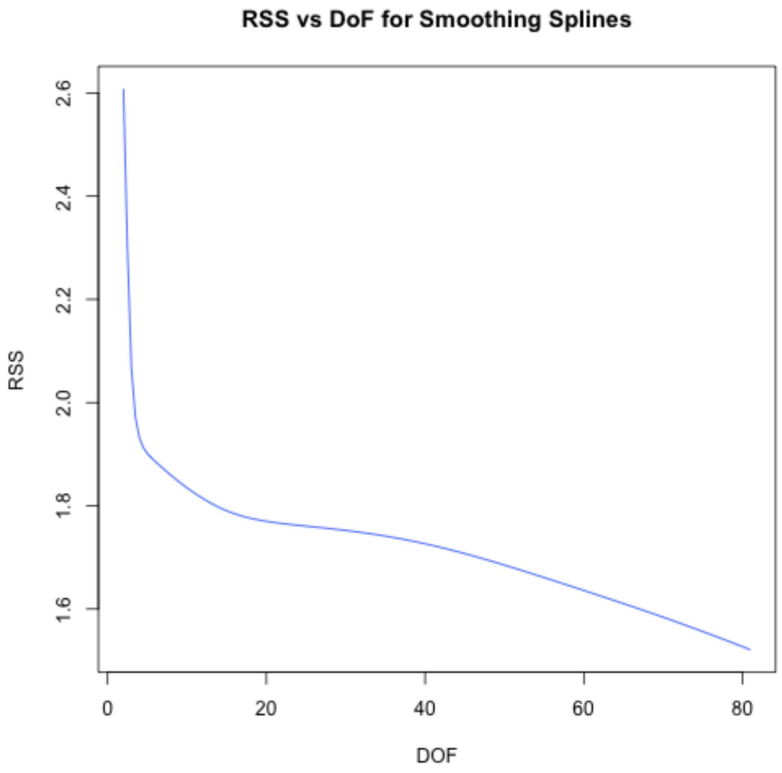
e)

```
ss.fits = list()
dfs = c(2, 5, 10, 20, 40, 80)
plot(nox ~ dis, data = Boston, main = "Regression Splines fit for NOx vs Distance")
for (i in 1:length(dfs)) {
    ss.fits[[i]] = smooth.spline(Boston$dis, Boston$nox, df = dfs[i])
    lines(ss.fits[[i]], col = i, lty = 1)
}
legend("topright", leg = sapply(dfs, sprintf, fmt = "DoF = %d"), col = 1:length(dfs),
    lty = 1)
```



The fit becomes more and more wiggly as the degrees of freedom increase. When the degrees of freedom is two, we have a linear model fit.

```
moredfs = seq(2, 81, 0.5)
ss.rss = sapply(moredfs, function(df) sum(residuals(smooth.spline(Boston$dis,
    Boston$nox, df = df))^2))
plot(moredfs, ss.rss, main = "RSS vs DoF for Smoothing Splines", typ = "l",
    col = "blue", xlab = "DOF", ylab = "RSS")
```



As expected, the RSS decreases with increasing degrees of freedom.

f)

```
set.seed(2)
perm = sample(nrow(Boston), nrow(Boston))
nfolds = 5
ss.cv.1fold.err <- function(fold, sz, df) {
    idx = perm[(sz * (fold - 1) + 1):(sz * fold)]
    this.fit <- smooth.spline(Boston$dis[-idx], Boston$nox[-idx], df = df)
    mean((predict(this.fit, Boston$dis[idx])$y - Boston$nox[idx])^2)
}
ss.cv.nfold.err <- function(df, n = nfolds) mean(sapply(1:n, ss.cv.1fold.err,
    sz = floor(nrow(Boston)/nfolds), df = df))
trydfs = seq(2, 80)
cv.errs = sapply(trydfs, ss.cv.nfold.err)
plot(trydfs, cv.errs, typ = "b", main = "Cross Validation Error vs DoF", xlab = "DoF",
    ylab = "CV Error")
```


```
trydfs[which.min(cv.errs)]
```

```
## [1] 15
```

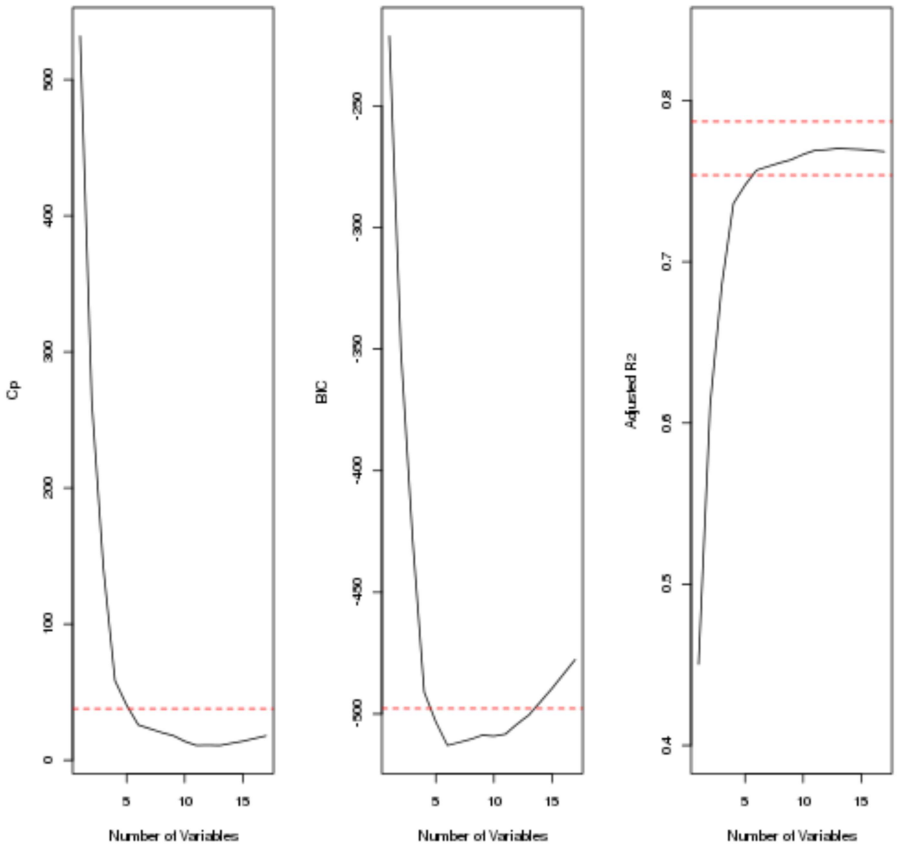
We use 15 degrees of freedom as the cross validation error is minimized there.

Problem 4

Solution credit: Yi Shan

a

```
set.seed(1)
library(ISLR)
library(leaps)
attach(College)
train = sample(length(Outstate), length(Outstate)/2)
test = -train
College.train = College[train, ]
College.test = College[test, ]
reg.fit = regsubsets(Outstate ~ ., data = College.train, nvmax = 17, method =
"forward")
reg.summary = summary(reg.fit)
par(mfrow = c(1, 3))
plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
min.cp = min(reg.summary$cp)
std.cp = sd(reg.summary$cp)
abline(h = min.cp + 0.2 * std.cp, col = "red", lty = 2)
abline(h = min.cp - 0.2 * std.cp, col = "red", lty = 2)
plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
min.bic = min(reg.summary$bic)
std.bic = sd(reg.summary$bic)
abline(h = min.bic + 0.2 * std.bic, col = "red", lty = 2)
abline(h = min.bic - 0.2 * std.bic, col = "red", lty = 2)
plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R2",
   type = "l", ylim = c(0.4, 0.84))
max.adjr2 = max(reg.summary$adjr2)
std.adjr2 = sd(reg.summary$adjr2)
abline(h = max.adjr2 + 0.2 * std.adjr2, col = "red", lty = 2)
abline(h = max.adjr2 - 0.2 * std.adjr2, col = "red", lty = 2)
```



All cp, BIC and adjr2 scores show that size 6 is the minimum size for the subset for which the scores are withing 0.2 standard deviations of optimum. We pick 6 as the best subset size and find best 6 variables using entire data.

```
reg.fit = regsubsets(Outstate ~ ., data = College, method = "forward")
coefi = coef(reg.fit, id = 6)
names(coefi)
```

```
## [1] "(Intercept)" "PrivateYes"  "Room.Board"  "PhD"         "perc.alumni"
## [6] "Expend"      "Grad.Rate"
```

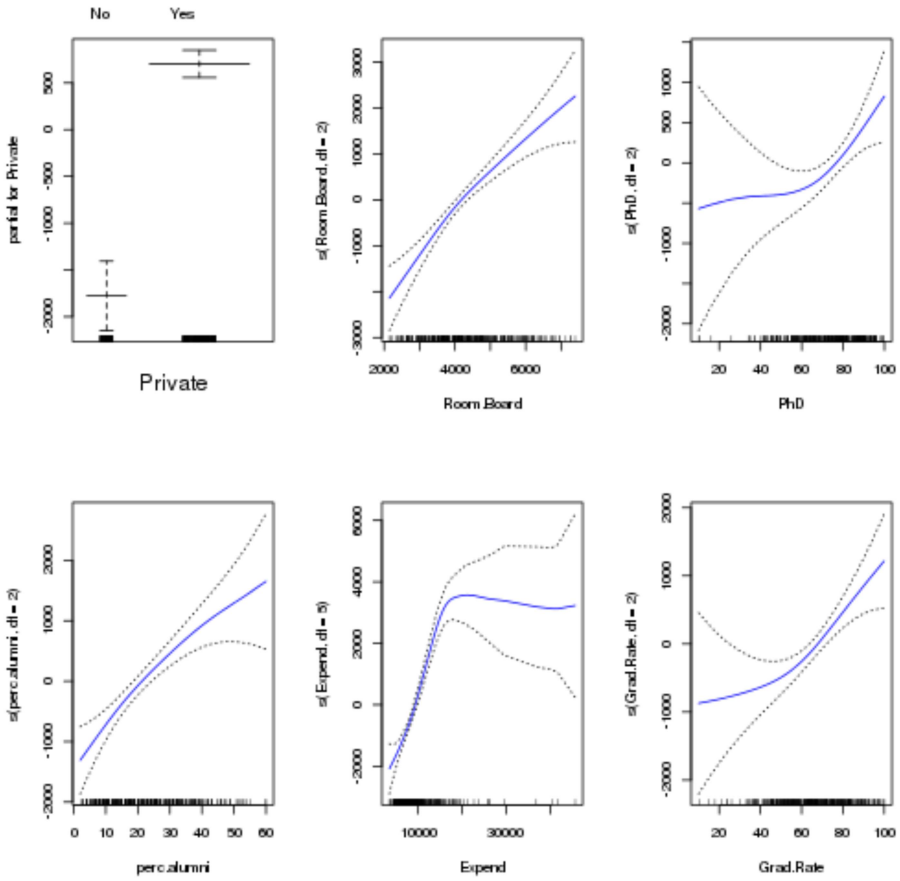
b

```
library(gam)
```

```
## Loading required package: splines
## Loaded gam 1.09
```

```
gam.fit = gam(Outstate ~ Private + s(Room.Board, df = 2) + s(PhD, df = 2) +
    s(perc.alumni, df = 2) + s(Expend, df = 5) + s(Grad.Rate, df = 2), data = College.train)
par(mfrow = c(2, 3))
```

plot(gam.fit, se = T, col = "blue")



c

gam.pred = predict(gam.fit, College.test)
gam.err = mean((College.test$Outstate - gam.pred)^2)
gam.err

[1] 3745460

gam.tss = mean((College.test$Outstate - mean(College.test$Outstate))^2)
test.rss = 1 - gam.err/gam.tss
test.rss

[1] 0.7697

We obtain a test R-squared of 0.77 using GAM with 6 predictors. This is a slight improvement over a test RSS of 0.74 obtained using OLS.

d

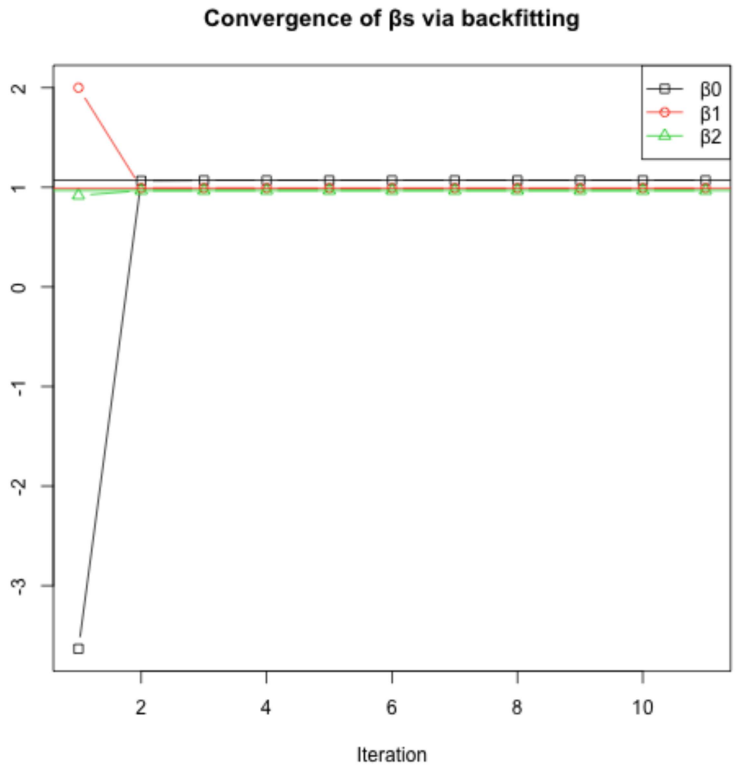
```
summary(gam.fit)

##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, df = 2) + s(PhD,
##     df = 2) + s(perc.alumni, df = 2) + s(Expend, df = 5) + s(Grad.Rate,
##     df = 2), data = College.train)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -4977.7 -1184.5    58.3  1220.0  7688.3
##
## (Dispersion Parameter for gaussian family taken to be 3300711)
##
##     Null Deviance: 6.222e+09 on 387 degrees of freedom
## Residual Deviance: 1.231e+09 on 373 degrees of freedom
## AIC: 6942
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                     Df   Sum Sq  Mean Sq F value  Pr(>F)
## Private              1 1.78e+09 1.78e+09   539.1 < 2e-16 ***
## s(Room.Board, df = 2)    1 1.22e+09 1.22e+09   370.2 < 2e-16 ***
## s(PhD, df = 2)           1 3.82e+08 3.82e+08   115.9 < 2e-16 ***
## s(perc.alumni, df = 2)   1 3.28e+08 3.28e+08    99.5 < 2e-16 ***
## s(Expend, df = 5)        1 4.17e+08 4.17e+08   126.2 < 2e-16 ***
## s(Grad.Rate, df = 2)     1 5.53e+07 5.53e+07    16.8 5.2e-05 ***
## Residuals              373 1.23e+09 3.30e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                     Npar Df Npar F Pr(F)
## (Intercept)
## Private
## s(Room.Board, df = 2)        1   3.56 0.060 .
## s(PhD, df = 2)               1   4.34 0.038 *
## s(perc.alumni, df = 2)       1   1.92 0.167
## s(Expend, df = 5)            4  16.86 1e-12 ***
## s(Grad.Rate, df = 2)         1   3.72 0.055 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Non-parametric Anova test shows a strong evidence of non-linear relationship between response and Expend, and a moderately strong non-linear relationship (using p value of 0.05) between response and Grad.Rate or PhD.

Problem 5

```
# a
n = 100
x1 = 10 * runif(n)
x2 = 5 * runif(n)
y = 1 + x1 + x2 + rnorm(n)
# b
Î²1 = 2
# c
a = y - Î²1 * x1
Î²2 = lm(a ~ x2)$coef[2]
Î²0 = lm(a ~ x2)$coef[1]
Î²s = c(Î²0, Î²1, Î²2)

# d,e
for (i in 1:10) {
    a = y - Î²2 * x2
    Î²1 = lm(a ~ x1)$coef[2]
    a = y - Î²1 * x1
    Î²2 = lm(a ~ x2)$coef[2]
    Î²0 = lm(a ~ x2)$coef[1]
    Î²s = cbind(Î²s, c(Î²0, Î²1, Î²2))
}
matplot(t(Î²s), typ = "b", lty = 1, main = "Convergence of Î²s via backfitting",
    ylab = "", xlab = "Iteration", pch = 0:2)
legend("topright", leg = c("Î²0", "Î²1", "Î²2"), col = 1:3, lty = 1, pch = 0:2)
Î²s[, 11]

## (Intercept)                      x2
##      1.0699      0.9929      0.9667
```

```
# f
mult.fit = lm(y ~ x1 + x2)
mult.fit$coefficients

## (Intercept)          x1          x2
##      1.0699      0.9929      0.9667

abline(h = mult.fit$coefficients[1], col = 1)
abline(h = mult.fit$coefficients[2], col = 2)
abline(h = mult.fit$coefficients[3], col = 3)
```



g) For this data set we could get a very good fit within four iterations only! The backfitting algorithm, which consists of iteratively fitting each term in an additive model, converges to the same result as simultaneously fitting all the terms. For some additive models, such as the one in this example (multiple linear regression), one can prove analytically that this convergence holds.