Homework 06 Solutions

2022-10-25

36-617: Applied Linear Models
Fall 2022
Solutions

```
## library(arm)  ## includes lme4, MASS, Matrix

library(ggplot2); theme_set(theme_bw())
library(gridExtra) ## to arrange ggplots...

## library(GGally) ## for ggpairs...


library(leaps) ## regsubsets(), summary(), coef()

## library(car)   ## subsets(), mmps(), vif(), etc.

## library(marginalmodelplots) ## because mmps doesn't work for glm's...
## library(DHARMa)  ## for better glm residual plots...

## library(tidyverse) ## at last...

## library(foreign)  ## for "read.dta" for problem 3

library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.3
```

```
library(splines)

library(boot)

library(mgcv)

set.seed(1000000)  ## repeatable "random" selection of CV subsets...
```

Problem 1: ISLR #9, p. 324.

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 nox as the response.

Problem 1(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.

```
data(Boston)

str(Boston)
```

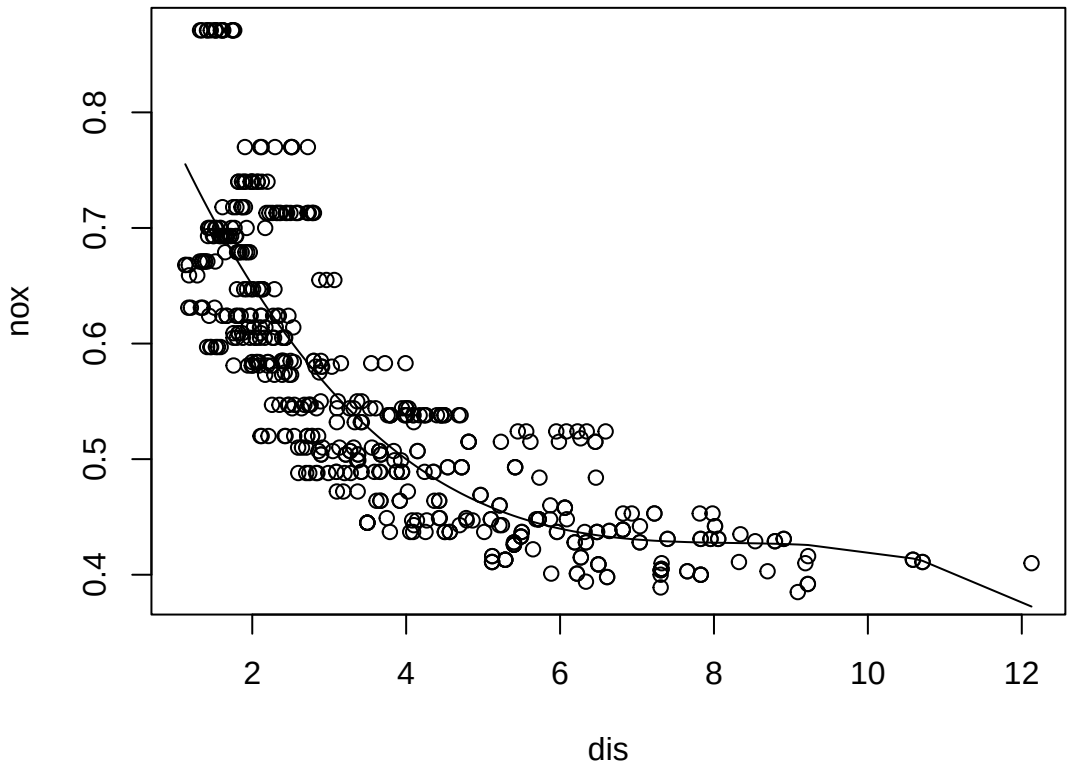
```
## 'data.frame':    506 obs. of  13 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
## help(Boston)   -- to get the definitions of the variables

lm.1 <- glm(nox ~ poly(dis,3), data=Boston)
summary(lm.1)
```

```
##
## Call:
## glm(formula = nox ~ poly(dis, 3), data = Boston)
##
## Deviance 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
##
## (Dispersion parameter for gaussian family taken to be 0.003852802)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9341  on 502  degrees of freedom
## AIC: -1370.9
##
## Number of Fisher Scoring iterations: 2
```

```
plot(nox ~ dis, data=Boston)
fits <- predict(lm.1)
```

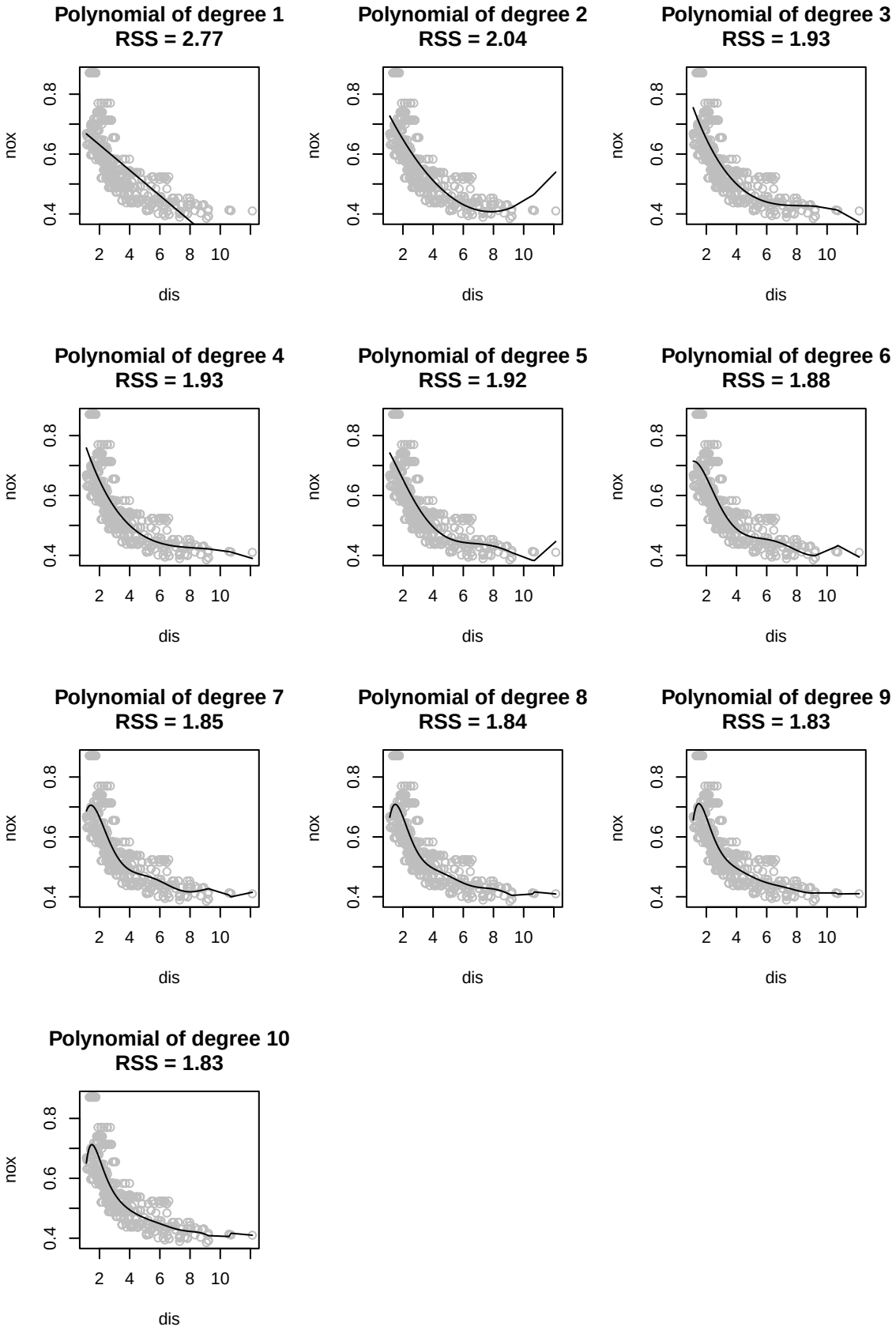
```
ord <- order(Boston$dis)
lines(fits[ord] ~ dis[ord], data=Boston)
```



Problem 1(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.

```
par(mfrow=c(4,3))
for (i in 1:10) {
  lm.fit <- glm(nox ~ poly(dis,i), data=Boston)
  rss <- sum(residuals(lm.fit)^2)
  plot(nox ~ dis, data=Boston, col="grey",
       main=paste0("Polynomial of degree ",i,"\nRSS = ",round(rss,2)))
  fits <- predict(lm.fit)
  ord <- order(Boston$dis)
  lines(fits[ord] ~ dis[ord], data=Boston)
}
```


Polynomial of degree 1
RSS = 2.77
Polynomial of degree 2
RSS = 2.04
Polynomial of degree 3
RSS = 1.93
Polynomial of degree 4
RSS = 1.93
Polynomial of degree 5
RSS = 1.92
Polynomial of degree 6
RSS = 1.88
Polynomial of degree 7
RSS = 1.85
Polynomial of degree 8
RSS = 1.84
Polynomial of degree 9
RSS = 1.83
Polynomial of degree 10
RSS = 1.83
nox
dis
0.4
0.6
0.8
2
4
6
8
10

Problem 1(c)

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

```
## From help(cv.glm, package=boot) we see that
## (1) cv.glm uses mean squared prediction error as the cross-validation criterion
## (2) We want $delta[1] as the raw cross-validation mean squared prediction error

## First we try LOOCV...

cv.err <- NULL
for (i in 1:10) {
  lm.fit <- glm(nox ~ poly(dis,i), data=Boston)
  cv.err <- c(cv.err,cv.glm(Boston,lm.fit)$delta[1])
}
cv.err
```

```
##  [1] 0.005523868 0.004079449 0.003874762 0.003887521 0.004164865 0.005384278
##  [7] 0.011068782 0.008121397 0.017616356 0.004430276
```

```
cat(paste("Optimal LOOCV polynomial degree =",which(cv.err==min(cv.err)),"\n"))
```

```
## Optimal LOOCV polynomial degree = 3
```

```
## Now let's try 10-fold cross-validation...

cv.err <- NULL
for (i in 1:10) {
  lm.fit <- glm(nox ~ poly(dis,i), data=Boston)
  cv.err <- c(cv.err,cv.glm(Boston,lm.fit, K=10)$delta[1])
}
cv.err
```

```
##  [1] 0.005501407 0.004084270 0.003896280 0.003879808 0.004152568 0.005817020
##  [7] 0.010192597 0.011093052 0.017637806 0.004047557
```

```
cat(paste("Optimal 10-fold CV polynomial degree =",which(cv.err==min(cv.err)),"\n"))
```

```
## Optimal 10-fold CV polynomial degree = 4
```

Leave-one-out cross validation (LOOCV) does not depend on a random selection of folds; LOOCV uses all n folds with training set of size $n-1$ and test set of size 1 (the one observation we leave out). On the other hand K-fold cross-validation will depend somewhat on what random set of folds get selected.

Here, LOOCV suggests that the optimal polynomial fit has degree 3, and 10-fold cross-validation suggests degree 4. While LOOCV will always choose degree 3 for this data (it is not random), the degree chosen by 10-fold cross-validation may change, depending on the particular random folds that are chosen. We will use degree 3 in the rest of this problem.

Problem 1(d)

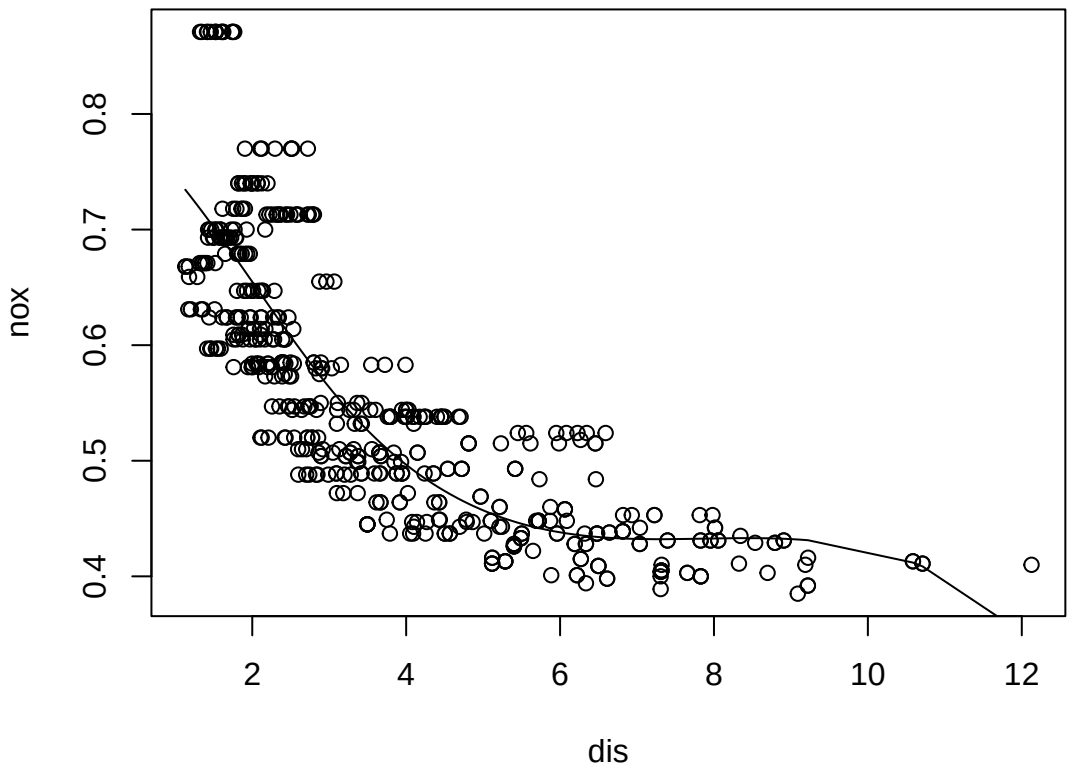
Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```
lm.2 <- glm(nox ~ bs(dis,df=4),data=Boston)
summary(lm.2)
```

```
##
## Call:
```

```
## glm(formula = nox ~ bs(dis, df = 4), data = Boston)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.124622  -0.039259  -0.008514   0.020850   0.193891
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.73447    0.01460  50.306  < 2e-16 ***
## bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.003837874)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9228  on 501  degrees of freedom
## AIC: -1371.9
##
## Number of Fisher Scoring iterations: 2
```

```
plot(nox ~ dis, data=Boston)
fits <- predict(lm.2)
ord <- order(Boston$dis)
lines(fits[ord] ~ dis[ord], data=Boston)
```



According to help(bs,package=splines):

- *The default degree of the spline is 3.*
- *When we specify "df=4", the bs() function chooses df − degree = 4 − 3 = 1 internal knot, and also one knot each at the min and max of "dis", for a total of 3 knots. The internal knot is at the median of "dis".*

This means we are fitting one cubic between min(dis) and median(dis), and another cubic between median(dis) and max(dis).

Problem 1(e)

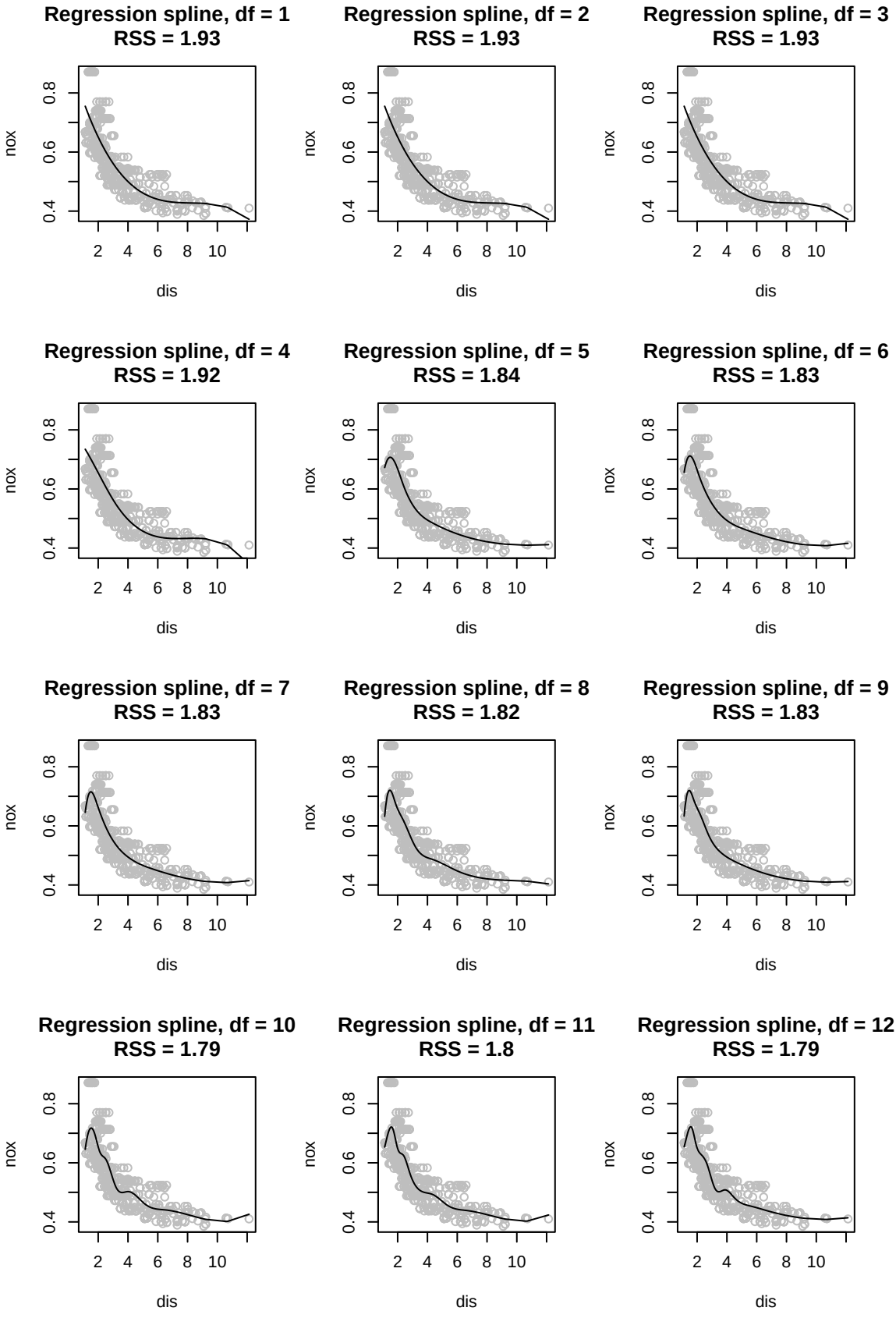
Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```
par(mfrow=c(4,3))
for (i in 1:12) {
  lm.fit <- glm(nox ~ bs(dis,df=i), data=Boston)
  rss <- sum(residuals(lm.fit)^2)
  plot(nox ~ dis, data=Boston, col="grey",
       main=paste0("Regression spline, df = ",i,"\nRSS = ",round(rss,2)))
  fits <- predict(lm.fit)
  ord <- order(Boston$dis)
  lines(fits[ord] ~ dis[ord], data=Boston)
```

```
}
```

```
## Warning in bs(dis, df = i): 'df' was too small; have used 3
```

```
## Warning in bs(dis, df = i): 'df' was too small; have used 3
```


Regression spline, df = 1
RSS = 1.93
Regression spline, df = 2
RSS = 1.93
Regression spline, df = 3
RSS = 1.93
Regression spline, df = 4
RSS = 1.92
Regression spline, df = 5
RSS = 1.84
Regression spline, df = 6
RSS = 1.83
Regression spline, df = 7
RSS = 1.83
Regression spline, df = 8
RSS = 1.82
Regression spline, df = 9
RSS = 1.83
Regression spline, df = 10
RSS = 1.79
Regression spline, df = 11
RSS = 1.8
Regression spline, df = 12
RSS = 1.79
nox
dis
0.4
0.6
0.8
2
4
6
8
10

From the warnings above and what we learned about the bs() function in part (d), when the formula "df − degree" says we should use 0 knots or a negative number of knots, the bs() function just sticks a knot at the median again. Effectively, when we specify a df that is too small, the bs() function just increases the df to the minimum acceptable, which is df=3 in our case.

Problem 1(f)

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.

```
## again, we can try LOOCV first...
cv.err <- NULL
for (i in 1:12) {
  lm.fit <- glm(nox ~ bs(dis,df=i), data=Boston)
  cv.err <- c(cv.err,cv.glm(Boston,lm.fit)$delta[1])
}
cv.err
```

```
##  [1] 0.003874762 0.003874762 0.003874762 0.003893623 0.003704252 0.003704711
##  [7] 0.003711441 0.003699853 0.003731180 0.003692067 0.003715438 0.003701170
```

```
cat(paste("Optimal LOOCV df for bs() =",which(cv.err==min(cv.err)),"\n"))
```

```
## Optimal LOOCV df for bs() = 10
```

```
## ...and then 10-fold cv...
cv.err <- NULL
for (i in 1:12) {
  lm.fit <- glm(nox ~ bs(dis,df=i), data=Boston)
  cv.err <- c(cv.err,cv.glm(Boston,lm.fit, K=10)$delta[1])
}
cv.err
```

```
##  [1] 0.003891716 0.003879143 0.003886573 0.003913647 0.003687029 0.003698626
##  [7] 0.003684441 0.003694159 0.003737521 0.003690349 0.003712073 0.003709678
```

```
cat(paste("Optimal 10-fold CV df for bs() =",which(cv.err==min(cv.err)),"\n"))
```

```
## Optimal 10-fold CV df for bs() = 7
```

The 10-fold cross-validation result is somewhat unstable here and depends on which random folds we get. In different runs, I've gotten df=5, 6, 7, 8 and 10 as optimal. On the other hand, the LOOCV result is stable and non-random (because it considers all $n = 506$ folds of size $n - 1$, leaving one observation out at a time as a "test dataset"), and the result is df=10.

As a final check I will compare AIC and BIC for the 3rd degree polynomial and several of the regression spline models:

```
lm.poly3 <- lm.1
lm.bs3 <- lm.2
lm.bs5 <- glm(nox ~ bs(dis,df=5), data=Boston)
lm.bs6 <- glm(nox ~ bs(dis,df=6), data=Boston)
lm.bs7 <- glm(nox ~ bs(dis,df=7), data=Boston)
lm.bs8 <- glm(nox ~ bs(dis,df=8), data=Boston)
lm.bs10 <- glm(nox ~ bs(dis,df=10), data=Boston)

models <- list(lm.poly3,lm.bs3,lm.bs5,lm.bs6,lm.bs7,lm.bs8,lm.bs10)
names(models) <- strsplit("lm.poly3,lm.bs3,lm.bs5,lm.bs6,lm.bs7,lm.bs8,lm.bs10",
                          ",")[[1]]
```

```
comparison <- data.frame(
  logLik = sapply(models,logLik),
  df = sapply(models, function(x) attributes(logLik(x))$df),
  AIC = sapply(models,AIC),
  BIC = sapply(models,BIC)
)
rownames(comparison) <- names(models)

round(comparison,2)
```

```
##          logLik df      AIC      BIC
## lm.poly3 690.44  5 -1370.88 -1349.75
## lm.bs3   691.93  6 -1371.85 -1346.50
## lm.bs5   703.04  7 -1392.07 -1362.49
## lm.bs6   703.89  8 -1391.78 -1357.97
## lm.bs7   704.45  9 -1390.91 -1352.87
## lm.bs8   706.24 10 -1392.49 -1350.22
## lm.bs10  709.67 12 -1395.34 -1344.63
```

We see that AIC likes the regression spline with df = 10 best, and BIC likes the df = 5 regression spline model best. Referring back to the regression spline plots, the curves for df = 5,6 or 7 seem preferable to the somewhat wiggly df = 8 or 10 plots. (This is my usual "interpretability" bias: smooth curves are generally easier to tell a story about than wiggly ones. If your goal is prediction, however, you may indeed prefer the df = 10 plot, which did better with AIC and LOOCV.)

Problem 2: ISLR #10, pp. 324–325.

This question relates to the College data set.

Problem 2(a)

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.

```
data(College)
str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
```

```
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

```
## help(College)  -- to get the definitions of the variables
```

We have $n = 777 < 1000$ which is a smallish sample size for cross-validation, so I will use a 50-50 split (see lecture 8 from class).

```
n <- dim(College)[1] ## should be n=777
train <- sample(1:n,size=floor(n/2))
train <- ifelse(1:n %in% train,T,F)
test <- !train
train.data <- College[train,]
test.data <- College[test,]

n.train <- dim(train.data)[1]
n.test <- dim(test.data)[1]


forward <- regsubsets(Outstate ~ ., data=train.data,method="forward",nvmax=18)

tmp <- summary(forward)
p.max <- dim(train.data)[2]
p <- 1:(p.max-1)

results <- data.frame(tmp$which,rss=tmp$rss,adjr2=tmp$adjr2,
                      bic=n.train*log(tmp$rss)+log(n.train)*(p+2),
                      aic=n.train*log(tmp$rss)+2*(p+2),
                      caic=n.train*log(tmp$rss) + 2*(p+2) +
                        2*(p+2)*(p+1)/(n.train-p-1))

## results

names(results)
```

```
##  [1] "X.Intercept." "PrivateYes"   "Apps"         "Accept"       "Enroll"
##  [6] "Top10perc"    "Top25perc"    "F.Undergrad"  "P.Undergrad"  "Room.Board"
## [11] "Books"        "Personal"     "PhD"          "Terminal"     "S.F.Ratio"
## [16] "perc.alumni"  "Expend"       "Grad.Rate"    "rss"          "adjr2"
## [21] "bic"          "aic"          "caic"
```

```
names(results)[2] <- "Private"

minimize <- function(res,col) {
  obj <- res[,col]
  k <- (1:length(obj))[obj==min(obj)]
  return(res[k,])
}


sel.BIC <- minimize(results,"bic")[2:p.max]
sel.AIC <- minimize(results,"aic")[2:p.max]

fla.BIC <- paste0("Outstate ~ ",paste(names(sel.BIC)[unlist(sel.BIC)],collapse=" + "))
fla.AIC <- paste0("Outstate ~ ",paste(names(sel.AIC)[unlist(sel.AIC)],collapse=" + "))
```

```
fwd.BIC <- lm(fla.BIC, data=train.data)
fwd.AIC <- lm(fla.AIC, data=train.data)
```

Just for fun, I compare the best AIC and BIC models with a partial F test, using the anova() function.

From the anova() output below we see that the AIC model adds the variables Top25perc, Books and GradRate to the BIC model.

Unsurprisingly, AIC and the F test prefer the more complex model for this split of the data. However, I will work with the simpler BiC model in what follows.

```
anova(fwd.BIC,fwd.AIC)
```

```
## Analysis of Variance Table
##
## Model 1: Outstate ~ Private + Accept + Enroll + Room.Board + Personal +
##     PhD + perc.alumni + Expend
## Model 2: Outstate ~ Private + Accept + Enroll + Top25perc + Room.Board +
##     Books + Personal + PhD + perc.alumni + Expend + Grad.Rate
##   Res.Df        RSS Df Sum of Sq      F  Pr(>F)
## 1    379 1553264546
## 2    376 1512317067  3  40947479 3.3935 0.01808 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the details of the results above depend strongly on the particular split of the data into training and test subsets. A different split, with a different set.seed() seed, could produce different (and possibly non-nested) "best" AIC and BIC models; the result of the anova() test (when the models turn out to be nested) can also change.

Problem 2(b)

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.

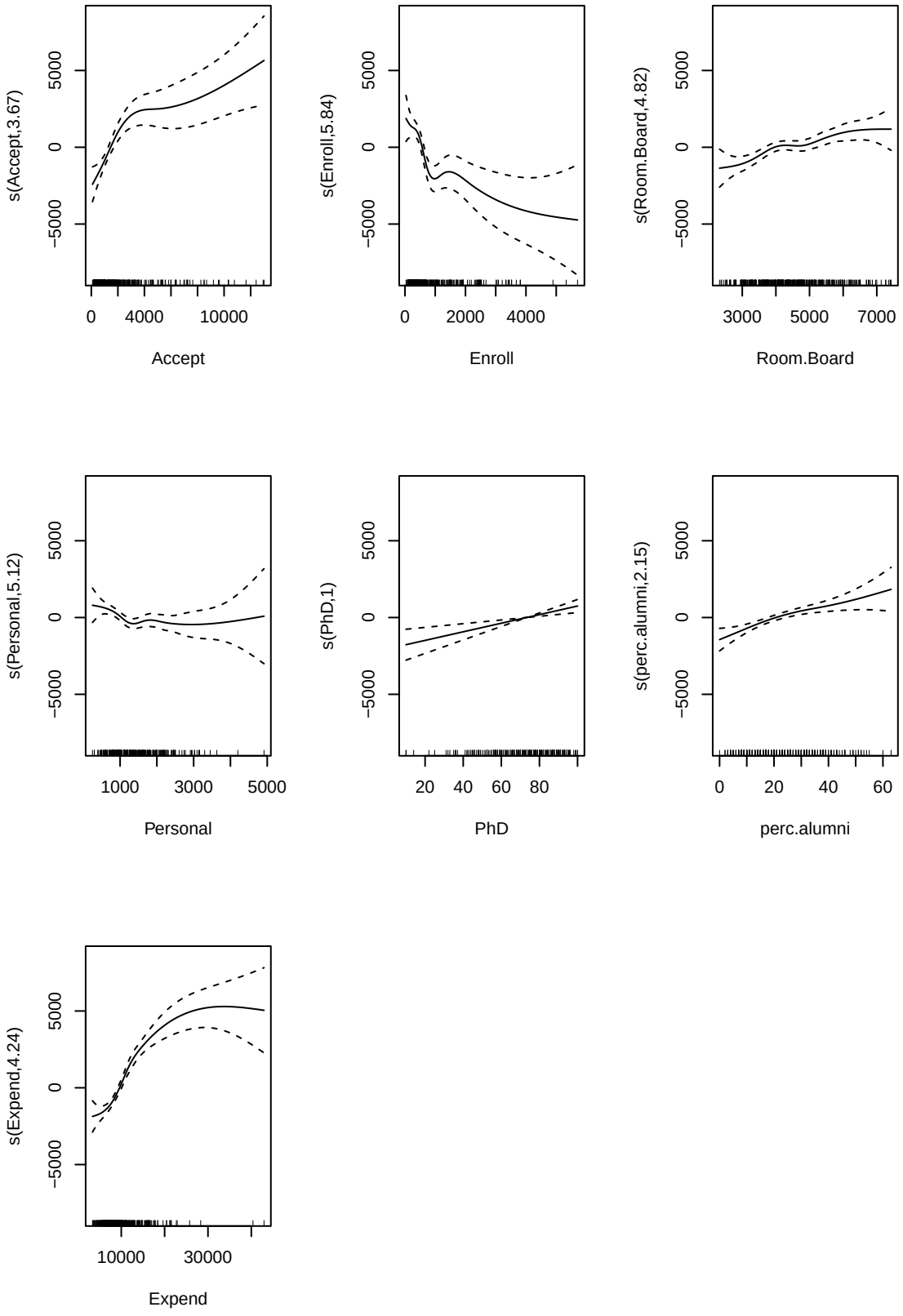
Since "Private" is in the model, but "Private" is a 0/1 variable, it doesn't get a transformation. But all the other variables are quantitative and so get a transformation

```
## Note that I am using cubic smoothing splines (bs='cr' in the code below)
## to be consistent with what we did in class, but you could equally well
## use mgcv's default "thin plate" splines...

fla.gam <- paste0("Outstate ~ Private + s(",
                  paste(names(sel.BIC)[-1][unlist(sel.BIC)[-1]],
                  collapse=",bs='cr') + s("),",bs='cr')")

gam.1 <- gam(formula(fla.gam), data=train.data,method="GCV.Cp")

summary(gam.1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Outstate ~ Private + s(Accept, bs = "cr") + s(Enroll, bs = "cr") +
##     s(Room.Board, bs = "cr") + s(Personal, bs = "cr") + s(PhD,
```

```
##     bs = "cr") + s(perc.alumni, bs = "cr") + s(Expend, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8720.5      271.9  32.071  < 2e-16 ***
## PrivateYes    2618.3      346.8   7.549 3.65e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(Accept)       3.666  4.275  8.276 2.70e-06 ***
## s(Enroll)       5.840  6.524  5.509 9.60e-06 ***
## s(Room.Board)   4.821  5.801  4.829 0.000168 ***
## s(Personal)     5.120  6.052  2.393 0.025545 *
## s(PhD)          1.000  1.000 12.364 0.000494 ***
## s(perc.alumni) 2.150  2.733 10.397 8.82e-06 ***
## s(Expend)       4.244  4.925 32.353  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.796   Deviance explained = 81.1%
## GCV = 3.5935e+06  Scale est. = 3.3264e+06  n = 388
```

```
plot(gam.1,pages=1)
```


s(Accept,3.67)
Accept
s(Enroll,5.84)
Enroll
s(Room.Board,4.82)
Room.Board
s(Personal,5.12)
Personal
s(PhD,1)
PhD
s(perc.alumni,2.15)
perc.alumni
s(Expend,4.24)
Expend

It appears that PhD and perc.alumni not need transformations, while Accept and Expend clearly do. In particular it looks like

- *Out of state tuition increases with institutional expenditures per student, up to a point at which it levels off (perhaps this is the highest tuition that the market will bear)*
- *Out of state tuition also increases with the number of students accepted (which seems a bit suprising to me) but the influence of more students accepted, on out of state tuition, lessens after roughly Accept=4000.*

Problem 2(c)

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

First we do mean squared prediction error for the two models

```
mse.BIC <- mean((test.data$Outstate - predict(fwd.BIC,newdata=test.data))^2)
mse.gam <- mean((test.data$Outstate - predict(gam.1,newdata=test.data))^2)

mse.results <- data.frame("Prediction MSE"=c(fwd.BIC=mse.BIC,gam.1=mse.gam))
round(mse.results)
```

```
##         Prediction.MSE
## fwd.BIC        4524190
## gam.1          3520081
```

Now we do a goodness-of-fit comparison:

```
test.BIC <- lm(formula(fla.BIC),data=test.data)
test.gam <- gam(formula(fla.gam),data=test.data)
models <- list(test.BIC,test.gam)
names(models) <- strsplit("test.BIC,test.gam",",")[[1]]

comparison <- data.frame(
  logLik = sapply(models,logLik),
  df = sapply(models, function(x) attributes(logLik(x))$df),
  AIC = sapply(models,AIC),
  BIC = sapply(models,BIC)
)
rownames(comparison) <- names(models)

round(comparison,2)
```

```
##            logLik    df     AIC     BIC
## test.BIC -3502.34 10.00 7024.68 7064.32
## test.gam -3442.53 20.45 6925.96 7007.01
```

All three comparisons, prediction MSE, AIC, and BIC, prefer the gam model in this example.

Explanation: The extra flexibility of the nonparametric transformations of the predictors provide better predictions, reducing the mean squared prediction error by $(3.5200809 \times 10^6)/(4.5241897 \times 10^6) = 0.78$*, as well as dramatic improvements in fit as reflected by reductions of 60–100 in AIC and BIC.*

Problem 2(d)

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

From the transformation plots above, it seems clear that Accept and Expand have a non-linear relationship with the response. There really isn't a clear case for any of the others, though one could make an argument

for Enroll and Personal.

One could more formally assess whether a nonlinear term is needed for Personal, for example, by comparing the model with s(Personal) as a predictor, with the model that just uses Personal as a predictor.

Once again, these results may vary, depending on what split into training and test data sets you are working with.

Problem 3 (Based on Gelman & Hill, Chapter 9, #4).

The table below describes a hypothetical experiment on 2400 persons. Each row of the table specifies a category of person, as defined by his or her pre-treatment predictor x, treatment indicator T, and potential outcomes y^0 and y^1. (For simplicity, we assume—unrealistically—that all people in this experiment fit into one of these eight categories).

Category	# persons in category	x	T	y^0	y^1
1	300	0	0	4	6
2	300	1	0	4	6
3	500	0	1	4	6
4	500	1	1	4	6
5	200	0	0	10	12
6	200	1	0	10	12
7	200	0	1	10	12
8	200	1	1	10	12

In making this table we are assuming omniscience, so that we know both y^0 and y^1 for all observations. But the (non-omniscient) investigator woud only observe x, T, and $y = y^T$ for each unit. (For example, a person in category 1 would have $x = 0$, $T = 0$ and $y = 4$, and a person in category 3 would have $x = 0$, $T = 1$, and $y = 6$.)

Let's start by putting the data above into R...

```
data_mat <- matrix(c(
  1, 300, 0, 0, 4, 6,
  2, 300, 1, 0, 4, 6,
  3, 500, 0, 1, 4, 6,
  4, 500, 1, 1, 4, 6,
  5, 200, 0, 0, 10, 12,
  6, 200, 1, 0, 10, 12,
  7, 200, 0, 1, 10, 12,
  8, 200, 1, 1, 10, 12), byrow = TRUE, nrow = 8)

colnames(data_mat) <- c("Category", "num", "x",
                        "T", "y_0", "y_1")
df <- data.frame(data_mat)
```

Problem 3(a)

What is the true ACE (average causal effect), if we could observe y^1 and y^0 for every person in this population of 2400 persons?

```
N <- sum(df$num)
ace <- 1/N*sum(df$num * (df$y_1 - df$y_0))
ace
```

```
## [1] 2
```

We can actually directly see that for each individual, the difference between y^1 and y^0 is 2 (which means the average is also this value), but the above equation also shows us the the ACE is 2.

Problem 3(b)

Another population quantity is the mean of y for those who received the treatment, minus the mean of y for those who did not. What is the relationship between this quantity and the quantity you calculated in part (a)?

```
N_T <- sum(df$num[df$T == 1])
N_not_T <- sum(df$num[df$T == 0])

1/N_T * sum((df$num * df$y_1)[df$T == 1]) -
  1/N_not_T * sum((df$num * df$y_0)[df$T == 0])
```

```
## [1] 1.314286
```

If there was no other confounders we'd expect this value to approximate the ACE. Interestingly it does not! Below in part (d) we explore the data further and discover that there is another confounding variable not accounted for in the data.

Problem 3(c)

Suppose we draw a person randomly from this population or 2400 people. What is the probability that $T = 1$ for this person? If I tell you that $x = 1$ for this person, does that change the probability that $T = 1$? I.e., is $P[T = 1|x = 1] = P[T = 1]$?

```
prob_T <-sum(df$num * (df$T == 1)) / sum(df$num)

prob_T
```

```
## [1] 0.5833333
```

```
prob_T_given_x1 <- sum(df$num * (df$x == 1) * (df$T == 1)) /
  sum(df$num * (df$x == 1))

prob_T_given_x1
```

```
## [1] 0.5833333
```

We find that, in the above population, $P[T = 1] = 0.5833$, which is the same as $P[T = 1|x = 1]$.

Problem 3(d)

Is it plausible to believe this data came from a randomized experiment? Defend your answer.

From part (c) we know that $P[T = 1] = P[T = 1|x = 1]$, and more generally, $P[T = t, x = x] = P[T = t]P[x = x]$, i.e. T and X are independent. So, if we think x is the only pre-treatment covariate (as suggested in the beginning of the problem), it is plausible that this is data from a randomized experiemnt (since randomization forces T to be indpendent of any pre-treatment covariates).

However, looking back at the data table, it seems that there is a jump of 6 in the level of response (for both y^0 and y^1) going from the first four categories, vs. the last four categories. For example, if y is measuring a health outcome after a hospital stay, and the first four categories represent people who would go to hospital #1 and the second four categories represent people who would go to hospital #2, it appears that the first group of people are generally less healthy than the second group.

If we define another grouping variable

```
df$x2 <- c(1,1,1,1,0,0,0,0)
```

then (i) we can see that `x2` *and* `T` *are not independent:* `P[T=1|x2=1] = 1000/1600 = 0.625` $\neq$ `P[T=1|x2=0] = 400/800 = 0.5`*, and (ii) in the regression* `y ~ T + x2`*, the coefficient* $\hat{\beta}_T$ *would plausibly be an unbiased estimate of ACE. The simpler regression* `y ~ T` *(which is essentially what we were doing in part (b) above) would produce a biased* $\hat{\beta}_T$*.*

Problem 3(e)

For this hypothetical data, figure out (by hand, or by using R) the estimate and standard error of the coefficient of T in a regression of y on T and x. Is this a reasonable way to estimate the true ACE?

Below are two approaches to estimate the desired coefficient and se. Note that the coefficient values for the intercept and T *directly related to parts of the calculation in part (b). The intercept being the* `mean(y[T=0])` *and the coefficient of* T *being the difference between* `mean(y[T=1]) - mean(y[T=0])`*.*

Splitting the data into individual cases

```
new.df <- data.frame(0,0,0,0,0,0,0)  ## a dummy line so we can rbind to it
names(new.df) <- names(df)
for (i in 1:8) {
  for (j in 1:df[i,"num"]) {
    new.df <- rbind(new.df,df[i,])
  }
}
new.df <- new.df[-1,]  ## remove the dummy line

new.df$y <- ifelse(new.df$T==0,new.df$y_0,new.df$y_1)

summary(lm.splitting <- lm(y ~ T + x,data=new.df))
```

```
##
## Call:
## lm(formula = y ~ T + x, data = new.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.400 -1.886 -1.714  3.600  4.286
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.400e+00  1.058e-01   60.51   <2e-16 ***
## T           1.314e+00  1.163e-01   11.30   <2e-16 ***
## x           2.901e-15  1.147e-01    0.00        1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.81 on 2397 degrees of freedom
## Multiple R-squared:  0.05055,    Adjusted R-squared:  0.04976
## F-statistic: 63.81 on 2 and 2397 DF,  p-value: < 2.2e-16
```

Weighted regression approach

A weighted regression solution is faster than expanding out the full dataset,

```
df$y <- ifelse(df$T==0,df$y_0,df$y_1)

lm.weighted <- lm(y ~ T + x, data = df, weights = num)
summary(lm.weighted)  ## uncorrected summary
```

```
##
## Call:
## lm(formula = y ~ T + x, data = df, weights = num)
##
## Weighted Residuals:
##      1      2      3      4      5      6      7      8
## -41.57 -41.57 -38.33 -38.33  50.91  50.91  60.61  60.61
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.400      2.316   2.764   0.0397 *
## T              1.314      2.547   0.516   0.6279
## x              0.000      2.512   0.000   1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.52 on 5 degrees of freedom
## Multiple R-squared:  0.05055,    Adjusted R-squared:  -0.3292
## F-statistic: 0.1331 on 2 and 5 DF,  p-value: 0.8784
```

but requires a correction for the model's residual degrees of freedom. See this blogpost for more comments.

The correct residual degrees of freedom are

```
df.res <- sum(df$num) - length(lm.weighted$coefficients)
df.res
```

```
## [1] 2397
```

and then new s^2 *and* $SE(\hat{\beta}_T)$ *can be calculated with this new residual df.*

For now we will just observe that the point estimates are the same: Note that we get the same estimate either way:

```
round(rbind("b_T from lm.splitting"=summary(lm.splitting)$coef[2,1],
      "b_T from lm.weighted"=summary(lm.weighted)$coef[2,1]),2)
```

```
##                       [,1]
## b_T from lm.splitting 1.31
## b_T from lm.weighted  1.31
```

This is not *a reasonable way to estimate the ACE: Since we are not accounting for the variable* **x2**, *we get the same biased estimate as in part (b).*

If we include **x2** *also, we can get a better answer:*

```
corrected.lm.splitting <- lm(y ~ T + x + x2,data=new.df)
summary(corrected.lm.splitting)
```

```
##
## Call:
## lm(formula = y ~ T + x + x2, data = new.df)
##
## Residuals:
```

```
##        Min         1Q     Median         3Q        Max
## -3.731e-12 -9.000e-16  2.900e-15  3.900e-15  1.803e-12
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  1.000e+01  4.833e-15  2.069e+15  < 2e-16 ***
## T            2.000e+00  4.397e-15  4.549e+14  < 2e-16 ***
## x           -1.535e-14  4.304e-15 -3.566e+00  0.00037 ***
## x2          -6.000e+00  4.598e-15 -1.305e+15  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.054e-13 on 2396 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 5.977e+29 on 3 and 2396 DF,  p-value: < 2.2e-16
```

We can see that the coefficient $\hat{\beta}_T$ is now the correct value, 2, to many decimal places (and with a very tight confidence interval!).

Problem 4 (Based on Gelman & Hill, Chapter 9, #6).

You are consulting for a researcher who has performed a randomized trial where the treatment was a series of 26 weekly therapy sessions, the control was no therapy, and the outcome was self-report of emotional state one year later. However, most people in the treatment group did not attend every therapy session. In fact there was a good deal of variation in the number of therapy sessions actually attended. The researcher is concerned that her results represent "watered down'' estimates because of this variation and suggests adding in another predictor to the model: number of therapy sessions attended.

Problem 4(a)

What would you advise her about her suggestion? Carefully justify your answer in terms of our discussion about controlled experiments, observational studies, confounders, and so forth.

The treatment variable is no longer randomized in the way intended by the researcher, because individuals have selected their own level of treatment. This selection process is very likely confounded with the outcome variable, meaning that there is some unobserved variable that is correlated with both the treatment variable and the outcome.

For example, imagine that the therapy actually has no effect, but there is an "optimism" variable that determines both how many sessions a person attends and their emotional state at the end of the study. In other words, the people who engage the most with the treatment are those in the best state at the end of the study, but not because of the therapy. Then the number of sessions attended will strongly predict the outcome variable, but not because of an effect of the therapy.

An unobserved confounder could also obscure a real treatment effect. Suppose that instead of an optimism variable, there is a variable representing emotional wellbeing that is inversely correlated with the number of sessions attended. That is, those who start out the most depressed seek the most treatment, perhaps because they feel that they have the most to gain. Suppose the treatment does *have a beneficial effect that increases with the number of sessions attended, and suppose that everyone seeks out just enough treatment to get them to roughly the same level of emotional wellbeing at the end of the study. Then a coefficient for number of treatment sessions could end up being 0 even though there is a treatment effect!*

In sum: *whenever there is noncompliance in an experiment, there is potential for confounding due to selection effects. Including a variable for the amount of treatment received* does not *address this and can lead to highly misleading results.*

Problem 4(b)

Can you suggest another kind of analysis that might give her a better estimate of the treatment effect? Carefully explain why this might work. *(Hint: Think about the more sophisticated analyses presented in lecture 16 in class.)*

This is perfectly set up for an Instrumental Variable analysis. The assignment variable (assignment to treatment or control) is the instrument, and the number of sessions attended is the treatment variable. The assignment variable almost certainly affects the treatment variable, but it should have no effect on the outcome through other pathways. (That is, the mere fact of having been assigned to the treatment or control group shouldn't affect participants' emotional wellbeing at the end of the study.) An experimental design with noncompliance is almost always a good candidate for an IV analysis.

An analysis based on propensity score matching won't work here unless we have variables with which to construct the propensity score. The propensity score is the probability of receiving treatment as a function of some set of covariates X. That is, the propensity score is a function $\pi(x) := P(A = 1|X = x)$, where $A = 1$ means that someone receives treatment. We can define it more broadly as the probability of receiving a particular amount of treatment a: $\pi_a(x) = P(A = a|X = x)$. We want as rich a set of covariates X as possible in order to perform matching. If the investigator collected these prior to the start of the study, then this analysis may be possible.