Homework-6

tg289@njit.edu

1. Exercises 1, 3, and 9 from Section 6.8 of our textbook.

Q1.

A. Best Subset selection has the smallest training RSS. The other 2 methods determine models with a path dependency on which predictors they pick first as they iterate to the Kth model.

B. It is unclear which selection algorithm will produce the model with the lowest RSS. The model produced by best subset selection certainly has higher chances of producing the model with the lowest training RSS since all subsets are considered. However, it may also overfit the model causing the other approaches to have a better test RSS. Thus, with the information given it is unclear which model will perform best in terms of the test RSS.

C.

i. **True**. Forward step selection does not drop the previously chosen predictors but instead continues in the same search path.

ii. **True**. One predictor at a time will be dropped by the backward step selection(with k+1 predictors) and it will select the k-predictor subset that will provide the lowest training RSS.

iii. **False**. The statement is not always true, the forward step selection algorithm predicts the lowest RSS for the 1-variable model. However, a combination of other predictors may have a better RSS score than the model with only 1-variable.

iv. **False**. The statement is similar to the one above, the algorithms may not consider same subsets of predictors.

v. **False**. Best subset selection may drop previously chosen predictors when a new one is added, since all possible subsets are considered.

Q3.

A. **(iv) steadily Decreases**, There will be a monotonically decrease in RSS as a bigger set of solutions becomes feasible and an increase in variance occurs.

B. **(i) Decrease Initially**, An increase in s means there will be an increase in flexibility in the model.

C. **(iii) Steadily Increase**, When s=0, the model effectively predicts a constant and has almost no variance. As we increase s, the models include more β's and their values start increasing. At this point, the values of β's become highly dependent on training data, thus increasing the variance.

D. **(iv) Steadily Decrease**, When s=0, the model effectively predicts a constant and hence the prediction is far from actual value. Thus bias is high. As s increases, more β s become non-zero and thus the model continues to fit training data better. And thus, bias decreases.

E. **(v) Remains constant**, By definition, irreducible error is model independent and hence irrespective of the choice of s, remains constant.

Q9.

A.

Code-

```
library(ISLR)
data(College)
set.seed(11)
train = sample(1:dim(College)[1], dim(College)[1] / 2)
test <- -train
College.train <- College[train, ]
College.test <- College[test, ]
```

B.

Code-

```
fit.lm <- lm(Apps ~ ., data = College.train)
pred.lm <- predict(fit.lm, College.test)
mean((pred.lm - College.test$Apps)^2)
```

Output-

```
[1] 1026096
```

The test error obtained with a **linear model using least squares is 1026096**.

C.

Code-

```
train.mat <- model.matrix(Apps ~ ., data = College.train)
test.mat <- model.matrix(Apps ~ ., data = College.test)
grid <- 10 ^ seq(4, -2, length = 100)
fit.ridge <- glmnet(train.mat, College.train$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
cv.ridge <- cv.glmnet(train.mat, College.train$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
bestlam.ridge <- cv.ridge$lambda.min
bestlam.ridge
```

Output-

```
 0.01
```

Best lambda is 0.01

Code-

```
pred.ridge <- predict(fit.ridge, s = bestlam.ridge, newx = test.mat)
mean((pred.ridge - College.test$Apps)^2)
```

Output-

```
[1] 1026069
```

The test error here is **lower than the linear model using least squares**.

D.

Code-

```
fit.lasso <- glmnet(train.mat, College.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
cv.lasso <- cv.glmnet(train.mat, College.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
bestlam.lasso <- cv.lasso$lambda.min
bestlam.lasso
```

Output-

```
[1] 0.01
```

The best **lambda value for lasso is 0.01**

Code-

```
pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
mean((pred.lasso - College.test$Apps)^2)
```

Output-

```
[1] 1026036
```

The test error is still **lower than the linear model and even ridge model** . LASSO model has **13 non-zero coefficients**

E.

Code-

```
library(pls)
fit.pcr <- pcr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(fit.pcr, val.type = "MSEP")
pred.pcr <- predict(fit.pcr, College.test, ncomp = 10)
mean((pred.pcr - College.test$Apps)^2)
```

Output-

```
[1] 1867486
```

Now with pcr model we get a test error higher than linear model, ridge and lasso model.

For getting the value of M, we will use the **summary() function in R-**

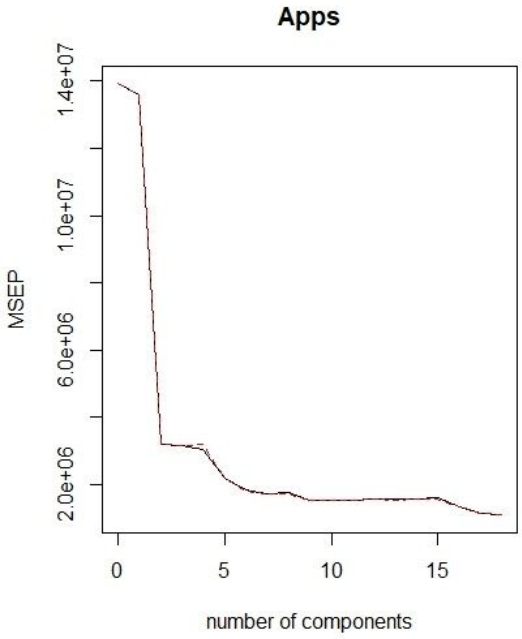
Code-

```
summary(fit.pcr)
```

Output-

```
Data:    X dimension: 388 17
         Y dimension: 388 1
Fit method: svdpc
Number of components considered: 17
VALIDATION: RMSEP
Cross-validated using 10 random segments.
      (Intercept)  1 comps  2 comps  3 comps  4 comps
CV           4306     4319     2385     2341     1966
adjCV        4306     4321     2382     2341     1928
      5 comps  6 comps  7 comps  8 comps  9 comps  10 comps
CV       1933     1911     1912     1913     1819      1805
adjCV    1918     1903     1908     1911     1810      1794
      11 comps  12 comps  13 comps  14 comps  15 comps
CV        1801      1802      1791      1802      1761
adjCV     1793      1794      1784      1797      1746
      16 comps  17 comps
CV        1368      1283
adjCV     1354      1272
TRAINING: % variance explained
     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X     32.5151    58.15    64.55    70.40    75.79    80.74
```

Apps 0.4738 70.38 71.98 81.87 82.28 82.41
7 comps 8 comps 9 comps 10 comps 11 comps 12 comps
X 84.23 87.51 90.49 92.84 95.02 96.79
Apps 82.44 82.71 84.57 84.98 85.13 85.13
13 comps 14 comps 15 comps 16 comps 17 comps
X 97.83 98.69 99.35 99.82 100.00
Apps 85.55 85.55 89.68 92.81 93.43
The value of M is **17**



F.

Code-

```
fit.pls <- plsr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(fit.pls, val.type = "MSEP")
pred.pls <- predict(fit.pls, College.test, ncomp = 10)
mean((pred.pls - College.test$Apps)^2)
```

Output-

[1] 1031287

The test MSE with PLS model is 1031287, It is less than the PLS model but higher than linear, Ridge and Lasso models. The value of M for the PLS model is **M= 17**.

G.

Code-

```
test.avg <- mean(College.test$Apps)
lm.r2 <- 1 - mean((pred.lm - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
ridge.r2 <- 1 - mean((pred.ridge - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
lasso.r2 <- 1 - mean((pred.lasso - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
pcr.r2 <- 1 - mean((pred.pcr - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
pls.r2 <- 1 - mean((pred.pls - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
```

test.avg
lm.r2
ridge.r2
lasso.r2
pcr.r2
pls.r2

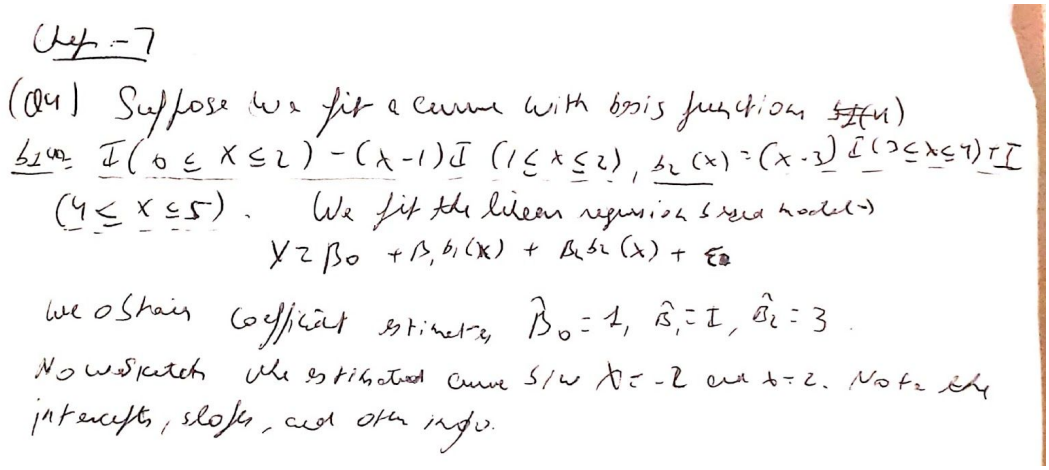
Output-

```
> test.avg
[1] 2889.411
> lm.r2
[1] 0.9104228
> ridge.r2
[1] 0.9104252
> lasso.r2
[1] 0.910428
> pcr.r2
[1] 0.8369703
> pls.r2
[1] 0.9099696
```

The accuracy for linear model is 0.9104, for ridge regression is 0.9104, for lasso 0.9104, for ocr 0.8369, for pls 0.9099. The test errors for the first 3 parts are similar (**LM,Ridge and Lasso**), but it is pretty different for **PCR and PLS**.

2. Exercises 4, 9, and 10 from Section 7.9 of our textbook.

Q4.



Chap-7

(Q4) Suppose we fit a curve with basis functions $b_1(x) = I(0 \le x \le 2) - (x-1)I(1 \le x \le 2)$, $b_2(x) = (x-3)I(3 \le x \le 4) + I(4 \le x \le 5)$. We fit the linear regression model-

$$Y = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) + \epsilon$$

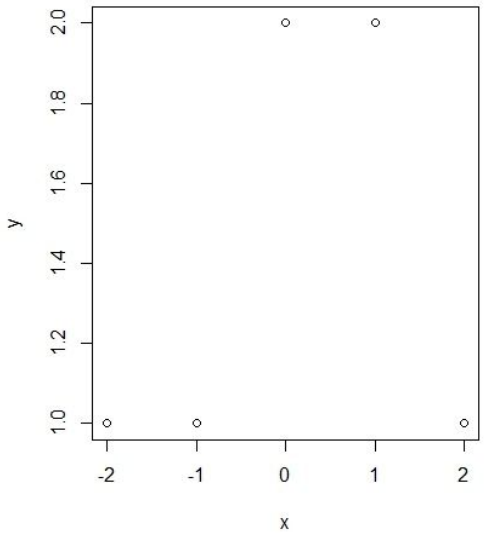
We obtain coefficient estimates $\hat{\beta}_0 = 1$, $\hat{\beta}_1 = 1$, $\hat{\beta}_2 = 3$.

Now sketch the estimated curve $f(x)$ over $-2 \le x \le 2$. Note the intercepts, slopes, and other info.

Code-

```
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)
```

Output-



The curve is constant between −2 and 0 : y=1, constant between 0 and 1 : y=2, and linear between 1 and 2 : y=3−x.

Q9

A.

code-

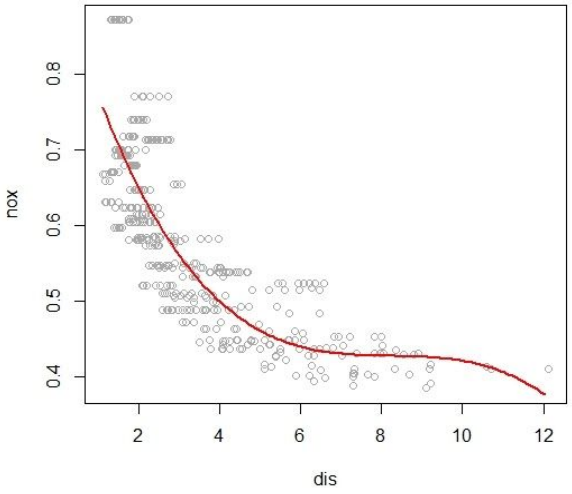
```
set.seed(1)
library(MASS)
attach(Boston)
lm.fit= lm(nox~ poly(dis,3), data=Boston)
summary(lm.fit)
```

Output-

```
Call:
lm(formula = nox ~ poly(dis, 3), data = Boston)
Residuals:
     Min       1Q   Median       3Q
-0.121130 -0.040619 -0.009738  0.023385
     Max
 0.194904
Coefficients:
             Estimate Std. Error
(Intercept)    0.554695   0.002759
poly(dis, 3)1 -2.003096   0.062071
poly(dis, 3)2  0.856330   0.062071
poly(dis, 3)3 -0.318049   0.062071
            t value Pr(>|t|)
(Intercept)   201.021  < 2e-16 ***
poly(dis, 3)1 -32.271  < 2e-16 ***
poly(dis, 3)2  13.796  < 2e-16 ***
poly(dis, 3)3  -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
```

Code-

```
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)
```



The summary shows that all **polynomial terms are significant while predicting nox using dis**.
Plot shows a **smooth curve fitting the data fairly well**.

B.

We plot polynomials of **degrees 1 to 10 and save train RSS**

Code-

```
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
```

Output-

```
[1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257
 [7] 1.849484 1.835630 1.833331 1.832171
```

As expected, **train RSS decreases with degree of polynomial**.

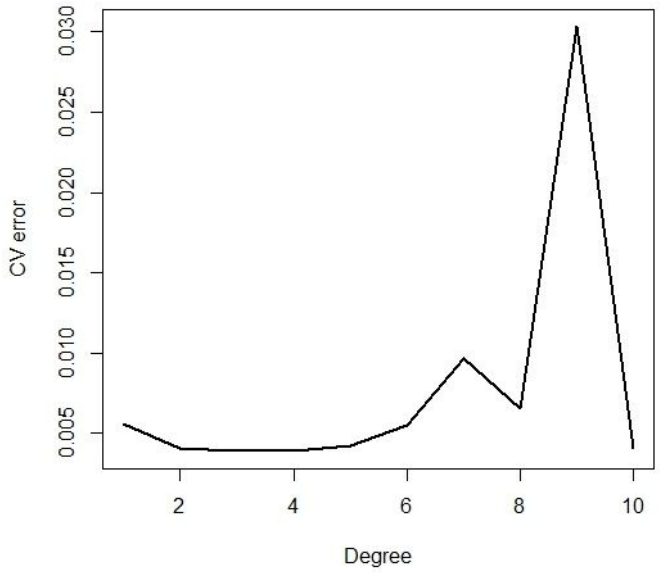
C.

We use a **10-fold cross validation to pick the best polynomial degree.**

Code-

```
library(boot)
all.deltas= rep(NA,10)
for (i in 1:10) {
  glm.fit = glm(nox ~ poly(dis, i), data = Boston)
  all.deltas[i] = cv.glm(Boston, glm.fit, K = 10)$delta[2]
}
plot(1:10, all.deltas, xlab = "Degree", ylab = "CV error", type = "l", pch = 20,
    lwd = 2)
```

Output-



A **10-fold CV shows that the CV error reduces as we increase degree from 1 to 3**, stay almost **constant till degree 5**, and then **starts increasing for higher degrees**. We pick **4 as the best polynomial degree**.

D.

We see that dis has limits of about **1 and 13 respectively**. We split this **range in roughly equal 4 intervals** and establish knots at **[4,7,11]**.

Code-

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

Output-

```
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
(Intercept)                        0.73926    0.01331  55.537
bs(dis, df = 4, knots = c(4, 7, 11))1 -0.08861    0.02504  -3.539
bs(dis, df = 4, knots = c(4, 7, 11))2 -0.31341    0.01680 -18.658
bs(dis, df = 4, knots = c(4, 7, 11))3 -0.26618    0.03147  -8.459
bs(dis, df = 4, knots = c(4, 7, 11))4 -0.39802    0.04647  -8.565
bs(dis, df = 4, knots = c(4, 7, 11))5 -0.25681    0.09001  -2.853
bs(dis, df = 4, knots = c(4, 7, 11))6 -0.32926    0.06327  -5.204
                                  Pr(>|t|)
```

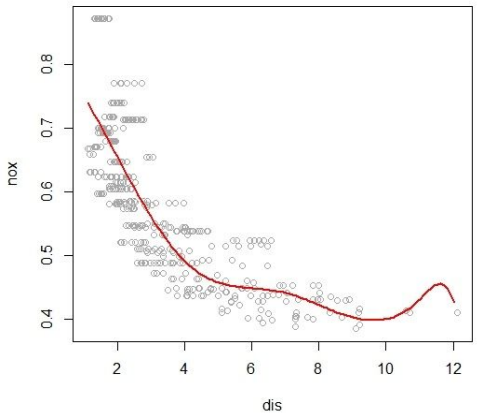
(Intercept) < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 11))1 0.00044 ***
bs(dis, df = 4, knots = c(4, 7, 11))2 < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 11))3 3.00e-16 ***
bs(dis, df = 4, knots = c(4, 7, 11))4 < 2e-16 ***
bs(dis, df = 4, knots = c(4, 7, 11))5 0.00451 **
bs(dis, df = 4, knots = c(4, 7, 11))6 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

Code-

```
sp.predict= predict(sp.fit, list(dis=dis.grid))
plot(nox~dis, data= Boston, col="darkgrey")
lines(dis.grid,sp.predict, col="red", lwd=2)
```

Output-



The summary shows **that all terms in spline fit are significant**. Plot shows that the spline fits data well **except at the extreme values of dis, (especially dis>10)**.

E.

We fit regression **splines with dfs between 3 and 16**.

Code-

```
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)]
```

Output-

[1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535
[9] 1.796992 1.788999 1.782350 1.781838 1.782798 1.783546

Train **RSS continuously decreases till df=14** and then **slightly increases for df=15 and df=16**.

F.

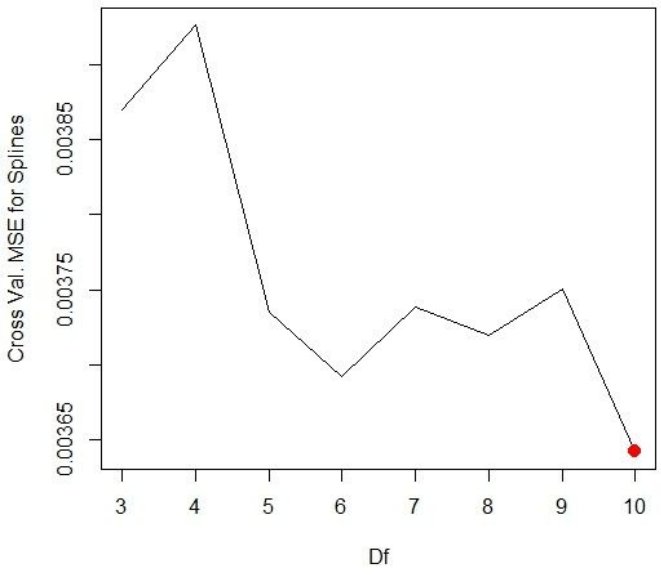
Finally, **we use a 10-fold cross validation to find best df**. We try all integer values of **df between 3 and 16**.

Code-

```
library(boot)
set.seed(42)
spline.mse=c()
for(df in 3:10){
  Boston.model=model.frame(nox~bs(dis,df=df),data=Boston)
  names(Boston.model)=c('nox','bs.dis')
# Note that because we are automatically setting the cutoffs we must do so in the entire
dataset, otherwise predictions cannot be made.
  spline.fit=glm(nox~bs.dis,data=Boston.model)
  mse=cv.glm(spline.fit,data=Boston.model,K=10)$delta[1]
  spline.mse=c(spline.mse,mse)
}

plot(3:10,spline.mse,type='l',xlab='Df',ylab='Cross Val. MSE for Splines')
x=which.min(spline.mse)
points(x+2,spline.mse[x],col='red',pch=20,cex=2)
```

Output-



It is clear that **fitting model with 10 degree of freedom performs well in this dataset**.

Q10.

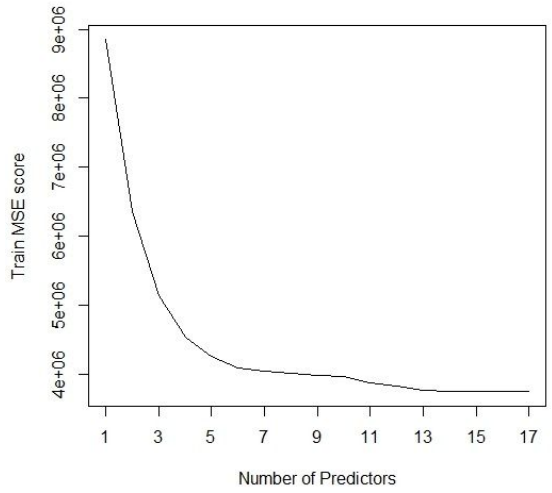
A.

```
library(ISLR)
set.seed(25)
train= sample(1:nrow(College),500)
test= -train
```

Code-

```
library(leaps)
forward= regsubsets(Outstate~., data=College, method='forward',nvmax=17)
plot(1/nrow(College)*summary(forward)$rss,type='l',xlab='Number of Predictors',ylab='Train MSE score',xaxt='n')
axis(side=1,at=seq(1,17,2),labels= seq(1,17,2))
```

Output-



From the plot we see that a good choice is the model with **7 predictors**. The predictors in this model are shown below.

Code-

```
which(summary(forward)$which[7,-1])
```

Output-

```
PrivateYes  Room.Board    Personal         PhD perc.alumni
         1           9          11          12          15
     Expend   Grad.Rate
         16          17
```

B.

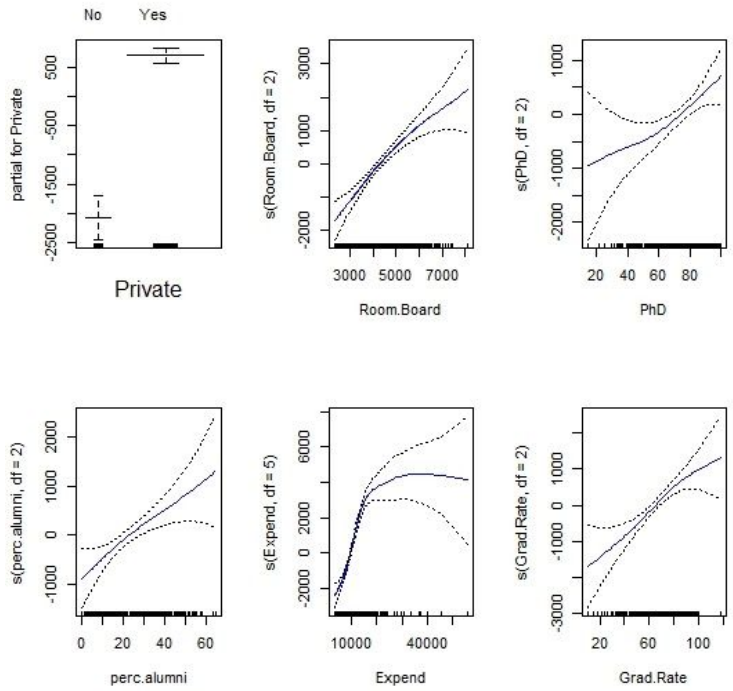
Here we fit a **GAM** by making use of smoothing splines for each of the **predictors selected, except 'Private' since it is a qualitative predictor.**

Code-

```
install.packages('gam')
library(gam)
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")

Output-



C.

Code-

gam.pred= predict(gam.fit, College[test,])

gam.mse= mean((College[test,'Outstate']-gam.pred)^2)

gam.mse

Output-

[1] 3467850

Evaluating the model on the **test set performs better than in the training set, since the MSE obtained is lower on the test set. Furthermore, below we can see that about 78% of the variance encountered in the data** is explained by this model.

Code-

gam.tss= mean((College[test,'Outstate']- mean(College[test,'Outstate']))^2)

test.rss= 1- gam.mse/gam.tss

test.rss

Output-

[1] 0.7558073

So test RSS= 0.7558073

D.

Code-

summary(gam.fit)

Output-

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
-7316.59	-1036.66	-11.26	1233.02	7951.98

(Dispersion Parameter for gaussian family taken to be 3505723)

Null Deviance: 8551607948 on 499 degrees of freedom

Residual Deviance: 1700276821 on 485.0003 degrees of freedom

AIC: 8970.662

Number of Local Scoring Iterations: 2

Anova for Parametric Effects

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Private	1	2346158114	2346158114	669.236	< 2.2e-16 ***
s(Room.Board, df = 2)	1	1829593310	1829593310	521.888	< 2.2e-16 ***
s(PhD, df = 2)	1	614859102	614859102	175.387	< 2.2e-16 ***
s(perc.alumni, df = 2)	1	270205500	270205500	77.076	< 2.2e-16 ***
s(Expend, df = 5)	1	673643000	673643000	192.155	< 2.2e-16 ***
s(Grad.Rate, df = 2)	1	95797265	95797265	27.326	2.559e-07 ***
Residuals	485	1700276821	3505723		

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	2.6681	0.1030
s(PhD, df = 2)	1	1.1851	0.2769
s(perc.alumni, df = 2)	1	0.5818	0.4460
s(Expend, df = 5)	4	24.3984	<2e-16 ***
s(Grad.Rate, df = 2)	1	1.8859	0.1703

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

From the output of ANOVA we can not that there is a significant evidence that a non-linear relationship for Expend,Grad.Rate,PhD,Personal, and Room.Board is present.