CHAPTER 4

Additional Regression Topics

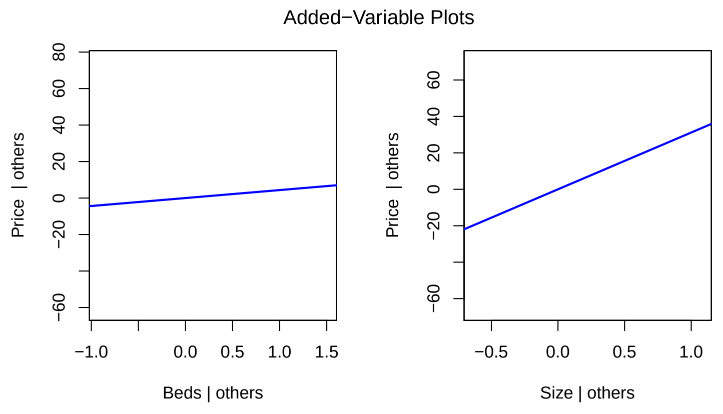
We now illustrate R code required for the various additional regression topics from Chapter 4 of the text by using many of the same examples given in that chapter.

4.1 Added Variable Plots

Example 4.1: House Prices

We could construct added variable plots in a step-by-step fashion, but the easy thing to do is to use an `avPlots()` command in the `car` package.

```
> library(car)
> modBoth <- lm(Price~Beds + Size, data=HousesNY)
> avPlots(modBoth)
```



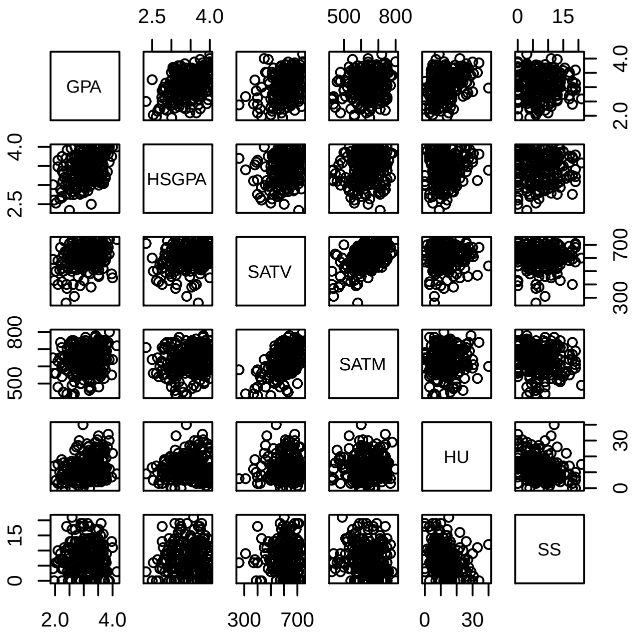
Added variable plot for Houses data using `avPlots`

4.2 Techniques for Choosing Predictors

Example 4.2: The GPA Data

We now illustrate R code for using automated procedures for model selection, using the same dataset the text uses. After getting the data, we ask for variable names. To replicate the matrix scatterplot in the text, we note that the variables we want in the plot—`GPA`, `HSGPA`, `SATV`, `SATM`, `HU`, and `SS`—are variables 1, 2, 3, 4, 6, and 7 among the 10 variables in the dataset. (We don't want the binary variables in the matrix plot.) Therefore, the call to `pairs` subsets out these columns from the `GPA` data frame.

```
> data(FirstYearGPA)
> names(FirstYearGPA)
 [1] "GPA"           "HSGPA"         "SATV"          "SATM"          "Male"
 [6] "HU"            "SS"            "FirstGen"      "White"         "CollegeBound"
> pairs(FirstYearGPA[,c(1,2,3,4,6,7)])
```



Scatterplot matrix for first-year GPA data

We compare GPAs for the values of our four binary predictors using boxplots. We use `par` to set up the graphics window to put four plots side by side. We then use four `boxplot` function calls, using the `xlab` argument to add a label under each boxplot.

```
> par(mfrow=c(1,4)) # puts all four plots side-by-side
> boxplot(GPA ~ Male, main="1=male; 0=female",horizontal=TRUE)
> boxplot(GPA ~ FirstGen, main="1=first generation; 0=not", horizontal=TRUE)
> boxplot(GPA ~ White, main="1=white; 0=others", horizontal=TRUE)
> boxplot(GPA ~ CollegeBound, main="1=more than half to college", horizontal=TRUE)
> par(mfrow=c(1,1)) # set graphics window back to one panel
```

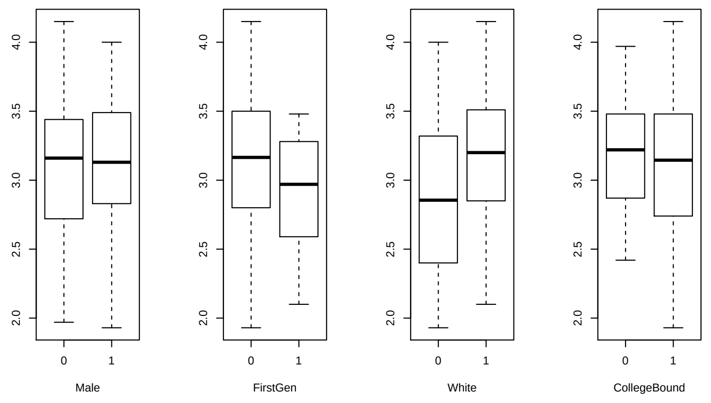


Figure 4.5 (page 157): GPA versus categorical predictors

Best Subsets

To obtain a best subsets analysis, we use a function in the `leaps` package called `regsubsets`.

```
> all <- regsubsets(GPA ~ ., nbest = 2, data=FirstYearGPA)
```

Notice here we use the compact option for specifying the model: `GPA ~ .` tells R to regress GPA on all of the predictors in the FirstYearGPA data frame.
We see the results of the `regsubsets()` command with

```
> summary(all)
```

(We have suppressed the output, which agrees with the Minitab output in the text.)

We can create a plot that shows which variables are in each model:

```
> plot(all, scale = "adjr2")
```

If we want to rank the models by Mallow's C_p rather than by adjusted R^2, then we use

```
> plot(all, scale = "Cp")
```

We can also get a listing of the C_p values for the models with

```
> summary(all)$cp  # Note that cp Is lowercase (not Cp)

[1] 42.181 74.541 21.683 22.210  6.531 11.407  3.900  6.426  3.892
4.764  4.849  5.080  6.081  6.766  8.036  8.046
```

We see that the ninth model has the smallest C_p. We need to look back at earlier output (suppressed here) to find that this is the model with `HSGPA`, `SATV`, `HU`, `SS`, and `White` as predictors.
Another way to look at the results of the `regsubets()` command is to use the `summaryHH()` command in the `HH` package.

```
> library(HH)
> summaryHH(all)
```

(Again, we suppress the output here.)

Backward Elimination

To use backward elimination, we use the `step()` function. We start by fitting the model with all predictors, and then we feed that model into the `step()` function.

```
> full <- lm(GPA ~ ., data = FirstYearGPA)
> step(full, direction = backward)


Start:  AIC=-410.16
GPA ~ HSGPA + SATV + SATM + Male + HU + SS + FirstGen + White +
    CollegeBound

               Df Sum of Sq    RSS     AIC
- SATM          1    0.0053 30.724 -412.12
- CollegeBound  1    0.0067 30.726 -412.11
- FirstGen      1    0.1031 30.822 -411.42
- Male          1    0.1052 30.824 -411.41
- SS            1    0.2556 30.975 -410.34
<none>                      30.719 -410.16
- SATV          1    0.3309 31.050 -409.81
- White         1    1.1545 31.873 -404.08
- HU            1    2.4409 33.160 -395.41
- HSGPA         1    6.4345 37.154 -370.51
```

```
Step:  AIC=-412.12
GPA ~ HSGPA + SATV + Male + HU + SS + FirstGen + White + CollegeBound

               Df Sum of Sq    RSS     AIC
- CollegeBound  1    0.0065 30.731 -414.07
- FirstGen      1    0.1072 30.832 -413.36
- Male          1    0.1421 30.866 -413.11
- SS            1    0.2503 30.975 -412.34
<none>                      30.724 -412.12
- SATV          1    0.4532 31.178 -410.91
- White         1    1.1778 31.902 -405.88
- HU            1    2.4567 33.181 -397.27
- HSGPA         1    6.5762 37.301 -371.64

Step:  AIC=-414.07
GPA ~ HSGPA + SATV + Male + HU + SS + FirstGen + White

           Df Sum of Sq    RSS     AIC
- FirstGen  1    0.1129 30.844 -415.27
- Male      1    0.1469 30.878 -415.03
- SS        1    0.2470 30.978 -414.32
<none>                  30.731 -414.07
- SATV      1    0.4677 31.199 -412.77
- White     1    1.1713 31.902 -407.88
- HU        1    2.4506 33.181 -399.27
- HSGPA     1    6.7560 37.487 -372.55

Step:  AIC=-415.27
GPA ~ HSGPA + SATV + Male + HU + SS + White

        Df Sum of Sq    RSS     AIC
- Male   1    0.1534 30.997 -416.18
- SS     1    0.2815 31.125 -415.28
<none>               30.844 -415.27
- SATV   1    0.5898 31.434 -413.12
- White  1    1.2934 32.137 -408.27
- HU     1    2.8154 33.659 -398.14
- HSGPA  1    6.6441 37.488 -374.55
```

```
Step:  AIC=-416.18
GPA ~ HSGPA + SATV + HU + SS + White

        Df Sum of Sq    RSS     AIC
<none>                30.997 -416.18
- SS     1    0.2951 31.292 -416.11
- SATV   1    0.7005 31.698 -413.29
- White  1    1.3133 32.310 -409.10
- HU     1    2.7987 33.796 -399.25
- HSGPA  1    6.4968 37.494 -376.51

Call:
lm(formula = GPA ~ HSGPA + SATV + HU + SS + White, data = GPA.df)

Coefficients:
(Intercept)        HSGPA         SATV           HU           SS        White
  0.5684876    0.4739983    0.0007481    0.0167447    0.0077474    0.2060408
```

If we want to conduct backward elimination but with C_p rather than AIC as the criterion used to decide which predictor to drop at each step, then we need to calculate MSE and include it in the `step()` command.

```
> MSE <- (summary(full)$sigma^2)
> step(full, scale=MSE, direction="backward")
```

Stepwise Regression

To use stepwise regression, we fit the model with no predictors (just a constant). Then we tell the `step()` command to start with that model (called "none" here) and to look for models up to and including the full model.

```
> full <- lm(GPA ~ ., data=FirstYearGPA)
> none <- lm(GPA ~ 1, data=FirstYearGPA)
> MSE <- (summary(full)$sigma^2)
> step(none, scope=formula(full), scale=MSE)
```

This produces

```
Start:  AIC=104.4
GPA ~ 1

               Df Sum of Sq  RSS    Cp
+ HSGPA         1      9.43 37.8  42.2
+ HU            1      4.68 42.6  74.5
+ SATV          1      4.37 42.9  76.6
+ White         1      3.75 43.5  80.8
+ SATM          1      1.78 45.4  94.2
+ FirstGen      1      1.16 46.1  98.5
<none>                      47.2 104.4
+ CollegeBound  1      0.19 47.0 105.1
+ Male          1      0.13 47.1 105.5
+ SS            1      0.00 47.2 106.3

Step:  AIC=42.18
GPA ~ HSGPA

               Df Sum of Sq  RSS    Cp
+ HU            1      3.31 34.5  21.7
+ White         1      3.23 34.6  22.2
+ SATV          1      2.19 35.6  29.3
+ FirstGen      1      1.63 36.2  33.1
+ SATM          1      0.77 37.0  39.0
+ Male          1      0.41 37.4  41.4
<none>                      37.8  42.2
+ CollegeBound  1      0.03 37.8  44.0
+ SS            1      0.00 37.8  44.2
- HSGPA         1      9.43 47.2 104.4

Step:  AIC=21.68
GPA ~ HSGPA + HU

               Df Sum of Sq  RSS    Cp
+ White         1      2.52 32.0  6.53
+ SATV          1      1.80 32.7 11.41
+ SATM          1      0.86 33.6 17.83
+ FirstGen      1      0.80 33.7 18.24
+ Male          1      0.43 34.1 20.73
+ SS            1      0.38 34.1 21.10
<none>                      34.5 21.68
+ CollegeBound  1      0.04 34.5 23.42
```

```
- HU             1      3.31 37.8 42.18
- HSGPA          1      8.06 42.6 74.54

Step:  AIC=6.53
GPA ~ HSGPA + HU + White

               Df Sum of Sq  RSS    Cp
+ SATV          1      0.68 31.3  3.90
+ FirstGen      1      0.31 31.7  6.43
<none>                      32.0  6.53
+ SATM          1      0.28 31.7  6.61
+ Male          1      0.28 31.7  6.63
+ SS            1      0.28 31.7  6.66
+ CollegeBound  1      0.05 31.9  8.20
- White         1      2.52 34.5 21.68
- HU            1      2.60 34.6 22.21
- HSGPA         1      7.77 39.7 57.39

Step:  AIC=3.9
GPA ~ HSGPA + HU + White + SATV

               Df Sum of Sq  RSS    Cp
+ SS            1      0.30 31.0  3.89
<none>                      31.3  3.90
+ Male          1      0.17 31.1  4.76
+ FirstGen      1      0.16 31.1  4.84
+ SATM          1      0.03 31.3  5.72
+ CollegeBound  1      0.01 31.3  5.81
- SATV          1      0.68 32.0  6.53
- White         1      1.40 32.7 11.41
- HU            1      2.50 33.8 18.94
- HSGPA         1      6.56 37.9 46.53

Step:  AIC=3.89
GPA ~ HSGPA + HU + White + SATV + SS

               Df Sum of Sq  RSS    Cp
<none>                      31.0  3.89
- SS            1      0.30 31.3  3.90
+ Male          1      0.15 30.8  4.85
+ FirstGen      1      0.12 30.9  5.08
+ SATM          1      0.05 30.9  5.52
+ CollegeBound  1      0.02 31.0  5.76
- SATV          1      0.70 31.7  6.66
- White         1      1.31 32.3 10.83
- HU            1      2.80 33.8 20.93
- HSGPA         1      6.50 37.5 46.09

Call:
lm(formula = GPA ~ HSGPA + HU + White + SATV + SS, data = FirstYearGPA)

Coefficients:
(Intercept)        HSGPA           HU        White         SATV           SS
   0.568488     0.473998     0.016745     0.206041     0.000748     0.007747
```

4.3 Cross-validation

Example 4.3: Houses in NY: Cross-validation
In the text we use cross-validation to check on the regression of `Price` on `Size` for the `HousesNY` data, using the first 35 observations as the training sample and the last 18 observations as the testing sample.

```
> data(HousesNY)
> train <- HousesNY[1:35,]  # select cases 1 - 35
> holdout <- HousesNY[36:53,]  # select cases 36 - 53
```

We fit the model using the training sample:

```
> modelTrain <- lm(Price ~ Size, data=train)
```

We can use `modelTrain` to make predictions for each house in the holdout sample, then we compute residuals for each of those houses:

```
> holdout$PriceHat <- predict(modelTrain, holdout)
> holdout$Residuals <- holdout$Price - holdout$PriceHat
```

We hope that these holdout sample residuals are about the same size, on average, as the residuals from the training sample. We can calculate the MSE using

```
> SSE <- sum(holdout$Residuals^2)
> MSEcv <- SSE/18 # we divide by 18 because there are 18 houses in the holdout sample
```

In this case the cross-validation MSE, which we call $MSEcv$, is 1432; this compares to 1217 as the MSE from the training model, found with

```
> anova(modelTrain)
```

In this example we used the first 35 observations as the training sample. Rather than use the first 35 houses, we could get a random selection of 35 observations (which is 2/3 of the number of houses in the data frame) with the first line of code below. The rest of the code goes through the same steps as in the previous example.

```
> sample <- sample.int(n = nrow(HousesNY), size = floor(0.67*nrow(HousesNY)),
            replace = FALSE)
> train <- HousesNY[sample,]
> holdout <- HousesNY[-sample,]
> modelTrain <- lm(Price ~ Size, data=train)
> anova(modelTrain)
> holdout$PriceHat <- predict(modelTrain, holdout)
> holdout$Residuals <- holdout$Price - holdout$PriceHat
> SSE <- sum(holdout$Residuals^2)
> MSEcv <- SSE/length(holdout$Residuals)
> MSEcv
```

The following four lines of code compute the cross-validation correlation and the shrinkage.

```
> crossR <- cor(holdout$Price,holdout$PriceHat)
> crossRsq <- crossR^2
> shrinkage <- summary(modelTrain)$r.squared-crossRsq
> c(crossR,crossRsq,shrinkage)
```

Of course, if we wanted to change from using 2/3 of the data in the training sample (and 1/3 in the holdout sample), we could just change 0.67 in the first line of code to some other number, such as 0.50.

4.4 Identifying Unusual Points in Regression

Example 4.7: More Butterfly Ballots

The textbook introduces these quantities for identifying unusual points in a regression analysis:

- Leverage
- Standardized and studentized residuals
- Cook's distance

All of these quantities are available through the `ls.diag` function, the output of which can be used to identify *moderately unusual* or *very unusual* points according to the summary table in the text, which we reproduce below. Recall that k is the number of predictors and n is the number of cases.

Statistic	Moderately unusual	Very unusual
Leverage, h_i	above $2(k+1)/n$	above $3(k+1)/n$
Standardized residual	beyond ± 2	beyond ± 3
Studentized residual	beyond ± 2	beyond ± 3
Cook's D	above 0.5	above 1.0

For this example, we return to the `PalmBeach` data frame and form the model that regresses `Buchanan` on `Bush`; we name the model `modBush`. We obtain a collection of regression diagnostics using the code below. The result of the call to `ls.diag` is a list of eight objects that includes the hat matrix entries (i.e., the h_i values), standardized residuals, studentized residuals, and Cook's distance, labeled, respectively, `hat`, `std.res`, `stud.res`, and `cooks` in the output. (*Note:* You can get a fuller explanation of all components of the list by typing `help(ls.diag)`.)

```
> modBush <- lm(Buchanan ~ Bush, data=PalmBeach)
> modBush.diag <- ls.diag(modBush)
> summary(modBush.diag)
             Length Class  Mode
std.dev       1     -none- numeric
hat          67     -none- numeric
std.res      67     -none- numeric
stud.res     67     -none- numeric
cooks        67     -none- numeric
dfits        67     -none- numeric
correlation   4     -none- numeric
std.err       2     -none- numeric
cov.scaled    4     -none- numeric
cov.unscaled  4     -none- numeric
```

To replicate the textbook's output that identifies the counties that exceed the threshold of $h_i > 6/n$, you can type in this code:

```
> PalmBeach[modBush.diag$hat > 6/67,] # We want all columns that satisfy
                                      # the row condition, thus we use the
                                      # wild card convention of blank space
                                      # after the comma.


        County Buchanan   Bush
6      BROWARD      789 177279
13        DADE      561 289456
29 HILLSBOROUGH     836 176967
52    PINELLAS     1010 184312
```

To obtain those cases that exceed either the $h_i > 6/n$ threshold or the threshold of the standardized residuals exceeding ± 2, we can use the following code. Notice the use of the | symbol to denote the logical "or."

```
# in the first two lines, we take the set of all subscripts---1, 2, ..., 67---
# and select out only those subscripts we want to look at.  The 67 subscripts
# correspond to the 67 Florida counties.
> id <- (1:67)[modBush.diag$hat > 6/67 | abs(modBush.diag$std.res) > 2]
> id
[1]  6 13 29 50 52

> results <- cbind(PalmBeach[id,],modBush$fit[id],
+ modBush$resid[id],modBush.diag$std.res[id])
> names(results) <- c("County","Buchanan","Bush","Fits",
+ "Resids","Stand. Resids")
> results
         County Buchanan   Bush      Fits     Resids Stand. Resids
6       BROWARD      789 177279  916.9402 -127.94023    -0.3807500
13         DADE      561 289456 1468.4953 -907.49526    -3.0591800
29 HILLSBOROUGH      836 176967  915.4062  -79.40618    -0.2362617
50   PALM BEACH     3407 152846  796.8074 2610.19263     7.6510719
52     PINELLAS     1010 184312  951.5203   58.47972     0.1749128
```

4.5 Coding Categorical Predictors

The file **ThreeCars2017** includes indicator variables `Accord`, `Maxima`, and `Mazda6` already created. If this were not true, we would use the R command

```
> ThreeCars2017$Accord <- as.numeric(ThreeCars2017$CarType=="Accord")
```

to create the indicator variable for Accord and likewise for Maxima and Mazda6.

If we wanted to produce the first of the predictions and intervals found in the text, we would first fit the regression model with the command

```
> modMilesType <- lm(Price~Mileage+Accord+Mazda6+Maxima, data=ThreeCars2017)
```

Then, we get the confidence interval with

```
> mod_eval(modMilesType, Mileage=50, Accord=1, Mazda6=0, Maxima=0, interval="confidence")
```

To get the prediction interval we use

```
> mod_eval(modMilesType, Mileage=50, Accord=1, Mazda6=0, Maxima=0, interval="prediction")
```

4.6 Randomization Test for a Relationship

Example 4.11: Predicting GPAs with SAT Scores

Here is an R script for carrying out the randomization test (also called a permutation test) of the null hypothesis that the population correlation is zero.

```
> data(SATGPA) #get the data!
> with(SATGPA, cor(GPA,VerbalSAT)) #should produce .2444543 as the sample r
> y <- SATGPA$GPA
> y <- SATGPA$VerbalSAT
> originalr <- cor(x,y)
> NEWy <- sample(SATGPA$GPA) #permutes the 24 GPA values
> cor(x,NEWy) #get the permutation cor value
#We might want to repeat this a few times to see how the correlations vary
#now let's do this many times
> N <- 1000 #set the number of simulation runs to 1000
> permcorr <- as.numeric(0) #create a place to store results
> for (i in 1:N){
   NEWy <- sample(SATGPA$GPA)
   permcorr[i] <- cor(x,NEWy)
   }

> permcorr[1:5] #look at the first 5 results
> hist(permcorr) #make a histogram of the results
> upper <- sum(permcorr>abs(originalr)) #count those results > 0.2444543
> lower<- sum(permcorr<(-abs(originalr))) #count the lower tail results
#N.B.: "permcorr<-abs(originalr)" would _assign_ .2444543 to permcorr
# so we have #to be careful and use parentheses around abs(originalr)!
> pvalue <- (upper+lower)/N
> pvalue #see the simulation P-value
```

The line `hist(permcorr)` produces something similar to Figure 4.16 but without the shading of the two areas.

In the R code above, we count the number of times that the permutation correlation is greater than the observed 0.244, and we make a separate count of the number of permutation correlations smaller than -0.244. This approach to conducting a two-sided test works well when the distribution of permutation results is reasonably symmetric, as can be seen in Figure 4.16.

An alternative approach that works even when the distribution is skewed is to count only the cases in the upper tail (when the observed test statistic is positive, as it is here) and then double the result to get the P-value for the nondirectional test. The R script below illustrates this.

```
> upper <- sum(permcorr>originalr) #count those results > 0.2444543
> pvalue <- upper*2/N
> pvalue #see the simulation P-value
```

4.7 Bootstrap for Regression

Example 4.12: Accord Prices

The Rmd file for Section 4.7 shows how to use the `do()` command from the `mosaic` package. Here we present a different R script for collecting bootstrap samples.

```
> data(AccordPrice) #get the data!
> originalmodel <- lm(Price~Mileage, data=AccordPrice)
> summary(originalmodel) #Look at the fitted model

> car <- 1:30 #create indices 1,2,...,30
> bootsample <- sample(car,replace=TRUE) #create a bootstrap sample
> head(AccordPrice[bootsample,]) #look at the first few boot cases

> bootmodel <- lm(Price~Mileage,data=AccordPrice[bootsample,])
#fit the model #using the bootstrap sample
> summary(bootmodel) #look at the fitted model from the bootstrap sample

#Now do this many times
> bootbetas <- matrix(0,nrow=5000, ncol=2) #set up a matrix to hold results

> for (i in 1:5000) {   #the main bootstrap loop
       bootsample <- sample(car,replace=TRUE)
       bootmodel <- lm(Price~Mileage,
                 data <- AccordPrice[bootsample,])
       bootbetas[i,] <- coef(bootmodel)
   }

> hist(bootbetas[,2]) #Make a histogram of the slopes (the second coefficient)
```

To construct a confidence interval using Method #2, we need the SD of the bootstrap slopes, found with the command

```
> sd(bootbetas[,2])
```

To construct a confidence interval using Method #1, we need quantiles of the bootstrap distribution. For this, we use the commands

```
> quantile(bootbetas[,2],.025)
> quantile(bootbetas[,2],.975)
```