An Introduction to Statistical Learning

## 6. Moving Beyond Linearity

### Conceptual

#### **1.** It was mentioned in the chapter that a cubic regression spline with one knot at can be obtained using a basis of the form , where if and equals otherwise. We will now show that a function of the form is indeed a cubic regression spline, regardless of the values of

1. Find a cubic polynomial such that for all . Express in terms of .

Since

,

therefore and

1. Find a cubic polynomial such that for all . Express in terms of .

For :

which we can rearrange to

and therefore have

, , and

1. Show that . That is is continous at .

equals

.

1. Show that . That is is continous at .

equals

1. Show that . That is is continous at . Therefore, is indeed a cubic spline.

equals

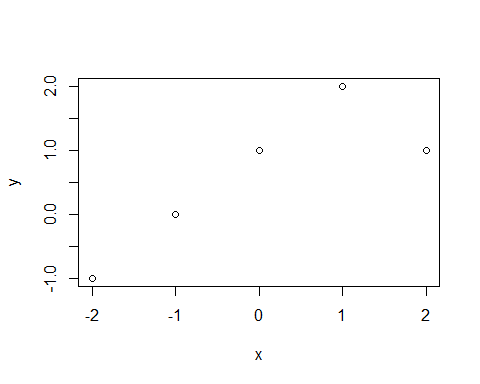
#### **2.** Suppose that a curve is computed to smoothly fit a set of n points using the following formula:

#### where represents the m-th derivative of g(and g(0)=g). Provide example sketches of in each of the following scenarios.

* , straight line through 0
* , straight line with intercept
* , (intercept and slope)
* $g = ax^2 +bx+c
* g will be very jumpy and exactly interpolate the training observations

#### **3.** Suppose we fit a curve with basis functions , . We fit the linear regression model and obtain coefficient estimates , ,. Sketch the estimated curve between and . Note the intercepts, slopes, and other relevant information.

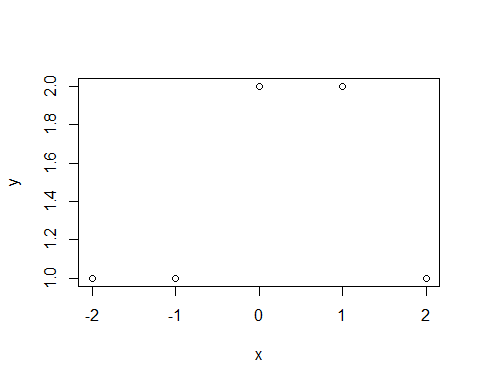
x = -2:2  
y = 1 + x + -2 \* (x-1)^2 \* I(x>1)  
plot(x, y)



* function is linear for all and quadratic for

#### **4.** Suppose we fit a curve with basis functions , We fit the linear regression model and obtain coefficient estimates , ,. Sketch the estimated curve between and . Note the intercepts, slopes, and other relevant information.

x = -2:2  
y = c(1 + 0 + 0, # x = -2  
 1 + 0 + 0, # x = -1  
 1 + 1 + 0, # x = 0  
 1 + (1-0) + 0, # x = 1  
 1 + (1-1) + 0 # x =2  
 )  
plot(x,y)



* the function is either constant or linear depending on the interval

#### **5.** Consider two curves, and , defined by

#### where represents the m-th derivative of g.

1. As , will or have the smaller training RSS?

* as the more flexible approach will have the lower training error

1. As , will or have the smaller training RSS?

* cannot say for sure, might overfit so could have a lower test error

1. For , will have the smaller training and test RSS?

* for , and they therefore have the same training and test errors

### Applied

#### **6.** In this exercise, you will further analyze the Wage data set considered throughout this chapter.

1. Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

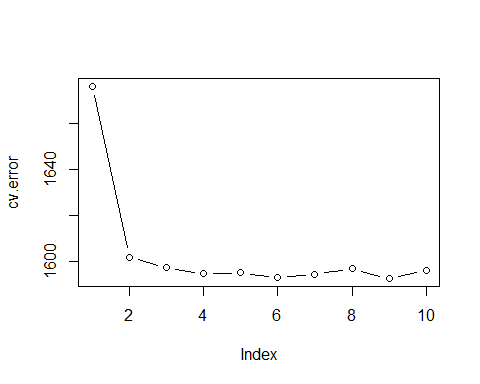
require(ISLR)  
require(boot)

## Loading required package: boot

data(Wage)  
set.seed(42)  
  
  
cv.error <- rep(0,10)  
for (i in 1:10) {  
 glm.fit <- glm(wage~poly(age,i), data=Wage)  
 cv.error[i] <- cv.glm(Wage, glm.fit, K=10)$delta[1]   
}  
cv.error

## [1] 1676.334 1601.952 1597.313 1594.688 1595.061 1592.922 1594.542 1596.803  
## [9] 1592.672 1596.282

plot(cv.error, type="b")

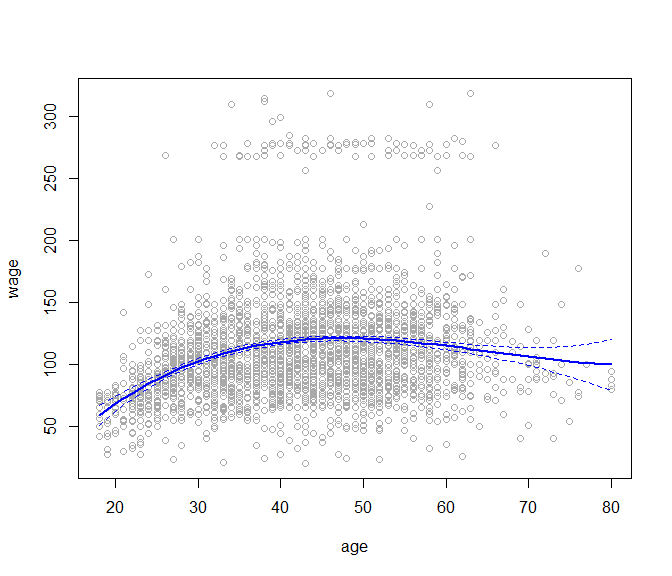


fit.01 <- lm(wage~age, data=Wage)  
fit.02 <- lm(wage~poly(age,2), data=Wage)  
fit.03 <- lm(wage~poly(age,3), data=Wage)  
fit.04 <- lm(wage~poly(age,4), data=Wage)  
fit.05 <- lm(wage~poly(age,5), data=Wage)  
fit.06 <- lm(wage~poly(age,6), data=Wage)  
fit.07 <- lm(wage~poly(age,7), data=Wage)  
fit.08 <- lm(wage~poly(age,8), data=Wage)  
fit.09 <- lm(wage~poly(age,9), data=Wage)  
fit.10 <- lm(wage~poly(age,10), data=Wage)  
anova(fit.01,fit.02,fit.03,fit.04,fit.05,fit.06,fit.07,fit.08,fit.09,fit.10)

## 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)  
## Model 6: wage ~ poly(age, 6)  
## Model 7: wage ~ poly(age, 7)  
## Model 8: wage ~ poly(age, 8)  
## Model 9: wage ~ poly(age, 9)  
## Model 10: wage ~ poly(age, 10)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 2998 5022216   
## 2 2997 4793430 1 228786 143.7638 < 2.2e-16 \*\*\*  
## 3 2996 4777674 1 15756 9.9005 0.001669 \*\*   
## 4 2995 4771604 1 6070 3.8143 0.050909 .   
## 5 2994 4770322 1 1283 0.8059 0.369398   
## 6 2993 4766389 1 3932 2.4709 0.116074   
## 7 2992 4763834 1 2555 1.6057 0.205199   
## 8 2991 4763707 1 127 0.0796 0.777865   
## 9 2990 4756703 1 7004 4.4014 0.035994 \*   
## 10 2989 4756701 1 3 0.0017 0.967529   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

* Based on Anova 2 or 3 degree poly performs best, compared to 4th degree with cv.

attach(Wage)  
agelims=range(age)  
age.grid=seq(from=agelims[1],to=agelims[2])  
fit <- lm(wage ~ poly(age, 3), data = Wage)  
preds=predict(fit,newdata=list(age=age.grid),se=TRUE)  
se.bands=cbind(preds$fit+2\*preds$se,preds$fit-2\*preds$se)  
plot(age,wage,col="darkgrey")  
lines(age.grid,preds$fit,lwd=2,col="blue")  
matlines(age.grid,se.bands,col="blue",lty=2)

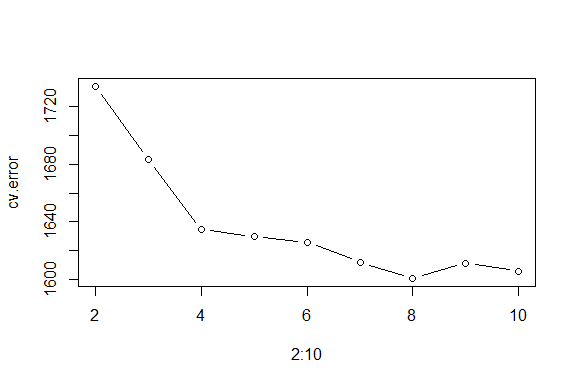


1. Fit a step function to predict wage using age, and perform cross-validation to choose the optimal number of cuts. Make a plot of the fit obtained.

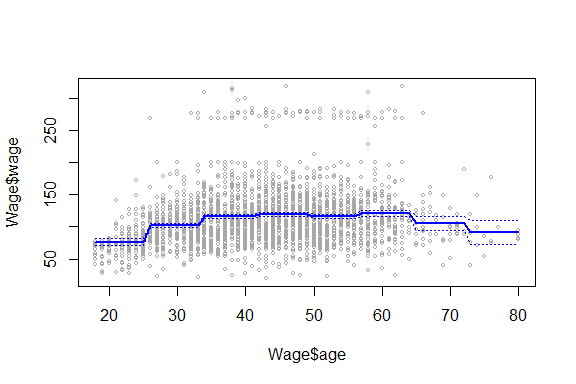
cv.error <- rep(0,9)  
for (i in 2:10) {  
 Wage$age.cut <- cut(Wage$age,i)  
 glm.fit <- glm(wage~age.cut, data=Wage)  
 cv.error[i-1] <- cv.glm(Wage, glm.fit, K=10)$delta[1]  
}  
cv.error

## [1] 1733.666 1683.238 1634.605 1630.162 1625.805 1612.094 1601.004 1611.510  
## [9] 1606.099

plot(2:10, cv.error, type="b")



cut.fit <- glm(wage~cut(age,8), data=Wage)  
preds <- predict(cut.fit, newdata=list(age=age.grid), se=TRUE)  
se.bands <- preds$fit + cbind(2\*preds$se.fit, -2\*preds$se.fit)  
plot(Wage$age, Wage$wage, xlim=agelims, cex=0.5, col="darkgrey")  
lines(age.grid, preds$fit, lwd=2, col="blue")  
matlines(age.grid, se.bands, lwd=1, col="blue", lty=3)



#### **7.** The Wage data set contains a number of other features not explored in this chapter, such as marital status (maritl), job class (jobclass),and others. Explore the relationships between some of these other predictors and wage, and use non-linear fitting techniques in order to fit flexible models to the data. Create plots of the results obtained, and write a summary of your findings.

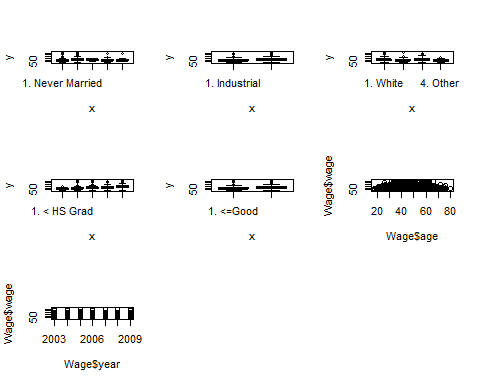
library(gam)

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.16.1

par(mfrow=c(3,3))  
plot(Wage$maritl, Wage$wage)  
plot(Wage$jobclass, Wage$wage)  
plot(Wage$race, Wage$wage)  
plot(Wage$education, Wage$wage)  
plot(Wage$health, Wage$wage)  
plot(Wage$age, Wage$wage)  
plot(Wage$year, Wage$wage)



library(gam)  
gam.fit1 = gam(wage~s(age,2)+year+education, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit2 = gam(wage~s(age,2)+year+education+maritl, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit3 = gam(wage~s(age,2)+year+education+maritl+race, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit4 = gam(wage~s(age,2)+year+education+maritl+race+ health+jobclass, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit5 = gam(wage~s(age,2)+year+education+maritl+ health+jobclass, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit6 = gam(wage~s(age,2)+year+education+race+health, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit7 = gam(wage~s(age,3)+year+education+race+health, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit8 = gam(wage~s(age,4)+year+education+race+health, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

gam.fit9 = gam(wage~s(age,5)+year+education+race+health, data=Wage)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

anova(gam.fit1, gam.fit2, gam.fit3, gam.fit4, gam.fit5, gam.fit6, gam.fit7, gam.fit8, gam.fit9, test= "F" )

## Analysis of Deviance Table  
##   
## Model 1: wage ~ s(age, 2) + year + education  
## Model 2: wage ~ s(age, 2) + year + education + maritl  
## Model 3: wage ~ s(age, 2) + year + education + maritl + race  
## Model 4: wage ~ s(age, 2) + year + education + maritl + race + health +   
## jobclass  
## Model 5: wage ~ s(age, 2) + year + education + maritl + health + jobclass  
## Model 6: wage ~ s(age, 2) + year + education + race + health  
## Model 7: wage ~ s(age, 3) + year + education + race + health  
## Model 8: wage ~ s(age, 4) + year + education + race + health  
## Model 9: wage ~ s(age, 5) + year + education + race + health  
## Resid. Df Resid. Dev Df Deviance F Pr(>F)   
## 1 2992 3725514   
## 2 2988 3620098 4.00000 105416 22.0557 < 2.2e-16 \*\*\*  
## 3 2985 3611272 3.00000 8826 2.4621 0.06077 .   
## 4 2983 3564340 2.00000 46933 19.6389 3.362e-09 \*\*\*  
## 5 2986 3574385 -3.00000 -10046 2.8024 0.03849 \*   
## 6 2988 3672490 -2.00000 -98105 41.0520 < 2.2e-16 \*\*\*  
## 7 2987 3650727 0.99992 21763 18.2146 2.036e-05 \*\*\*  
## 8 2986 3645157 1.00021 5570 4.6605 0.03094 \*   
## 9 2985 3642096 0.99952 3061 2.5630 0.10951   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

deviance(gam.fit1)

## [1] 3725514

deviance(gam.fit2)

## [1] 3620098

deviance(gam.fit3)

## [1] 3611272

deviance(gam.fit4)

## [1] 3564340

deviance(gam.fit5)

## [1] 3574385

deviance(gam.fit6)

## [1] 3672490

deviance(gam.fit7)

## [1] 3650727

deviance(gam.fit8)

## [1] 3645157

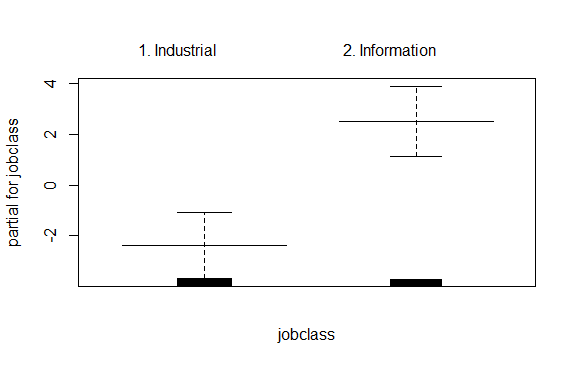
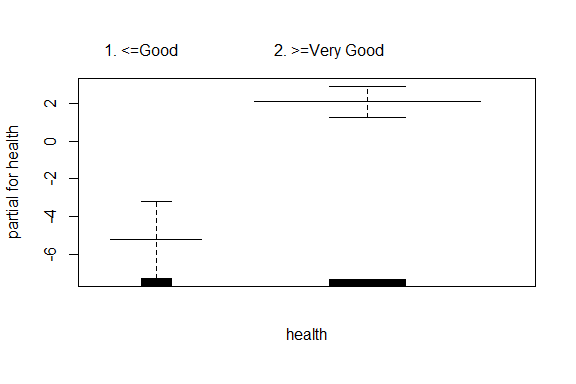
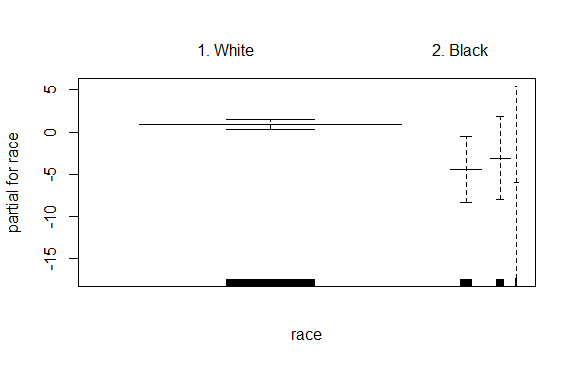
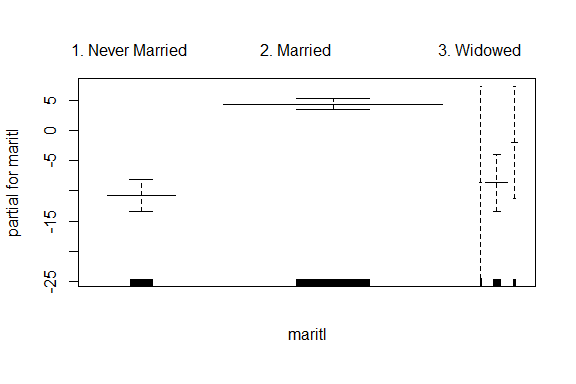
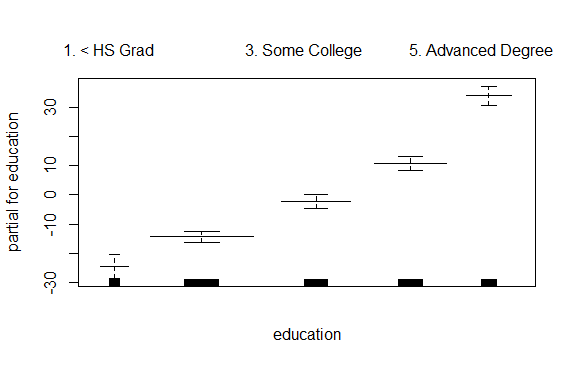
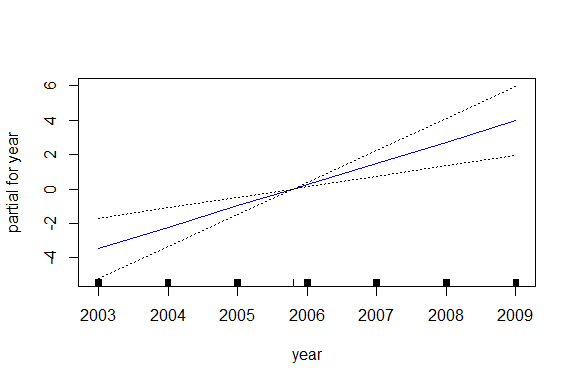
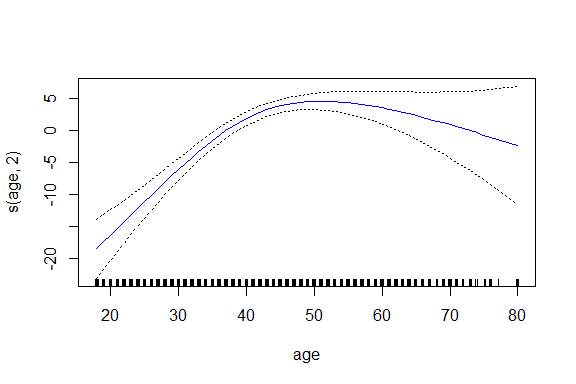
deviance(gam.fit9)

## [1] 3642096

summary(gam.fit4)

##   
## Call: gam(formula = wage ~ s(age, 2) + year + education + maritl +   
## race + health + jobclass, data = Wage)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -109.833 -19.054 -2.868 14.201 212.987   
##   
## (Dispersion Parameter for gaussian family taken to be 1194.884)  
##   
## Null Deviance: 5222086 on 2999 degrees of freedom  
## Residual Deviance: 3564340 on 2983 degrees of freedom  
## AIC: 29790   
##   
## Number of Local Scoring Iterations: 2   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## s(age, 2) 1 199870 199870 167.2711 < 2.2e-16 \*\*\*  
## year 1 19154 19154 16.0303 6.386e-05 \*\*\*  
## education 4 1116690 279173 233.6399 < 2.2e-16 \*\*\*  
## maritl 4 128341 32085 26.8523 < 2.2e-16 \*\*\*  
## race 3 9201 3067 2.5669 0.0528241 .   
## health 1 32201 32201 26.9489 2.229e-07 \*\*\*  
## jobclass 1 15913 15913 13.3176 0.0002674 \*\*\*  
## Residuals 2983 3564340 1195   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## s(age, 2) 1 53.712 2.972e-13 \*\*\*  
## year   
## education   
## maritl   
## race   
## health   
## jobclass   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

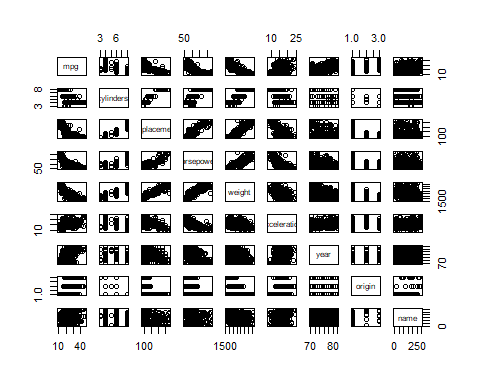
plot(gam.fit4, se=TRUE, col="blue")



#### **8.** Fit some of the non-linear models investigated in this chapter to the Auto data set. Is there evidence for non-linear relationships in this data set? Create some informative plots to justify your answer.

require(boot)  
require(gam)  
require(ISLR)  
data(Auto)  
set.seed(1)

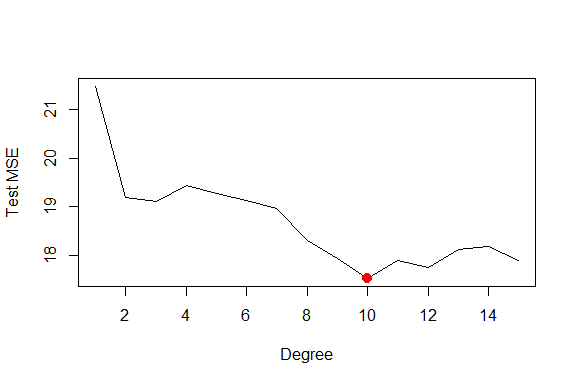
pairs(Auto)



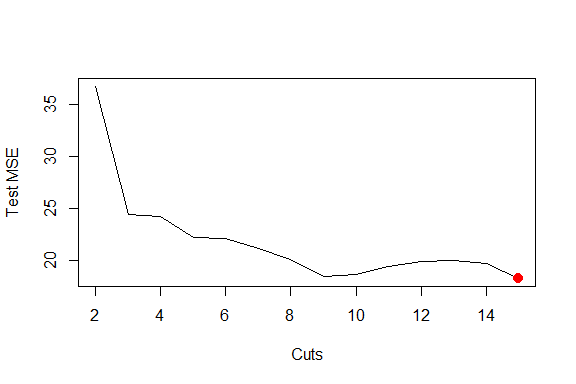
* there should be some nonlinear relationships in this data set

##### **mpg - displacement**

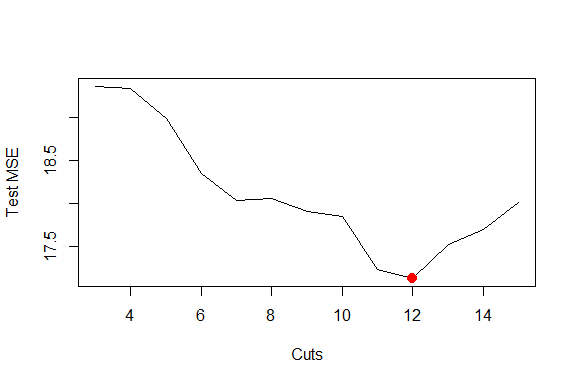
deltas <- rep(NA, 15)  
for (i in 1:15) {  
 fit <- glm(mpg ~ poly(displacement, i), data = Auto)  
 deltas[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(1:15, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")  
d.min <- which.min(deltas)  
points(which.min(deltas), deltas[which.min(deltas)], col = "red", cex = 2, pch = 20)



cvs <- rep(NA, 15)  
for (i in 2:15) {  
 Auto$dis.cut <- cut(Auto$displacement, i)  
 fit <- glm(mpg ~ dis.cut, data = Auto)  
 cvs[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(2:15, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")  
d.min <- which.min(cvs)  
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)

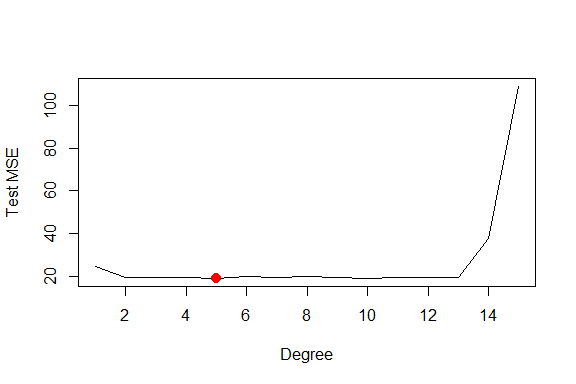


library(splines)  
cvs <- rep(NA, 15)  
for (i in 3:15) {  
 fit <- glm(mpg ~ ns(displacement, df = i), data = Auto)  
 cvs[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(3:15, cvs[-c(1, 2)], xlab = "Cuts", ylab = "Test MSE", type = "l")  
d.min <- which.min(cvs)  
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)

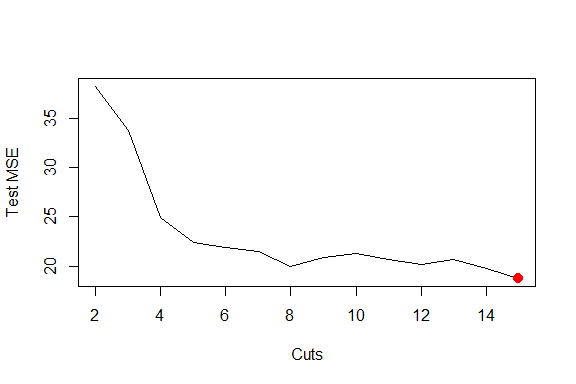


##### **mpg horsepower**

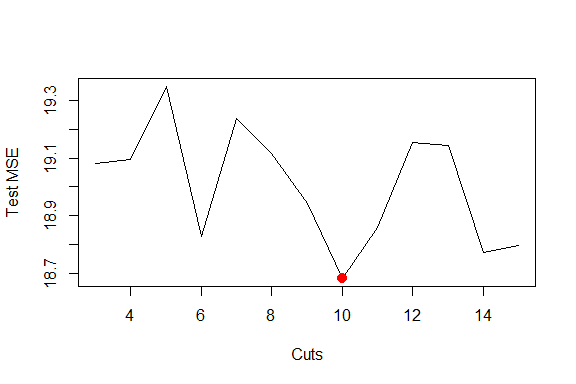
deltas <- rep(NA, 15)  
for (i in 1:15) {  
 fit <- glm(mpg ~ poly(horsepower, i), data = Auto)  
 deltas[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(1:15, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")  
d.min <- which.min(deltas)  
points(which.min(deltas), deltas[which.min(deltas)], col = "red", cex = 2, pch = 20)



cvs <- rep(NA, 15)  
for (i in 2:15) {  
 Auto$dis.cut <- cut(Auto$horsepower, i)  
 fit <- glm(mpg ~ dis.cut, data = Auto)  
 cvs[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(2:15, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")  
d.min <- which.min(cvs)  
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)

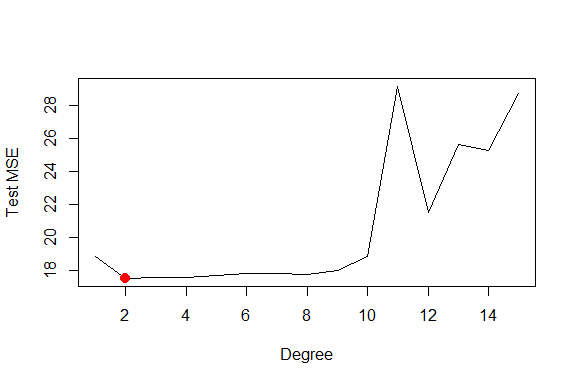


library(splines)  
cvs <- rep(NA, 15)  
for (i in 3:15) {  
 fit <- glm(mpg ~ ns(horsepower, df = i), data = Auto)  
 cvs[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(3:15, cvs[-c(1, 2)], xlab = "Cuts", ylab = "Test MSE", type = "l")  
d.min <- which.min(cvs)  
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)

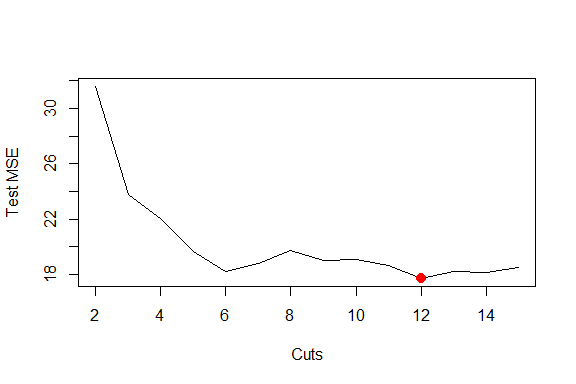


##### **mpg - weight**

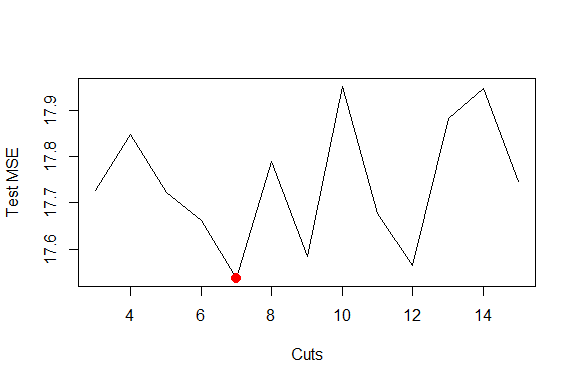
deltas <- rep(NA, 15)  
for (i in 1:15) {  
 fit <- glm(mpg ~ poly(weight, i), data = Auto)  
 deltas[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(1:15, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")  
d.min <- which.min(deltas)  
points(which.min(deltas), deltas[which.min(deltas)], col = "red", cex = 2, pch = 20)



cvs <- rep(NA, 15)  
for (i in 2:15) {  
 Auto$dis.cut <- cut(Auto$weight, i)  
 fit <- glm(mpg ~ dis.cut, data = Auto)  
 cvs[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(2:15, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")  
d.min <- which.min(cvs)  
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)



library(splines)  
cvs <- rep(NA, 15)  
for (i in 3:15) {  
 fit <- glm(mpg ~ ns(weight, df = i), data = Auto)  
 cvs[i] <- cv.glm(Auto, fit, K = 10)$delta[1]  
}  
plot(3:15, cvs[-c(1, 2)], xlab = "Cuts", ylab = "Test MSE", type = "l")  
d.min <- which.min(cvs)  
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)



##### **GAMs**

fit = gam(mpg ~ s(displacement, 11) + s(horsepower, 10)+ s(weight, 7)+ acceleration, data = Auto)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

deviance(fit)

## [1] 5068.748

summary(fit)

##   
## Call: gam(formula = mpg ~ s(displacement, 11) + s(horsepower, 10) +   
## s(weight, 7) + acceleration, data = Auto)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -9.9543 -1.8508 -0.2105 1.7793 16.2185   
##   
## (Dispersion Parameter for gaussian family taken to be 14.002)  
##   
## Null Deviance: 23818.99 on 391 degrees of freedom  
## Residual Deviance: 5068.748 on 362.0007 degrees of freedom  
## AIC: 2177.805   
##   
## Number of Local Scoring Iterations: 2   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## s(displacement, 11) 1 15063.8 15063.8 1075.8314 < 2.2e-16 \*\*\*  
## s(horsepower, 10) 1 1140.5 1140.5 81.4535 < 2.2e-16 \*\*\*  
## s(weight, 7) 1 295.2 295.2 21.0825 6.079e-06 \*\*\*  
## acceleration 1 111.2 111.2 7.9406 0.005099 \*\*   
## Residuals 362 5068.7 14.0   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## s(displacement, 11) 10 3.6826 0.0001044 \*\*\*  
## s(horsepower, 10) 9 6.8397 4.253e-09 \*\*\*  
## s(weight, 7) 6 0.5359 0.7809000   
## acceleration   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

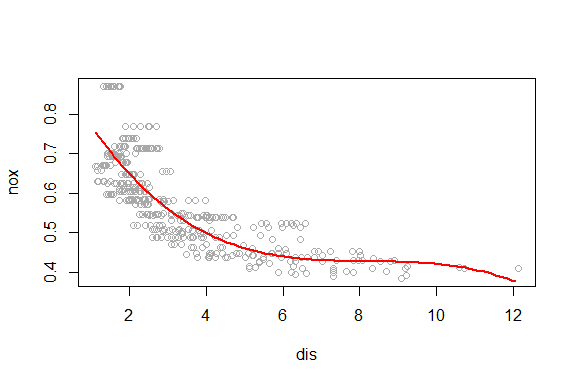
#### **9.** This question uses the variables dis(the weighted mean of distances to five Boston employment centers) and nox(nitrogen oxides concentration in parts per 10 million) from the Boston data. We will treat dis as the predictor and noxas the response.

1. 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.

library(MASS)  
set.seed(1)  
poly.fit <- lm(nox ~ poly(dis, 3), data = Boston)  
summary(poly.fit)

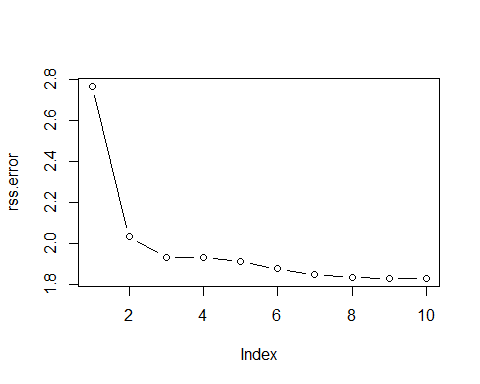
##   
## Call:  
## lm(formula = nox ~ poly(dis, 3), data = Boston)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.121130 -0.040619 -0.009738 0.023385 0.194904   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.554695 0.002759 201.021 < 2e-16 \*\*\*  
## 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

dislims = range(Boston$dis)  
dis.grid = seq(from = dislims[1], to = dislims[2], by = 0.1)  
preds <- predict(poly.fit, list(dis = dis.grid))  
plot(nox ~ dis, data = Boston, col = "darkgrey")  
lines(dis.grid, preds, col = "red", lwd = 2)



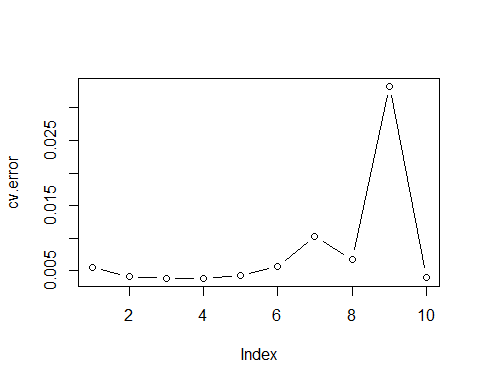
1. Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

rss.error <- rep(0,10)  
for (i in 1:10) {  
 lm.fit <- lm(nox~poly(dis,i), data=Boston)  
 rss.error[i] <- sum(lm.fit$residuals^2)  
}  
plot(rss.error, type="b")



1. Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

require(boot)  
set.seed(1)  
cv.error <- rep(NA,10)  
for (i in 1:10) {  
 glm.fit <- glm(nox~poly(dis,i), data=Boston)  
 cv.error[i] <- cv.glm(Boston, glm.fit, K=10)$delta[1]   
}  
plot(cv.error, type="b")



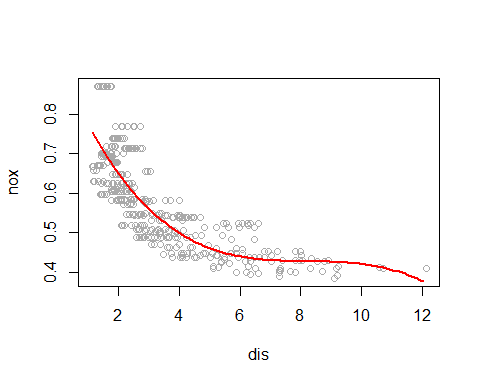
* 4th degree polynomial performs best, but 2 degree isn’t much worse

1. 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.

rs.fit <- lm(nox ~ bs(dis, knots = c(4, 7, 11)), data = Boston)  
summary(rs.fit)

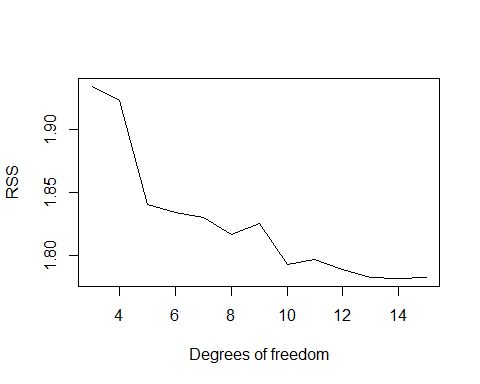
##   
## Call:  
## lm(formula = nox ~ bs(dis, knots = c(4, 7, 11)), data = Boston)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.124567 -0.040355 -0.008702 0.024740 0.192920   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.73926 0.01331 55.537 < 2e-16 \*\*\*  
## bs(dis, knots = c(4, 7, 11))1 -0.08861 0.02504 -3.539 0.00044 \*\*\*  
## bs(dis, knots = c(4, 7, 11))2 -0.31341 0.01680 -18.658 < 2e-16 \*\*\*  
## bs(dis, knots = c(4, 7, 11))3 -0.26618 0.03147 -8.459 3.00e-16 \*\*\*  
## bs(dis, knots = c(4, 7, 11))4 -0.39802 0.04647 -8.565 < 2e-16 \*\*\*  
## bs(dis, knots = c(4, 7, 11))5 -0.25681 0.09001 -2.853 0.00451 \*\*   
## bs(dis, 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

pred <- predict(rs.fit, list(dis = dis.grid))  
plot(nox ~ dis, data = Boston, col = "darkgrey")  
lines(dis.grid, preds, col = "red", lwd = 2)



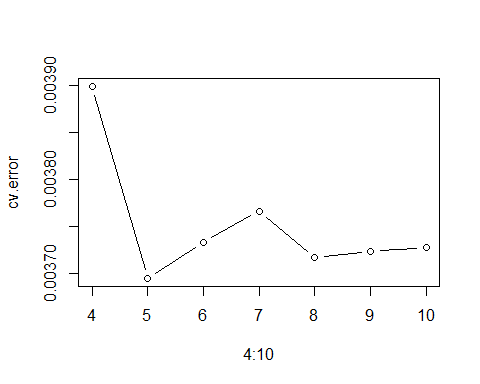
1. Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

rss <- rep(NA, 15)  
for (i in 3:15) {  
 range.fit <- lm(nox ~ bs(dis, df = i), data = Boston)  
 rss[i] <- sum(range.fit$residuals^2)  
}  
plot(3:15, rss[-c(1, 2)], xlab = "Degrees of freedom", ylab = "RSS", type = "l")



1. Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

cv.error <- rep(0,7)  
for (i in 4:10) {  
 glm.fit <- glm(nox~bs(dis, df=i), data=Boston)  
 cv.error[i-3] <- cv.glm(Boston, glm.fit, K=10)$delta[1]  
}  
plot(4:10, cv.error, type="b")



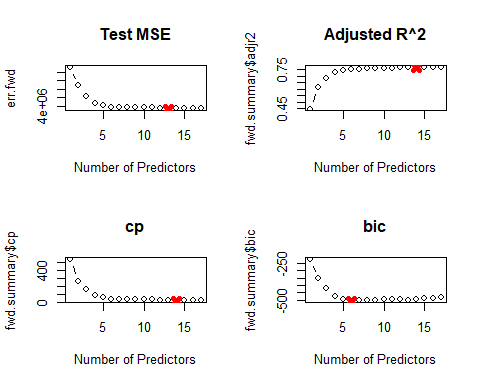
#### **10.** This question relates to the College data set.

1. Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.

require(ISLR)  
require(leaps)

## Loading required package: leaps

data(College)  
set.seed(1)  
  
trainid = sample(1:nrow(College), nrow(College)/2)  
train = College[trainid,]  
test = College[-trainid,]  
  
predict.regsubsets <- function(object, newdata, id, ...){  
 form <- as.formula(object$call[[2]])  
 mat <- model.matrix(form, newdata)  
 coefi <- coef(object, id=id)  
 xvars <- names(coefi)  
 mat[,xvars]%\*%coefi  
}  
  
fit.fwd = regsubsets(Outstate~., data=train, nvmax=ncol(College)-1, method ="forward")  
fwd.summary = summary(fit.fwd)  
  
err.fwd <- rep(NA, ncol(College)-1)  
for(i in 1:(ncol(College)-1)) {  
 pred.fwd <- predict(fit.fwd, test, id=i)  
 err.fwd[i] <- mean((test$Outstate - pred.fwd)^2)  
}  
  
par(mfrow=c(2,2))  
plot(err.fwd, type="b", main="Test MSE", xlab="Number of Predictors")  
min.mse <- which.min(err.fwd)   
points(min.mse, err.fwd[min.mse], col="red", pch=4, lwd=5)  
plot(fwd.summary$adjr2, type="b", main="Adjusted R^2", xlab="Number of Predictors")  
max.adjr2 <- which.max(fwd.summary$adjr2)   
points(max.adjr2, fwd.summary$adjr2[max.adjr2], col="red", pch=4, lwd=5)  
plot(fwd.summary$cp, type="b", main="cp", xlab="Number of Predictors")  
min.cp <- which.min(fwd.summary$cp)   
points(min.cp, fwd.summary$cp[min.cp], col="red", pch=4, lwd=5)  
plot(fwd.summary$bic, type="b", main="bic", xlab="Number of Predictors")  
min.bic <- which.min(fwd.summary$bic)   
points(min.bic, fwd.summary$bic[min.bic], col="red", pch=4, lwd=5)

 \* since scores aren’t improving that much after 6 predictors we will chose the 6 predictor model

err.fwd[6]

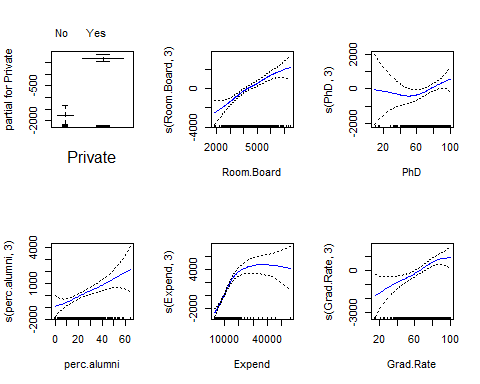
## [1] 3844857

1. Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.

library(gam)  
gam.fit = gam(Outstate ~ Private + s(Room.Board, 3) + s(PhD, 3) + s(perc.alumni, 3) + s(Expend, 3) + s(Grad.Rate, 3), data=train)

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

par(mfrow=c(2, 3))  
plot(gam.fit, se=T, col="blue")



1. Evaluate the model obtained on the test set, and explain the results obtained.

pred <- predict(gam.fit, test)  
(mse.error <- mean((test$Outstate - pred)^2))

## [1] 3353709

1. For which variables, if any, is there evidence of a non-linear relationship with the response?

summary(gam.fit)

##   
## Call: gam(formula = Outstate ~ Private + s(Room.Board, 3) + s(PhD,   
## 3) + s(perc.alumni, 3) + s(Expend, 3) + s(Grad.Rate, 3),   
## data = train)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -6963.2 -1131.7 -101.2 1322.2 7949.7   
##   
## (Dispersion Parameter for gaussian family taken to be 3821609)  
##   
## Null Deviance: 6989966760 on 387 degrees of freedom  
## Residual Deviance: 1417814885 on 370.9995 degrees of freedom  
## AIC: 7000.312   
##   
## Number of Local Scoring Iterations: 2   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## Private 1 1767246309 1767246309 462.435 < 2.2e-16 \*\*\*  
## s(Room.Board, 3) 1 1580386922 1580386922 413.540 < 2.2e-16 \*\*\*  
## s(PhD, 3) 1 351828206 351828206 92.063 < 2.2e-16 \*\*\*  
## s(perc.alumni, 3) 1 338018768 338018768 88.449 < 2.2e-16 \*\*\*  
## s(Expend, 3) 1 498727240 498727240 130.502 < 2.2e-16 \*\*\*  
## s(Grad.Rate, 3) 1 85973130 85973130 22.497 3.008e-06 \*\*\*  
## Residuals 371 1417814885 3821609   
## ---  
## 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, 3) 2 1.6491 0.1936   
## s(PhD, 3) 2 1.2597 0.2850   
## s(perc.alumni, 3) 2 0.2914 0.7474   
## s(Expend, 3) 2 30.9997 3.55e-13 \*\*\*  
## s(Grad.Rate, 3) 2 1.0910 0.3369   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

* only Expend is stat. sig. on the .05 level

#### **11.** In Section 7.7, it was mentioned that GAMs are generally fit using a backfitting approach. The idea behind backfitting is actually quite simple. We will now explore backfitting in the context of multiple linear regression. Suppose that we would like to perform multiple linear regression, but we do not have software to do so. Instead, we only have software to perform simple linear regression. Therefore, we take the following iterative approach: we repeatedly hold all but one coefficient estimate fixed at its current value, and update only that coefficientestimate using a simple linear regression. The process is continued until convergence that is, until the coefficient estimates stop changing. We now try this out on a toy example.

1. Generate a response Y and two predictors and , with n= 100.

set.seed(1)  
X1= rnorm(100)  
X2 = rnorm(100)  
eps = rnorm(100, sd=0.1)  
Y = 3 + 2.4\* X1 + 0.66 \*X2 + eps

1. Initialize to take on a value of your choice. It does not matter what value you choose.

beta1 = 10

1. Keeping fixed, fit the model

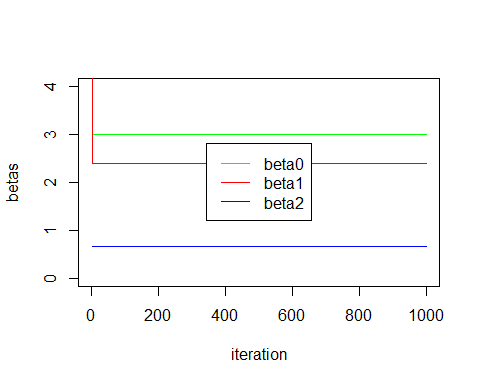
a = Y-beta1\*X1  
beta2 = lm(a~X2)$coef[2]

1. Keeping fixed, fit the model

a = Y-beta2\*X2  
beta1 = lm(a~X1)$coef[2]

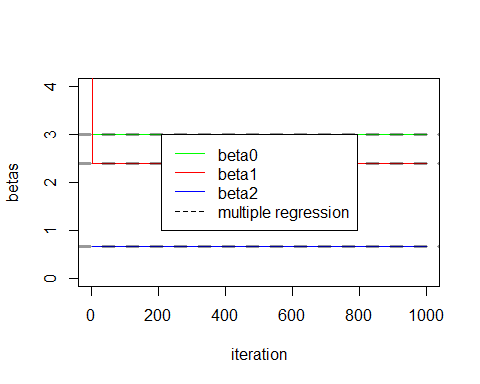
1. Write a for loop to repeat (c) and (d) 1,000 times. Report the estimates of , ,and at each iteration of the for loop. Create a plot in which each of these values is displayed, with , ,and each shown in a different color.

beta0 = rep(0, 1000)  
beta1 = rep(0, 1000)  
beta2 = rep(0, 1000)  
beta1 = 5  
for (i in 1:1000) {  
 a = Y - beta1[i] \* X1  
 beta2[i] = lm(a ~ X2)$coef[2]  
 a = Y - beta2[i] \* X2  
 lm.fit = lm(a ~ X1)  
 if (i < 1000) {  
 beta1[i + 1] = lm.fit$coef[2]  
 }  
 beta0[i] = lm.fit$coef[1]  
}  
  
plot(1:1000, beta0, type="l", xlab="iteration", ylab="betas", ylim=c(0, 4), col="green")  
lines(1:1000, beta1, col="red")  
lines(1:1000, beta2, col="blue")  
legend('center', c("beta0","beta1","beta2"), lty=1, col=c("green","red","blue"))



1. Compare your answer in (e) to the results of simply performing multiple linear regression to predict Y using and .Use the abline() function to overlay those multiple linear regression coefficient estimates on the plot obtained in (e)

lm.fit = lm(Y ~ X1 + X2)  
plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim=c(0, 4), col = "green")  
lines(1:1000, beta1, col = "red")  
lines(1:1000, beta2, col = "blue")  
abline(h = lm.fit$coef[1], lty = "dashed", lwd = 3, col = rgb(0, 0, 0, alpha = 0.4))  
abline(h = lm.fit$coef[2], lty = "dashed", lwd = 3, col = rgb(0, 0, 0, alpha = 0.4))  
abline(h = lm.fit$coef[3], lty = "dashed", lwd = 3, col = rgb(0, 0, 0, alpha = 0.4))  
legend("center", c("beta0", "beta1", "beta2", "multiple regression"), lty = c(1,   
 1, 1, 2), col = c("green", "red", "blue", "black"))



1. On this data set, how many backfitting iterations were requiredin order to obtain a “good” approximation to the multiple re-gression coefficient estimates?

beta0[1:3]

## [1] 3.002627 3.002535 3.002535

beta1[1:3]

## [1] 5.000000 2.402114 2.402111

beta2[1:3]

## [1] 0.6570755 0.6546533 0.6546533

* in this case one iteration is enough