ANLY512_HW7

Hongyang Zheng

2019/4/1

```r
library(ISLR)
library(leaps)
library(gam)
```

```
## Loading required package: splines
## Loading required package: foreach
## Loaded gam 1.16
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 2.0-16
```

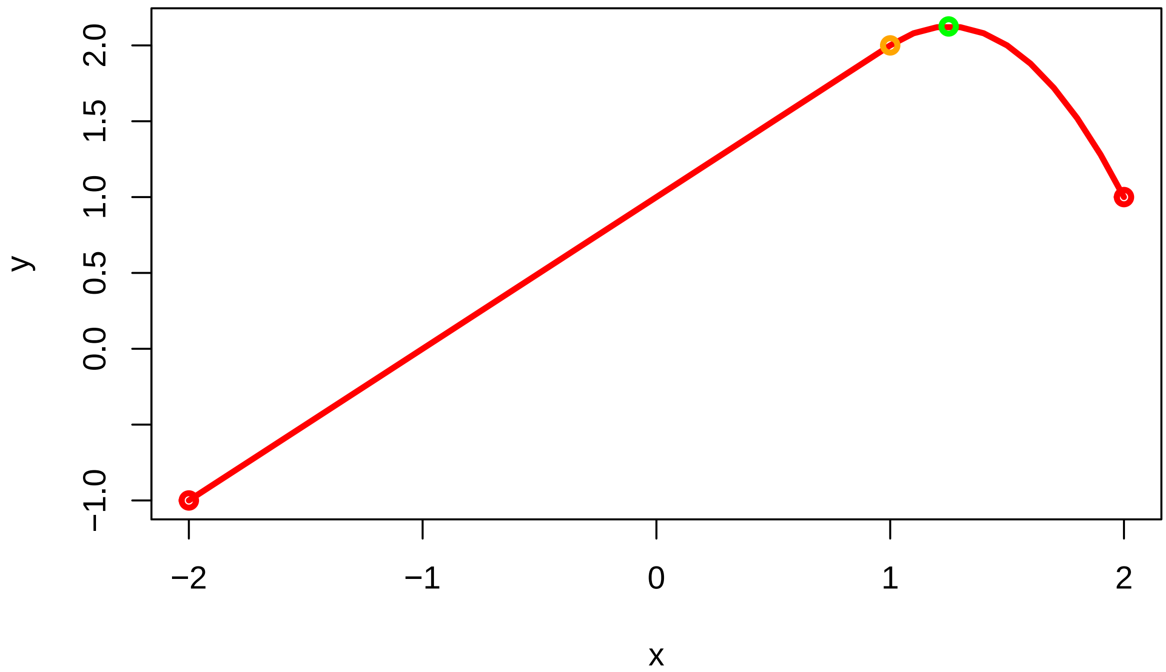
```r
require("splines")
```

Problem 3

For $X \in [-2, 1), \hat{Y} = 1 + X$. The slope is 1 and the endpoints are $(-2, -1)$ and $(1, 2)$.

For $X \in [1, 2], \hat{Y} = 1 + X - 2(X - 1)^2 = 1 + X - 2X^2 + 4X - 2 = -1 + 5X - 2X^2$. This is a Quadratic parabola. This parabola intersects the above linear line at $(1, 2)$. The endpoints are $(1, 2)$ and $(2, 1)$. This parabola peaks at the point $(1.25, 2.125)$

```r
# Make the plot
x = seq(-2, 2, 0.1)
y = 1 + x + -2 * (x-1)^2 * I(x>1)
plot(x, y, type='l', col='red', lwd=3)

# Add points
points(-2, -1, col='red', lwd=3)         #--endpoint
points(2, 1, col='red', lwd=3)           #--endpoint
points(1, 2, col='orange', lwd=3)        #--intersection
points(1.25, 2.125, col='green', lwd=3)  #--peak
```



Problem 5

a)

$\hat{g}_2$ will have the smaller training RSS because it will be a higher order polynomial due to the order of the derivative penalty function is higher.

b)

$\hat{g}_1$ will have the smaller test RSS, because $\hat{g}_2$ may overfit the data with the extra degree of freedom

c)

When $\lambda = 0, \hat{g}_1 = \hat{g}_2$, since the penalty has no power anymore. Therefore, they have the same training and test RSS.

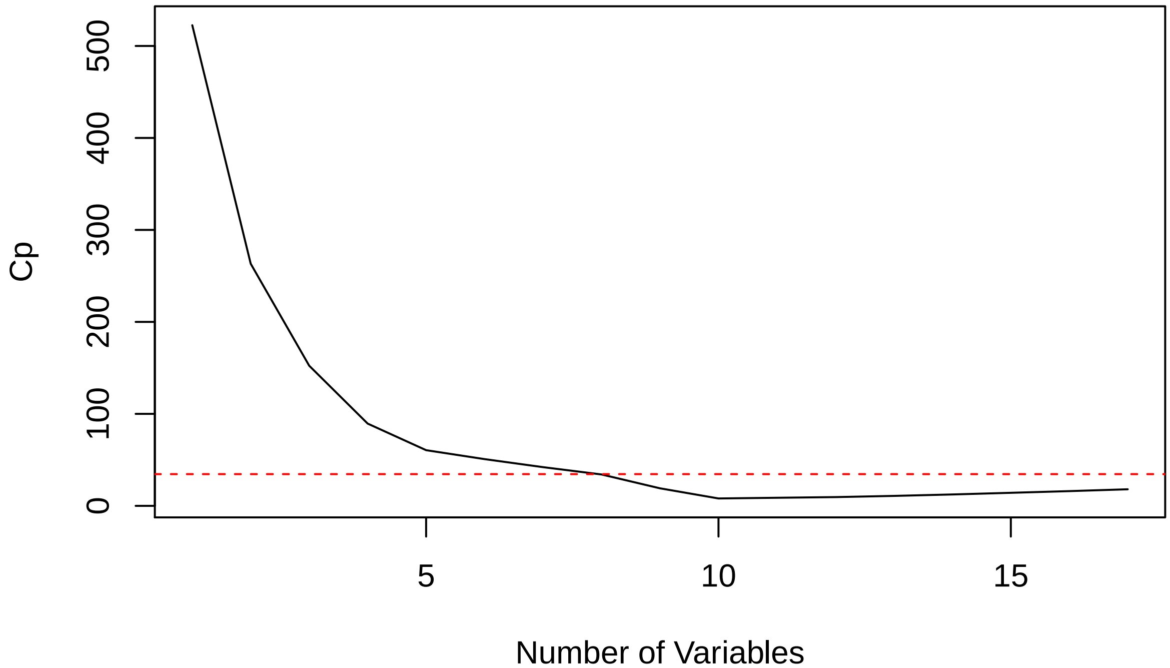
Problem 10

a)

```
set.seed(1234)
# Split the data into training(50%) and test(50%)
length = dim(College)[1]
train = sample(length, 0.5*length)
test = -train
College.train = College[train, ]
College.test = College[test, ]

# Forward stepwise selection
```

```r
model.fwd = regsubsets(Outstate ~ ., data = College.train, nvmax = 17, method = "forward")
fwd.summary = summary(model.fwd)
```

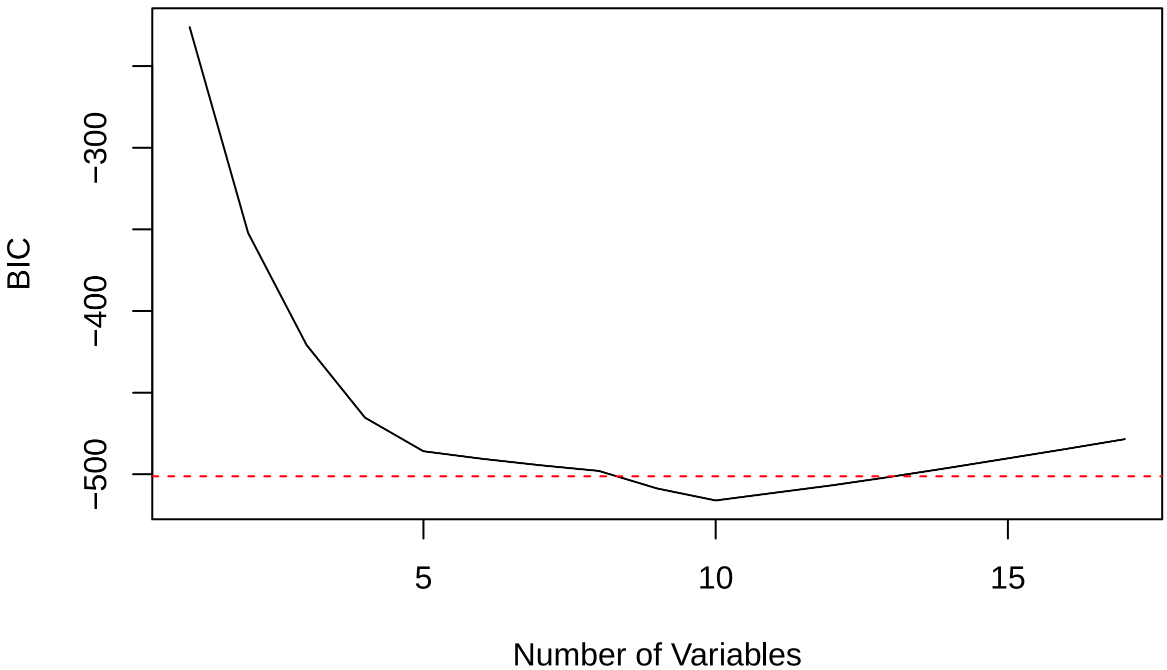
```r
# Plot Cp, BIC, Adjr2 to select predictors
plot(fwd.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
min.cp = min(fwd.summary$cp)
std.cp = sd(fwd.summary$cp)
abline(h = min.cp + 0.2 * std.cp, col = "red", lty = 2)
abline(h = min.cp - 0.2 * std.cp, col = "red", lty = 2)
```



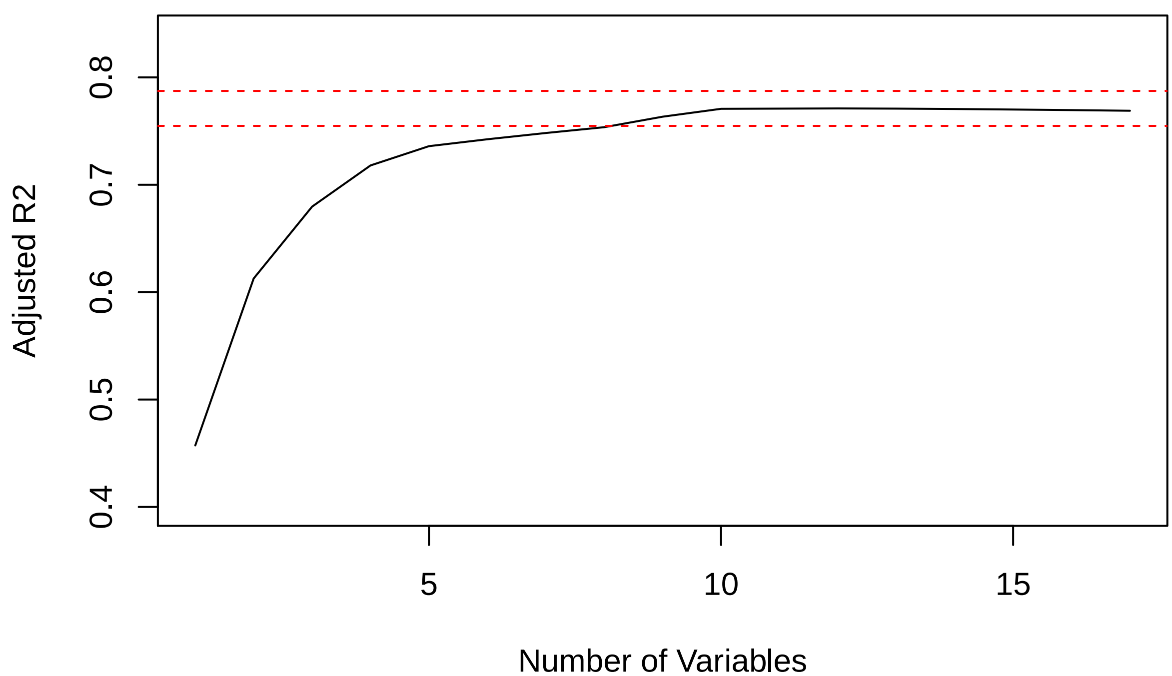

```r
which.min(fwd.summary$cp)
```

```
## [1] 10
```

```r
plot(fwd.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
min.bic = min(fwd.summary$bic)
std.bic = sd(fwd.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)
```




```
which.min(fwd.summary$bic)

## [1] 10

plot(fwd.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R2", type = "l", ylim = c(0.4, 0
max.adjr2 = max(fwd.summary$adjr2)
std.adjr2 = sd(fwd.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)
```




```
which.max(fwd.summary$adjr2)
```

```
## [1] 12
```

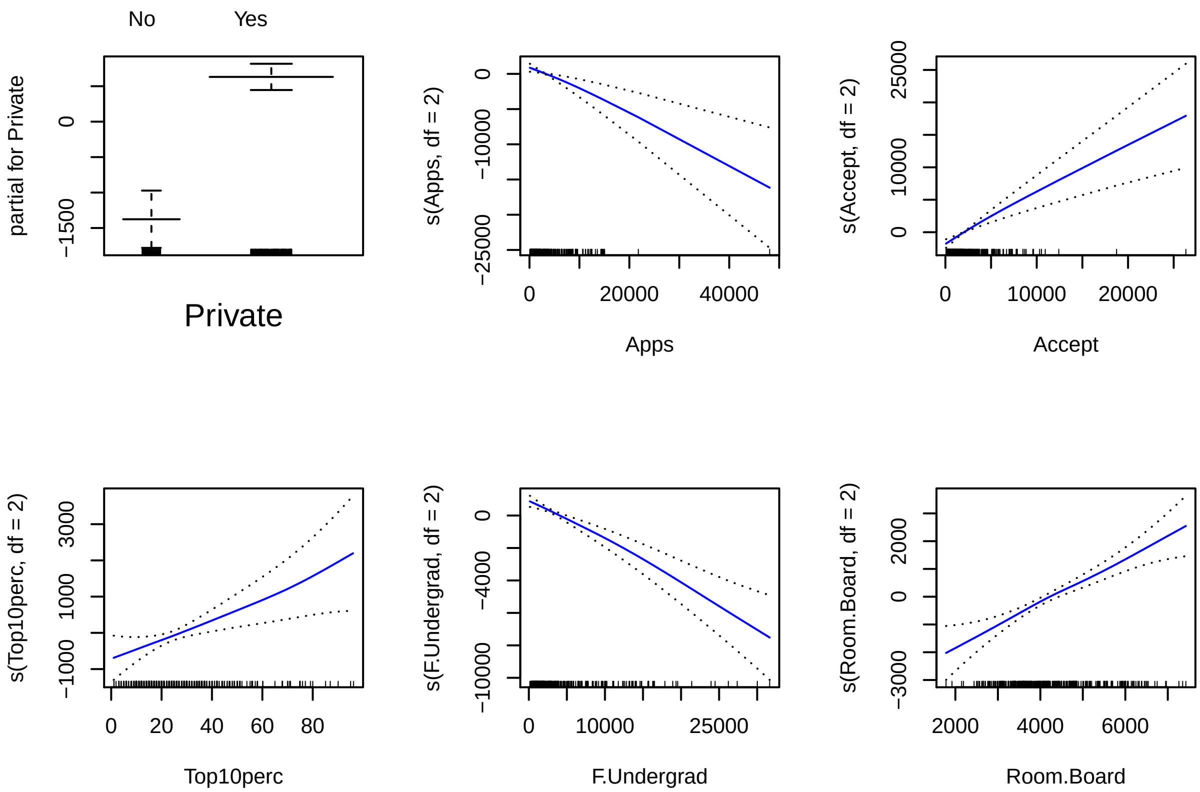
cp, BIC and adjr2 scores show that size 10 is the optimal size for the subset.

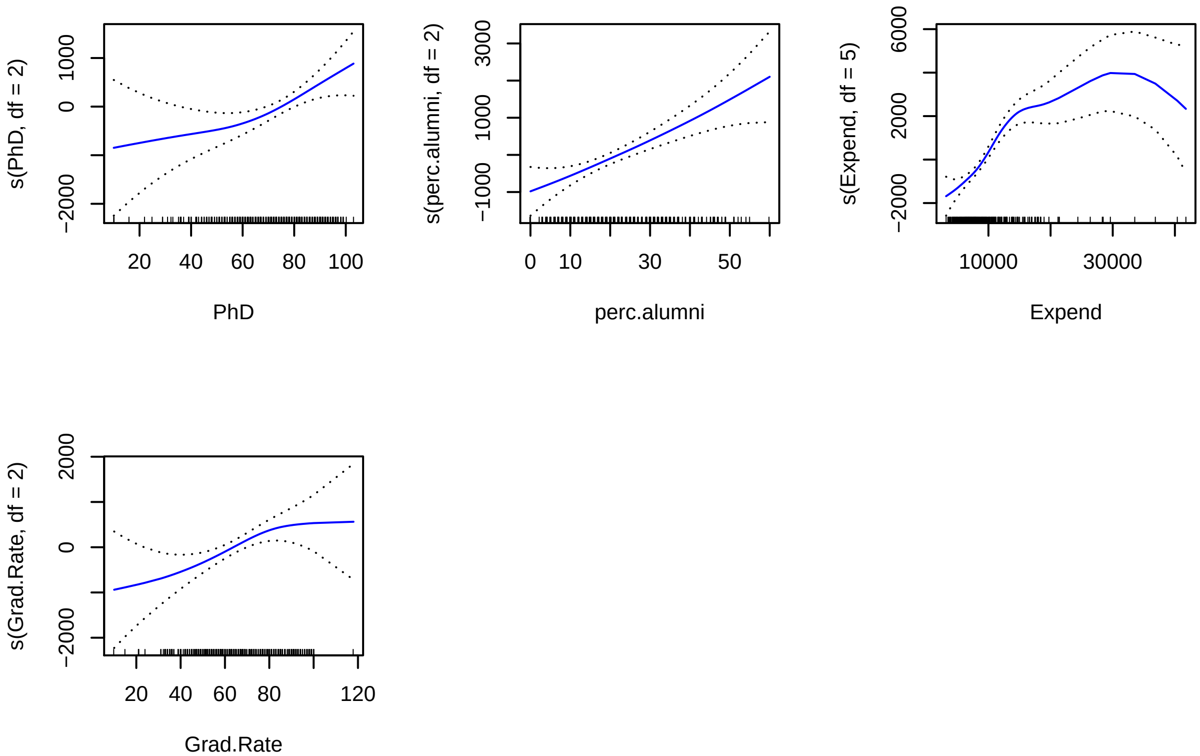
```r
coefi = coef(model.fwd, id = 10)
names(coefi)
```

```
##  [1] "(Intercept)" "PrivateYes"   "Apps"          "Accept"       "Top10perc"
##  [6] "F.Undergrad" "Room.Board"   "PhD"           "perc.alumni" "Expend"
## [11] "Grad.Rate"
```

b)

```r
gam.fit = gam(Outstate ~ Private + s(Apps, df = 2) + s(Accept, df = 2) +
    s(Top10perc, df = 2) + s(F.Undergrad, df = 2) + 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)

```r
# Calculate test error
gam.pred = predict(gam.fit, College.test)
gam.err = mean((College.test$Outstate - gam.pred)^2)
gam.err
```

```
## [1] 3670375
```

```r
# Calculate R2 for the test data
gam.tss = mean((College.test$Outstate - mean(College.test$Outstate))^2)
test.r2 = 1 - gam.err/gam.tss
test.r2
```

```
## [1] 0.7728951
```

```r
# Build a linear model
fit.linear=lm(Outstate ~ Private + Apps + Accept + Top10perc + F.Undergrad + Room.Board +
    PhD + perc.alumni + Expend + Grad.Rate, data = College.test)
summary(fit.linear)
```

```
##
## Call:
## lm(formula = Outstate ~ Private + Apps + Accept + Top10perc +
##     F.Undergrad + Room.Board + PhD + perc.alumni + Expend + Grad.Rate,
##     data = College.test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6949.2 -1239.9    24.2  1212.0 10451.6
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.270e+03  6.889e+02  -4.746 2.94e-06 ***
## PrivateYes   2.946e+03  3.950e+02   7.459 6.04e-13 ***
## Apps        -1.206e-01  9.479e-02  -1.272 0.204228
## Accept       5.830e-01  1.626e-01   3.585 0.000381 ***
## Top10perc    7.824e+00  9.216e+00   0.849 0.396444
## F.Undergrad -1.370e-01  6.365e-02  -2.153 0.031977 *
## Room.Board   9.217e-01  1.150e-01   8.017 1.36e-14 ***
## PhD          2.764e+01  8.746e+00   3.160 0.001703 **
## perc.alumni  4.770e+01  1.033e+01   4.616 5.37e-06 ***
## Expend       2.085e-01  2.658e-02   7.845 4.49e-14 ***
## Grad.Rate    2.869e+01  8.181e+00   3.507 0.000508 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2001 on 378 degrees of freedom
## Multiple R-squared:  0.7593, Adjusted R-squared:  0.7529
## F-statistic: 119.2 on 10 and 378 DF,  p-value: < 2.2e-16
```

We obtain a test R-squared of 0.77 using GAM with 10 predictors. This result is slightly higher than the result 0.75 we get from an OLS model using these 10 predictors.

d)

```r
summary(gam.fit)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Apps, df = 2) + s(Accept,
##     df = 2) + s(Top10perc, df = 2) + s(F.Undergrad, df = 2) +
##     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
## -6058.15 -1070.81    22.73  1201.47  4324.58
##
## (Dispersion Parameter for gaussian family taken to be 3296377)
##
##     Null Deviance: 6200724881 on 387 degrees of freedom
## Residual Deviance: 1203177827 on 365.0001 degrees of freedom
## AIC: 6948.62
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                         Df     Sum Sq    Mean Sq  F value    Pr(>F)
## Private                  1 2026635720 2026635720 614.8071 < 2.2e-16 ***
## s(Apps, df = 2)          1  362208581  362208581 109.8808 < 2.2e-16 ***
## s(Accept, df = 2)        1   87903227   87903227  26.6666 3.983e-07 ***
## s(Top10perc, df = 2)     1  889868580  889868580 269.9535 < 2.2e-16 ***
## s(F.Undergrad, df = 2)   1  263695085  263695085  79.9954 < 2.2e-16 ***
## s(Room.Board, df = 2)    1  405193075  405193075 122.9207 < 2.2e-16 ***
## s(PhD, df = 2)           1   75343091   75343091  22.8563 2.536e-06 ***
## s(perc.alumni, df = 2)   1  126073380  126073380  38.2460 1.672e-09 ***
## s(Expend, df = 5)        1  169121440  169121440  51.3053 4.394e-12 ***
```

```
## s(Grad.Rate, df = 2)     1   29110772   29110772   8.8311  0.003158 **
## Residuals              365 1203177827    3296377
## ---
## 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(Apps, df = 2)                1 3.0066   0.08377 .
## s(Accept, df = 2)              1 3.7382   0.05395 .
## s(Top10perc, df = 2)           1 0.6023   0.43823
## s(F.Undergrad, df = 2)         1 1.5411   0.21525
## s(Room.Board, df = 2)          1 0.8856   0.34729
## s(PhD, df = 2)                 1 2.4637   0.11737
## s(perc.alumni, df = 2)         1 0.3581   0.54992
## s(Expend, df = 5)              4 8.8498 7.898e-07 ***
## s(Grad.Rate, df = 2)           1 2.9876   0.08475 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Anova for Nonparametric Effects test shows a strong evidence for non-linear relationship between response and Expend.

Problem 11

a)

```r
set.seed(1234)

# Generate dataset
X1 = rnorm(100)
X2 = rnorm(100)
eps = rnorm(100, sd = 0.1)
Y = -3.2 + 2.8 * X1 - 0.53 * X2 + eps
```

b)

```r
# Initialized beta1
beta1=-10
```

c)

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

```
##         X2
## -0.8375226
```

d)

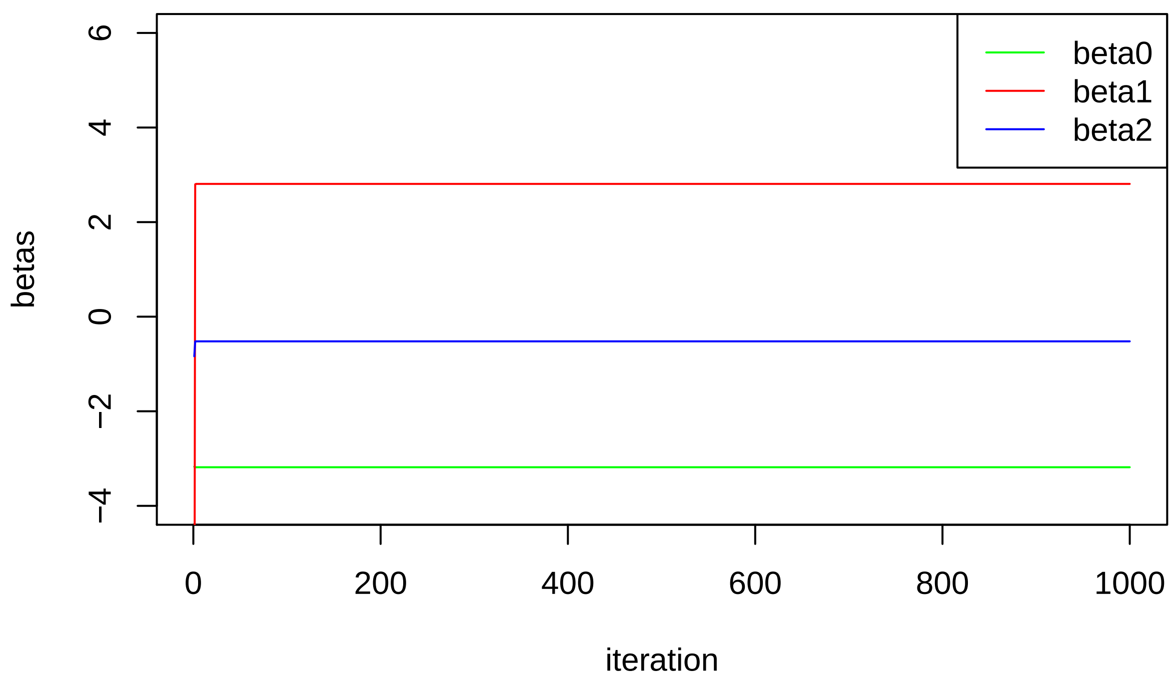
```r
b=Y-beta2*X2
beta1=lm(a~X1)$coef[2]
beta1
```

```
##       X1
## 12.82165
```

e)

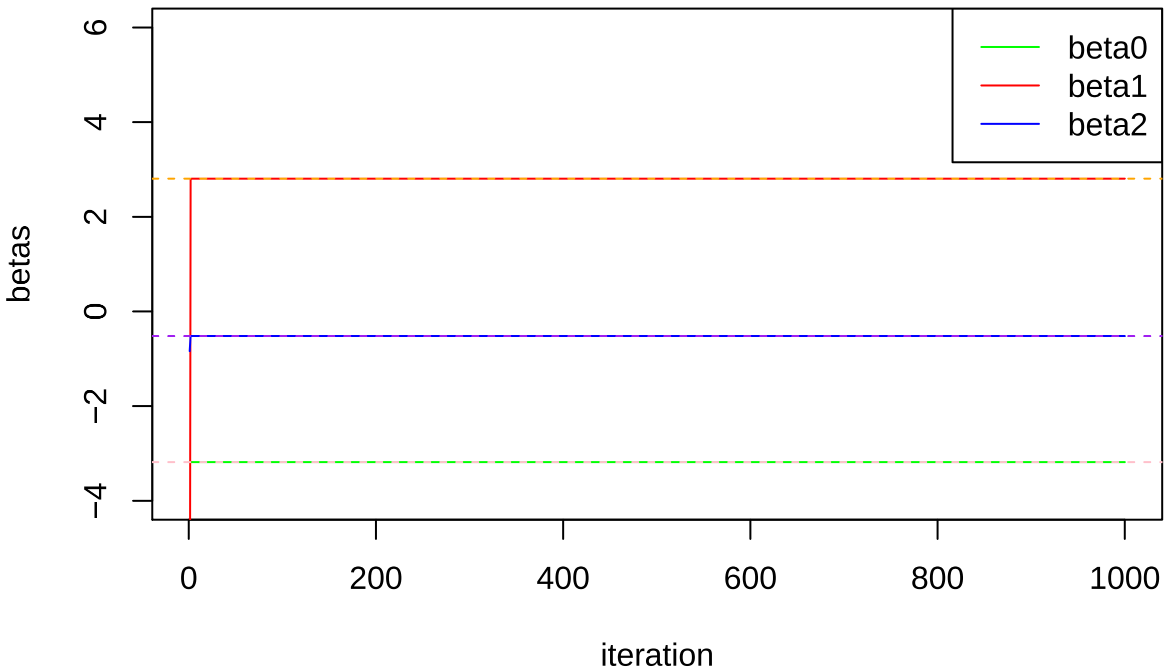
```r
# Initialize all betas and let beta1[1] = -10
beta0=rep(NA, 1000)
beta1=rep(NA, 1000)
beta2=rep(NA, 1000)
beta1[1]= -10

# For loop to repeat c and d 1000 times
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]
}
```

```r
# Make plots
plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim = c(-4, 6), col = "green")
lines(1:1000, beta1, col = "red")
lines(1:1000, beta2, col = "blue")
legend("topright", c("beta0", "beta1", "beta2"), lty = 1, col = c("green", "red", "blue"))
```



f)

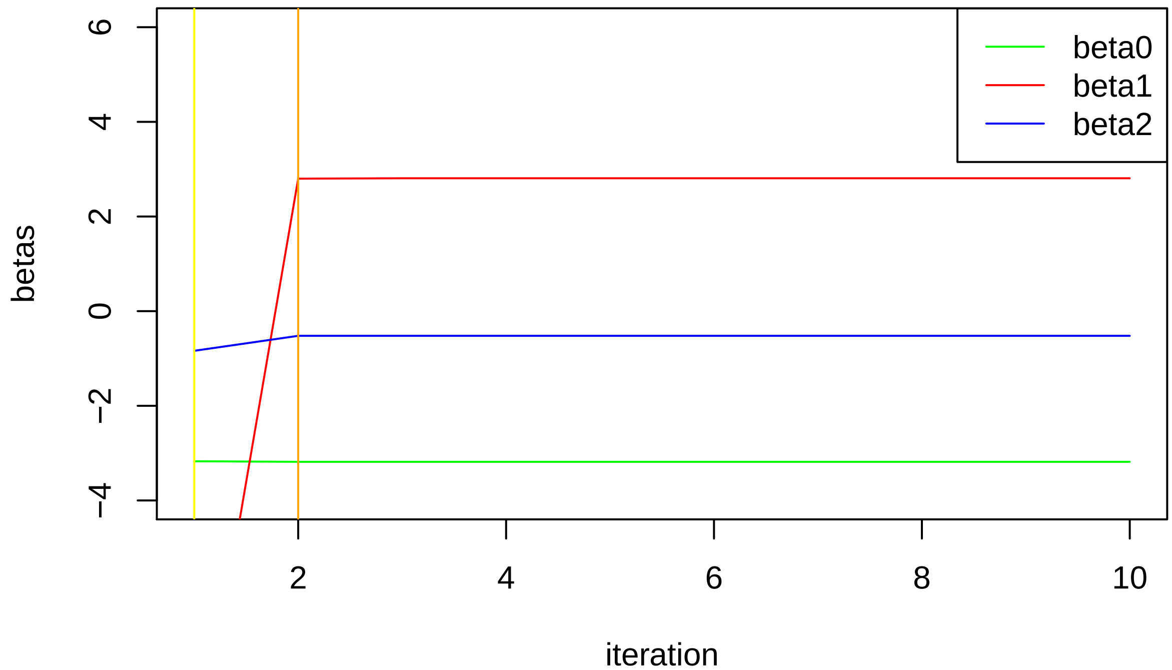
```r
# Fit a linear model
fit.model.2=lm(Y~X1+X2)
beta0.fit=fit.model.2$coef[1]
beta1.fit=fit.model.2$coef[2]
beta2.fit=fit.model.2$coef[3]

# Make a plott
plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas", ylim = c(-4, 6), col = "green")
lines(1:1000, beta1, col = "red")
lines(1:1000, beta2, col = "blue")
abline(h = beta0.fit, lty = "dashed", lwd = 1, col ='pink')
abline(h = beta1.fit, lty = "dashed", lwd = 1, col ='orange')
abline(h = beta2.fit, lty = "dashed", lwd = 1, col ='purple')
legend("topright", c("beta0", "beta1", "beta2"), lty = 1, col = c("green", "red", "blue"))
```



The dashed lines overlap the three horizontal lines from previous part, which indicates that the estimated multiple regression coefficients match exactly with the coefficients obtained using backfitting.

g)

```r
plot(1:10, beta0[1:10], type = "l", xlab = "iteration", ylab = "betas", ylim = c(-4, 6), col = "green")
lines(1:10, beta1[1:10], col = "red")
lines(1:10, beta2[1:10], col = "blue")
legend("topright", c("beta0", "beta1", "beta2"), lty = 1, col = c("green", "red", "blue"))
abline(v = 2, col = 'orange')
abline(v = 1, col = 'yellow')
```



From above graph we can see that at the first iteration, the coefficients start to converge. And at the second iteration, it has converged to the estimated multiple regression coefficients and does not change. Therefore, maybe 1 or 2 backfitting iterations is enough.

Problem Xtra #54

a)

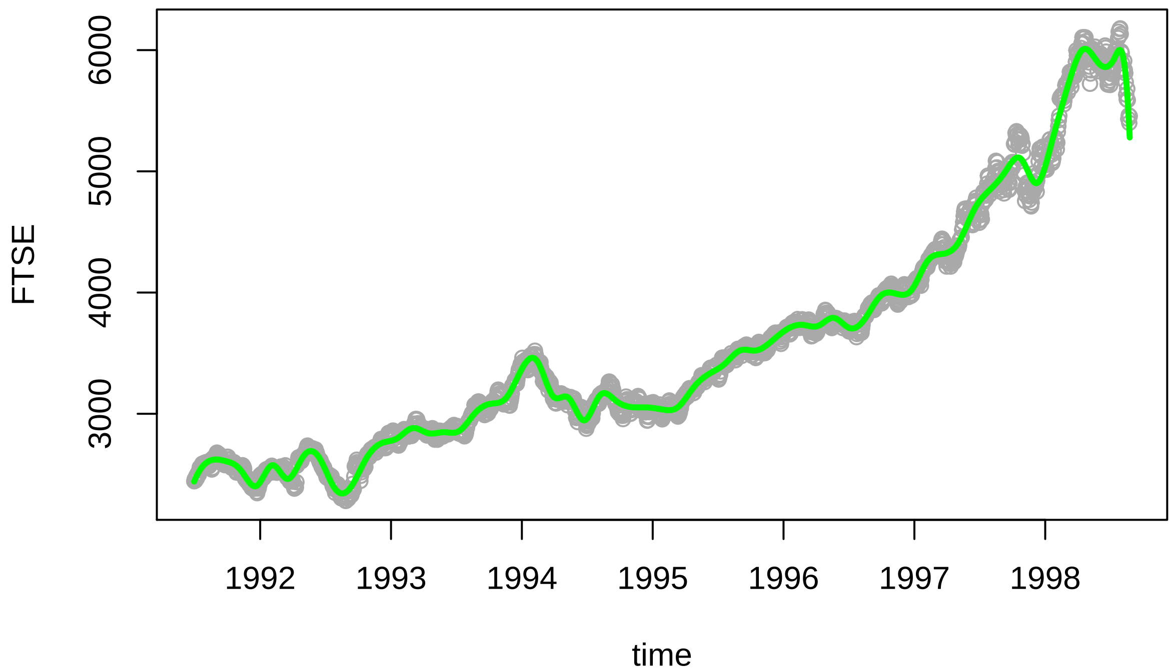
```
set.seed(1234)

# Extract the data
summary(EuStockMarkets)
```

```
##       DAX            SMI            CAC            FTSE     
##  Min.   :1402   Min.   :1587   Min.   :1611   Min.   :2281  
##  1st Qu.:1744   1st Qu.:2166   1st Qu.:1875   1st Qu.:2843  
##  Median :2141   Median :2796   Median :1992   Median :3247  
##  Mean   :2531   Mean   :3376   Mean   :2228   Mean   :3566  
##  3rd Qu.:2722   3rd Qu.:3812   3rd Qu.:2274   3rd Qu.:3994  
##  Max.   :6186   Max.   :8412   Max.   :4388   Max.   :6179
```

```
EuStockMarkets.1=as.data.frame(EuStockMarkets)
FTSE=EuStockMarkets.1$FTSE
n=dim(EuStockMarkets.1)[1]

# Set time
time=seq(1, n, 1)
time.1=as.numeric(time(EuStockMarkets))
```

```
# Build a base spline model with 60 evenly spaced knots
myknots = seq(1, 1860, by = 31)
base.spline = lm(FTSE ~ bs(time, knots = myknots))
```

```r
# Make a plot
plot(time.1, FTSE, col='darkgray', xlab='time')
lines(time.1, base.spline$fitted.values, col='green', lwd=3)
```



The base spline performs good on prediting the FTSE overall, especially the middle part. But it does not capture the peak at 1998 and after 1998, and also does not capture some fluctations during 1993-1994 and 1997-1998. There are no obvious oscillations (at the beginning, there are some fluctations, but it is normal because it reflects the data).

b)

```r
# Build lasso and find lambda 1se
matrix.1 = model.matrix(base.spline)
mod.lasso = cv.glmnet(matrix.1, FTSE, alpha=1)
lambda.1se = mod.lasso$lambda.1se
lambda.1se
```

```
## [1] 2.233976
```

```r
# Using lambda 1se to build new model and predict
fit.lasso.1se = glmnet(matrix.1, FTSE, alpha = 1, lambda = lambda.1se)
predict.lasso.1se = predict(fit.lasso.1se, newx = matrix.1)

# Find the df for the old and new model
fit.lasso.1se$df
```

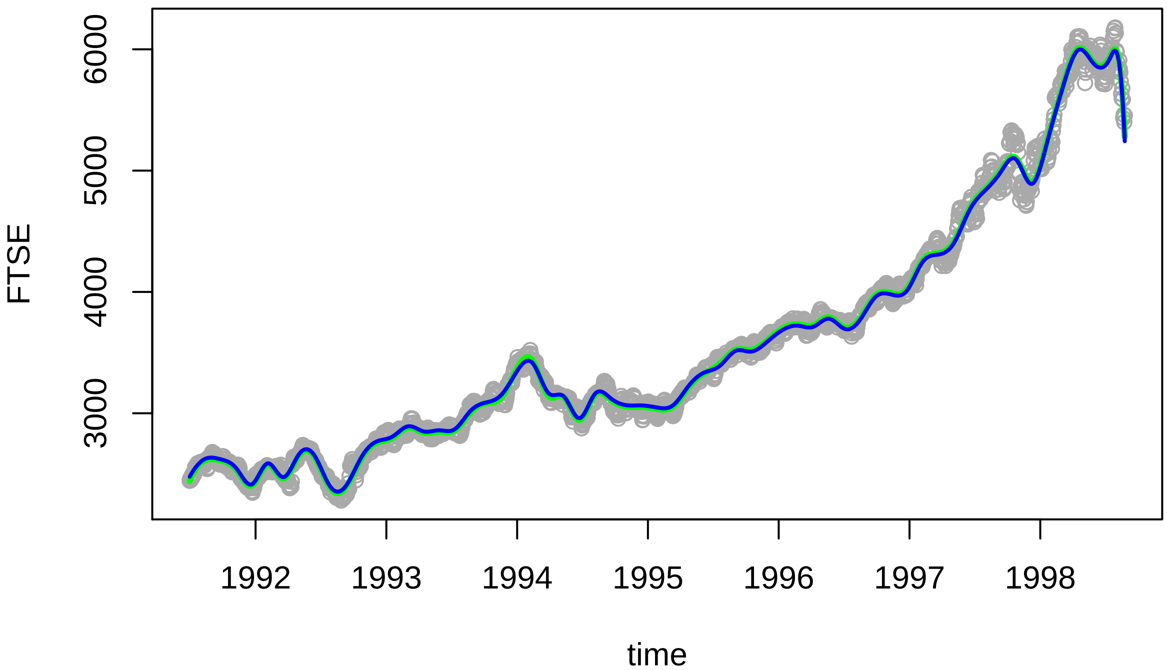
```
## [1] 62
```

```r
length(base.spline$coefficients)
```

```
## [1] 64
```

```r
# Make a plot
plot(time.1, FTSE, col='darkgray', xlab='time')
```

```
lines(time.1, base.spline$fitted.values, col='green', lwd=3)
lines(time.1, predict.lasso.1se, col='blue', lwd=2)
```



Using lambda 1se, the df for this new model is 62, which is less than the previous model's df 64. So some of spline model should not be used. Although we use less basis function, the predicted results are similar since the graph in blue is looks like the graph in green(part a)).

Problem Xtra #56

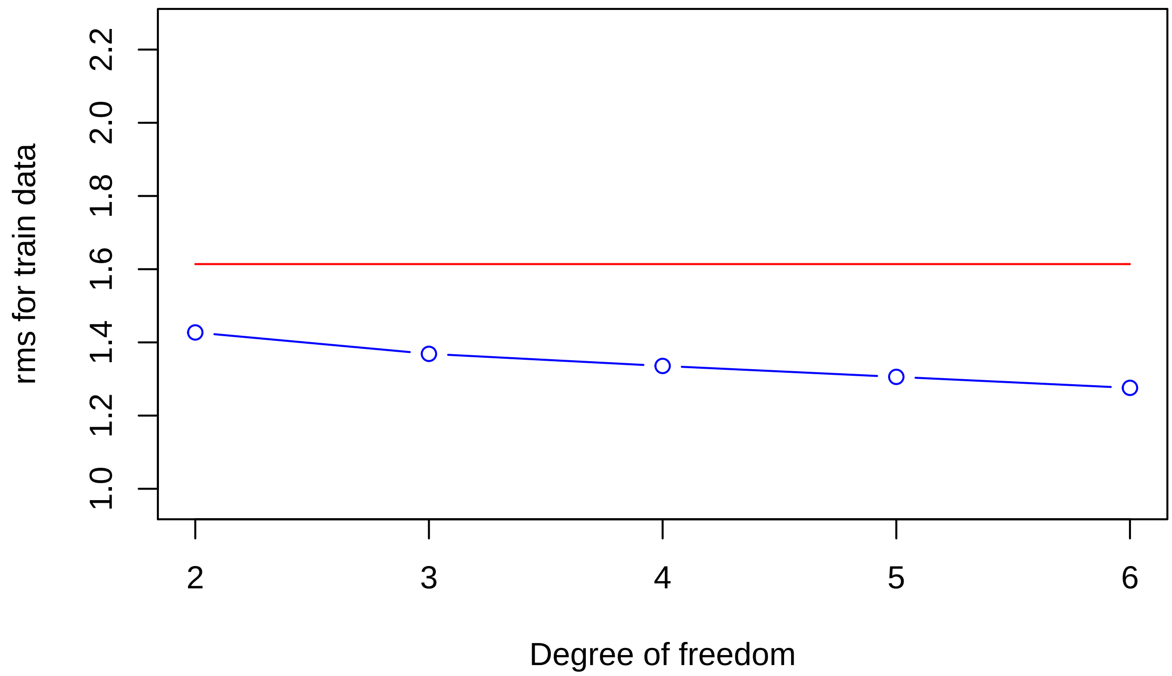
```
# Load data
AD=read.csv('~/Desktop/other/data/Advertising.csv')

# Split the data into train 70% and test 30%
n=dim(AD)[1]
train=sample(n, 0.7*n, replace = FALSE)
test=-train
AD.train=AD[train,]
AD.test=AD[test,]
```

a)

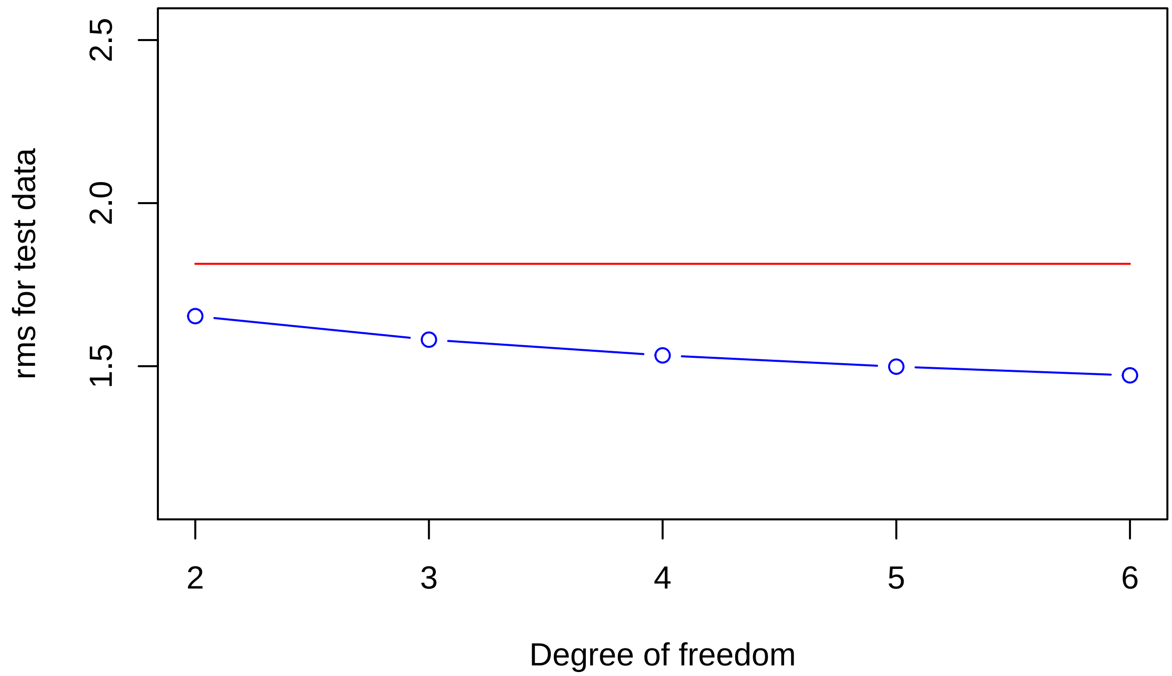
```
# Build a multiple regression model
lm.model=lm(Sales~TV+Radio+Newspaper, data=AD.train)

# Fit generalized additive models to predict sales
# Using smoothing splines of degrees 2, 3, 4, 5, 6 for the three predictors
train.rms=rep(NA, 5)
test.rms=rep(NA, 5)
for (i in (2:6))
{
```

```r
gam.fit=gam(Sales~s(TV,i)+s(Radio,i)+s(Newspaper,i), data=AD.train)
pred.train=gam.fit$fitted.values
pred.test=predict(gam.fit, newdata = AD.test)
train.rms[i-1] = sqrt(mean((AD.train$Sales-pred.train)^2))
test.rms[i-1] = sqrt(mean((AD.test$Sales-pred.test)^2))
}

# Calculate the train and test rms for the multiple linear model
pre.model.train=predict(lm.model)
pre.model.test=predict(lm.model, newdata = AD.test)
rms1 = sqrt(mean((AD.train$Sales-pre.model.train)^2))
rms2 = sqrt(mean((AD.test$Sales-pre.model.test)^2))
```

```r
# Make a plot for train data
plot(2:6, rep(rms1,5), type='l', col='red', xlab='Degree of freedom', ylab = 'rms for train data')
lines(2:6, train.rms, type = 'b', col='blue')
```




```r
# Make a plot for test data
plot(2:6, rep(rms2,5), type='l', col='red', xlab='Degree of freedom', ylab = 'rms for test data')
lines(2:6, test.rms, type = 'b', col='blue')
```



From the graphs we can see that the rms prediction errors are all smaller than the rms prediction error of a multiple regression model both on the training set and on the test data.

b)

Since with increasing of degree of freedom, the test rms continues decreasing, there is no overfitting.

c)

Since there is no overfitting, we should choose the model with the smallest rms error. Thus, we should use the model with df=6.