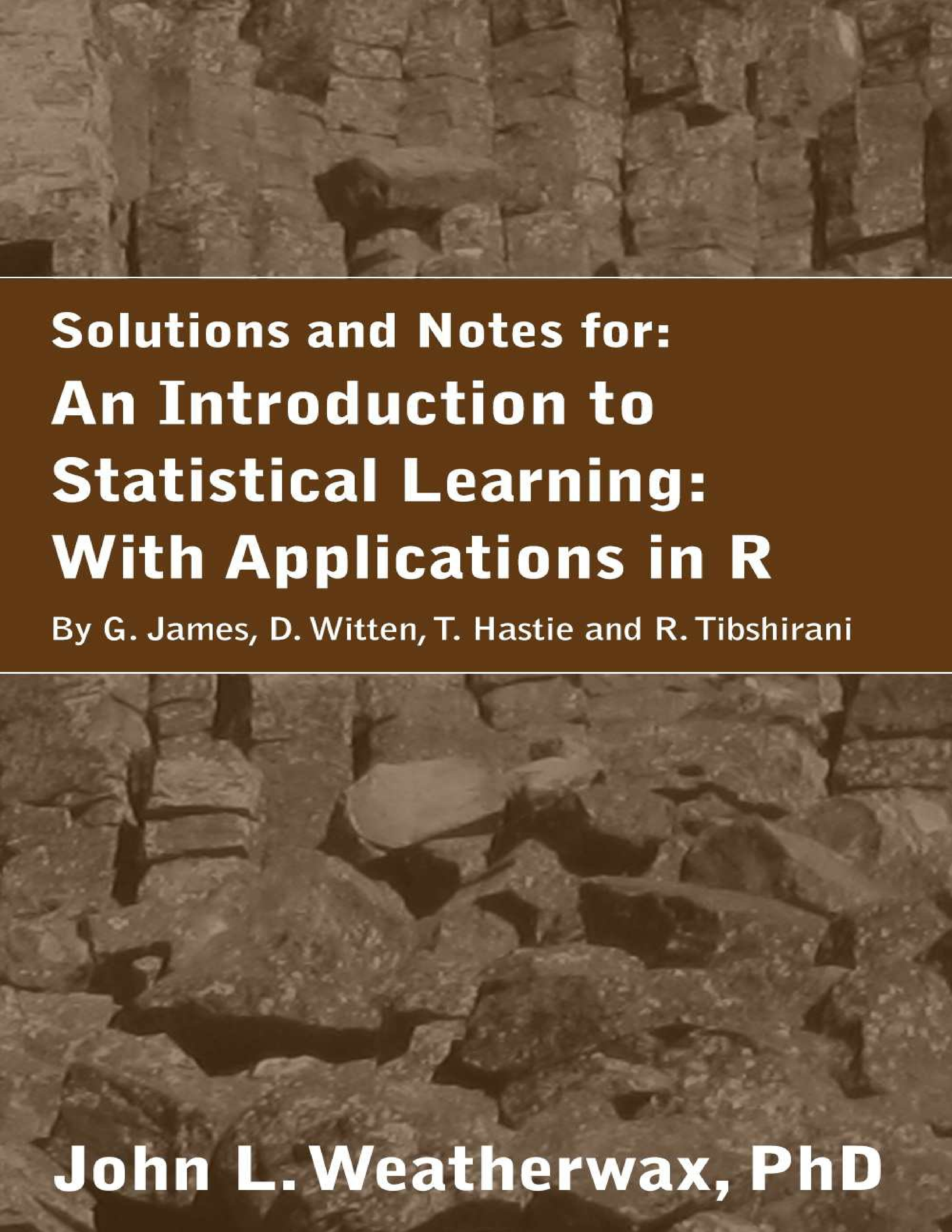

Solutions and Notes for: An Introduction to Statistical Learning: With Applications in R

By G. James, D. Witten, T. Hastie and R. Tibshirani

John L. Weatherwax, PhD

A Solution Manual and Notes for: An Introduction to Statistical Learning (ISLR): with Applications in R by G. James, D. Witten, T. Hastie, and R. Tibshirani

John L. Weatherwax[*]

August 10, 2014

[*] *wax@alum.mit.edu*

To Sophia, who will never know how unbelievably special she is.

Introduction

This document has notes and solutions to the end of chapter problems from the book[1] :

An Introduction to Statistical Learning: with Applications in R
by Gareth James, Daniela Witten, Trevor Hastie, & Robert Tibshirani

This book is somewhat like an earlier book[2] :

The Elements of Statistical Learning: Data Mining, Inference, and Prediction
by Trevor Hastie, Robert Tibshirani, & Jerome Friedman

In that it discusses a number of modern methods for statistical learning. If there was any drawback to the earlier book it was that the examples were presented without algorithmic code to see how to duplicate them. At various locations on the web, people were able to reverse engineer the figures from the book and write code that duplicated many of the results. This might not have necessary since much much of the code for doing statistical machine learning has been standardized in R packages that make performing any given analysis easy to do. In ISL after each topic is introduced there are R labs that demonstrate how to use these libraries to implement the techniques discussed. The text shows actual R input and output on provided data sets. It is then easy to modify these labs to perform analysis on any data source that might be of interest, thus no tedious reverse engineering is needed!

In addition, at the end of each chapter are “conceptual exercises” and “applied exercises”. In the conceptual exercises the reader is asked to reason about the techniques discussed in the chapter to ensure understanding of the presentation. In the applied exercises section the reader is asked to perform a more hands-on analysis using R and a number of provided data sets. This simulates the process one goes through when trying machine learning techniques on novel data sets. In this way, one gets a very applied view of machine learning: what to look for in a data set, how to apply various techniques, how to assess your algorithms performance, how to interpret your results, and what steps to take next.

To make sure I understood this material as well as possible, I worked *all* the conceptual and applied exercises at the end of each chapter. This document is the result of that effort. It is my hope that students of machine learning and statistics will find this material useful. The code snippets in R for the applied exercises can be found at the following location:

http://waxworksmath.com/Authors/G_M/James/james.html

As a final comment, I’ve worked hard to make these notes as good as I can, but I have no illusions that they are perfect. If you feel that that there is a better way to accomplish or explain an exercise or derivation presented in these notes; or that one or more of the explanations is unclear, incomplete, or misleading, please tell me. If you find an error of any kind – technical, grammatical, typographical, whatever – please tell me that, too. I’ll gladly add to the acknowledgments in later printings the name of the first person to bring each problem to my attention.

Acknowledgments

Special thanks to (most recent comments are listed first): to Greg Sternberg and Zachary Booker for their corrections.

Contents

Chapter 2 (Statistical Learning)

Conceptual Exercises

Exercise 1

Part (a): We expect that with a very large number of measurements *n*, a flexible learning method would be able to learn the signal without as much fear of overfitting.

Part (b): If the number of predictors *p* is very large and *n* small then there is a greater possibility that a flexible learning method would overfit. The we expect the inflexible method to be better in this case.

Part (c): A highly non-linear relationship would most likely need a flexible statistical learning method to perform optimally.

Part (d): With a very large error term variance σ^2 there is more worry about overfitting and thus an inflexible method would perform better.

Exercise 2

Part (a): Since CEO salary is a real valued function we would be interested in regression. Since we just want to know *which* factors affect salary we have a inference problem. We have *n* = 500 and *p* = 3.

Part (b): This would be a classification problem where we want accurate predictions. We have *n* = 20 and *p* = 3 + 10 = 13.

Part (c): This is a regression problem where we are interested in accurate predictions. We have *n* = 52 (number of weeks in a year) and *p* = 4.

Exercise 3

Recall the expression that expresses the mean square error in terms of "the bias-variance" decomposition

$$E\left[(y_0 - \hat{f}(x_0))^2\right] = \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Bias}(\hat{f}(x_0))^2 + \mathrm{Var}(\epsilon). \qquad (1)$$

The left-hand-side of the above is the expected mean square error (MSE) or a measure of how well on average our approximation function $\hat{f}$ at x_0 is estimating the true value y_0.

The first term on the right-hand-side of the above expression is the error in the MSE due to errors in “fitting” the true *f* with our approximate $\hat{f}$. This error come from the sensitivity of the learning procedure to the finite training data set. Typically more flexible fitting methods will be more sensitive to the errors and noise in the given training set. The second term on the right-hand-side is the error in $f \neq \hat{f}$ due to using a learning algorithm that might not be able to represent the complexities in *f*. For example, taking $\hat{f}$ to be linear when the true underlying function *f* is non-linear. The third term on the right-hand-side represents un-learnable error due to either not having all the predictive variables in our model (predictors that if we could get values for would improve our ability to learn the function *f*) or error that is just intrinsic to the process which we are trying to model. In either case, given the data we have there is no way to reduce this component of the MSE error.

A typical plots of the things suggested look like pieces from Figures 2.9 and Figure 2.12 from the book. In Figure 2.9 (right-hand-side) we have plots of the training error, testing error and the Bayes irreducible error curves. The training error shows a steady decrease (improvement) as the flexibility of the learning method increases. The test error is the red curve that initially decreases as the flexibility increase but then begins to increase again for continued increase in flexibility. The Bayes irreducible error is the constant dotted line. Notice that the point where the testing error is as close to the Bayes error would be the optimal operating point for this system. The distance between the lowest point on the testing error curve and the Bayes error gives an indication of how much bias there is in the given learning procedure, i.e. how far the best function $\hat{f}$ will be from *f*.

In Figure 2.12 (left plot) we have curves representing the individual components of the bias-variance decomposition. The blue curve is the squared bias which we see gets smaller as the complexity of the model increase (we are able to model more and more complicated patters in *f*). The orange curve shows the variance of the learned model $\hat{f}$ i.e. as we add complexity the dependence on the dataset as to exactly what function $\hat{f}$ we get out of our learning procedure gets more sensitive (and the error increase). The horizontal line is the irreducible Bayes error again.

Exercise 4

Part (a): Classification might be helpful in determining whether or not something is present given measurements that indicate how likely it might be. The classic example is the presence of a disease like cancer. In that case we would perform very different actions depending on the outcome. For classification a binary response $Y \in \{0, 1\}$ representing True or False might be the appropriate output. The goal would be prediction.

Part (b): One case where regression could be used would be to predict the amount a stock index might move after a specific economic event say the release of the oil inventory numbers. In that case we are interested in the return (change in price) the market index realizes given the values of the oil inventory number and possibly other macroeconomic variables like gross domestic product, interest rates, trade deficits, etc.

Part (c): Cluster analysis might be useful when we don't know a-priori which of a set of samples will behave in a similar way. We can cluster them in feature space and then assume that their response or outcomes in another situation will be similar. Cluster analysis can be used with gene response data to cluster certain cancers into "types" depending on the genes their tissues express.

Exercise 5

A very flexible fitting procedure will fit non-linear functions better (if that is indeed the model generation process that is generating your data) but will be more susceptible to errors/noise in the training dataset. A less flexible approach exchanges where it makes errors. That is a less flexible fitting procedure is unable to model the exact non-linear *f* but its predictions are also likely to be more stable to errors/noise in the training dataset.

Exercise 6

A parametric learning procedure means the functional form of the mapping $\hat{f}$ is specified, except for the parameter *values*, which the learning procedure must estimate. A non-parametric learning procedure is much more flexible in the forms of *f* it can model and the learning procedure must "learn more" from the data (the functional form that $\hat{f}$ should take) and then the parameters needed to estimate it. A parametric approach is generally a less flexible fitting method while a non-parametric approach is a more flexible method with the trade-offs that that characterization contains.

Exercise 7

Part (a): We can perform this calculation with the simple R code

```
data = matrix( data=c(0,3,0,2,0,0,0,1,3,0,1,2,-1,0,1,1,1,1), nrow=6, ncol=3, byrow=T )
sqrt( colSums( t(data)^2 ) ) # the L2 distance from (0,0,0)
[1] 3.000000 2.000000 3.162278 2.236068 1.414214 1.732051
```

Part (b): With *K* = 1 we select the classification of the closest sample which is the 5th sample and we would predict Green.

Part (c): With *K* = 3 we would consider the vote of the three closest samples which would be the second, fifth, and sixths samples. These have labels of Red, Green, and Red respectively. A majority vote of these three gives the classification of Red.

Part (d): A large value of *K* implies lots of voting from the many data points included in the neighborhood and a smoother decision boundary. A small value of *K* provides a decision boundary that can vary quite a bit from point to point. Based on this, a highly non-linear decision boundary would be fit better with a *smaller* value of *K*.

Applied Exercises

Exercise 8

See the R code chap_2_prob_8.R where this problem is worked. Much of this problem is exploring the College data set. Some interesting observations might involve looking at which university has the most students in the top 10% of the class (MIT). What university has the smallest acceptance rate (Princeton) and what university has the largest acceptance rate (Emporia State University).

Exercise 9

See the R code chap_2_prob_9.R where this problem is worked.

Part (a): Using the summary command on the Auto dataset we easily see which predictors are quantitative: mpg, displacement, horsepower, weight, acceleration and year. While the variables cylinders (values are 3, 4, 5, 6, or 8), origin (values are 1, 2, and 3), and name are qualitative.

Part (b): Using range on the columns that are numeric we compute

```
> sapply( Auto[,-qualitative_columns], range )
      mpg displacement horsepower weight acceleration year
[1,]  9.0           68         46   1613          8.0   70
[2,] 46.6          455        230   5140         24.8   82
```

Here we use sapply for “simple apply” to map a function (here range) to each column of the Auto dataframe that is a quantitative variable.

Part (c): These would be given by

```
> sapply( Auto[,-qualitative_columns], mean )
         mpg displacement   horsepower       weight acceleration         year
    23.44592    194.41199    104.46939   2977.58418     15.54133     75.97959
> sapply( Auto[,-qualitative_columns], sd )
         mpg displacement   horsepower       weight acceleration         year
    7.805007   104.644004    38.491160   849.402560     2.758864     3.683737
```

Part (d): For the mean this could be done as follows

```
> sapply( Auto[-seq(10,85),-qualitative_columns], mean )
       mpg displacement   horsepower       weight acceleration        year
   24.40443    187.24051    100.72152   2935.97152     15.72690    77.14557
```

The other functions could be done in the same manner.

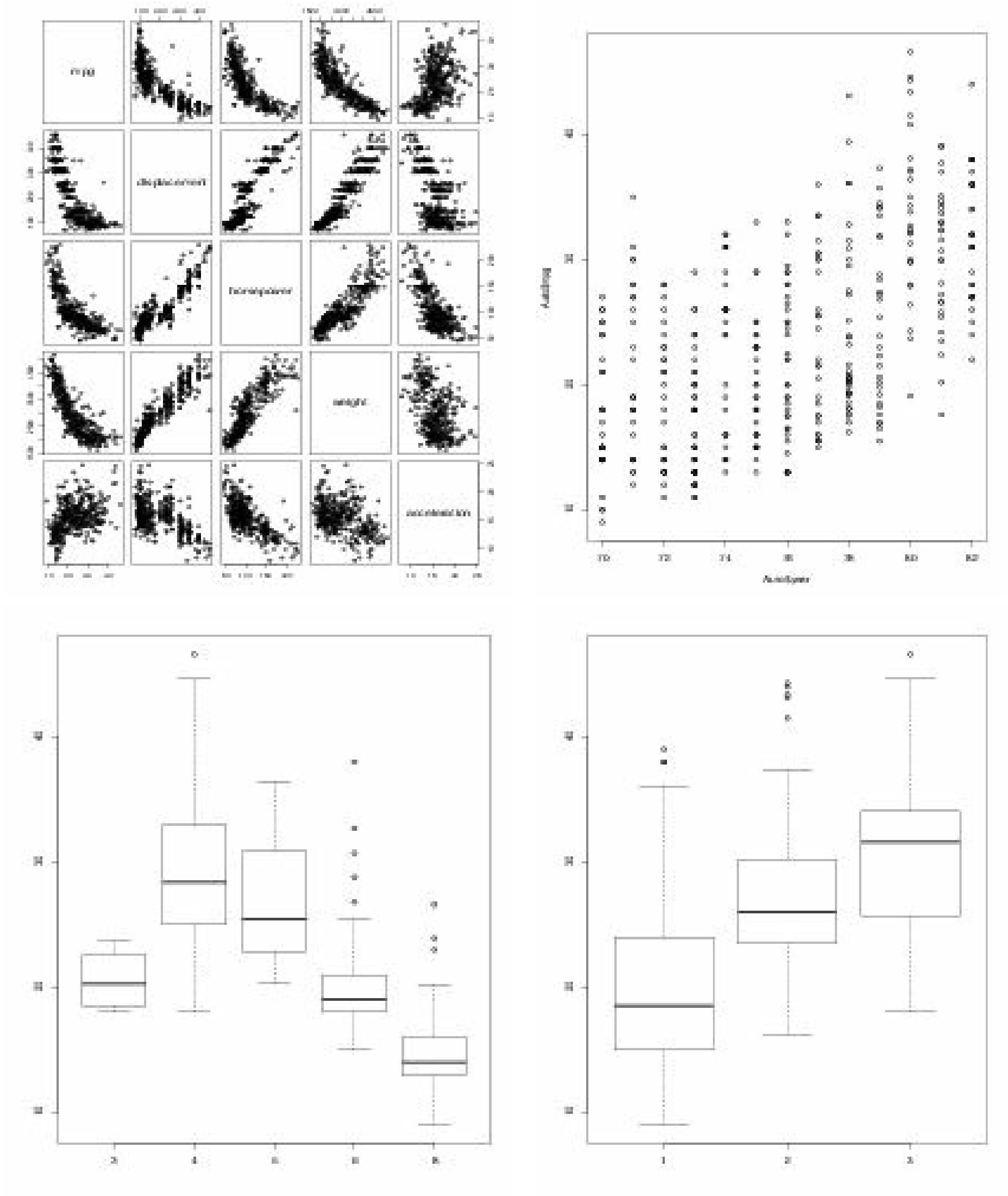

Figure 1: Upper Left: Scatter plots of the quantitative variables in the Auto dataset.

Notice that there seems to be a non-linear relationship between mpg (in the first row) and many of the other variables. Also notice that many of the variables seem quite correlated. **Upper Right:** A scatter plot of mpg as a function of year. Notice that as year has increased mpg has also increased. **Lower Left:** A boxplot of mpg as a function of cylinders. Notice that certain cylinder sizes (for example 4) seem to give vehicles with a larger value of mpg. **Lower Right:** A boxplot of mpg as a function of origin. We see very different values for mpg depending on the value of origin. This indicates that this is perhaps a strong predictor of mpg.

Part (e): See Figure 1 (and the corresponding caption) for plots of some of relationships between the variables in this dataset.

Part (f): The plots in Part (e) indicate that there are several variables that could be predictive of mpg.

Exercise 10

See the R code chap_2_prob_10.R where this problem is worked.

Part (a): The Boston dataset has 506 rows and 14 columns. The rows represent regions in the Boston area and the columns represent data on variables that might influence home prices in the various region.

Part (b): There are so many variables in this dataset it seems worthwhile to produce a scatter plot of only the variables that would most likely influence the variable medv (the median home value in each region). One way to do this would be to select the variables to plot “by hand”. Another (more automated way) would be to look at the correlation of each variable with the variable that we hope to predict or medv and then consider the ones that are the most correlated with medv. To do this we first compute the correlations to find

```
> all_correlations = cor( Boston )
> print( all_correlations[,14] ) # correlations with "medv"
      crim         zn      indus       chas        nox         rm        age
-0.3883046  0.3604453 -0.4837252  0.1752602 -0.4273208  0.6953599 -0.3769546
       dis        rad        tax    ptratio      black      lstat       medv
 0.2499287 -0.3816262 -0.4685359 -0.5077867  0.3334608 -0.7376627  1.0000000
```

Sorting these we get

```
> sort( all_correlations[,14] )
     lstat    ptratio      indus        tax        nox       crim        rad
-0.7376627 -0.5077867 -0.4837252 -0.4685359 -0.4273208 -0.3883046 -0.3816262
       age       chas        dis      black         zn         rm       medv
```

```
-0.3769546  0.1752602  0.2499287  0.3334608  0.3604453  0.6953599  1.0000000
```

The variables with the largest correlation with medv then are lstat, rm, ptratio, indus, and tax. Plotting a scatter plot of these variables, with medv in the top row we get the plot given in Figure 2. From that plot we see that several variables seem very predictive of medv.

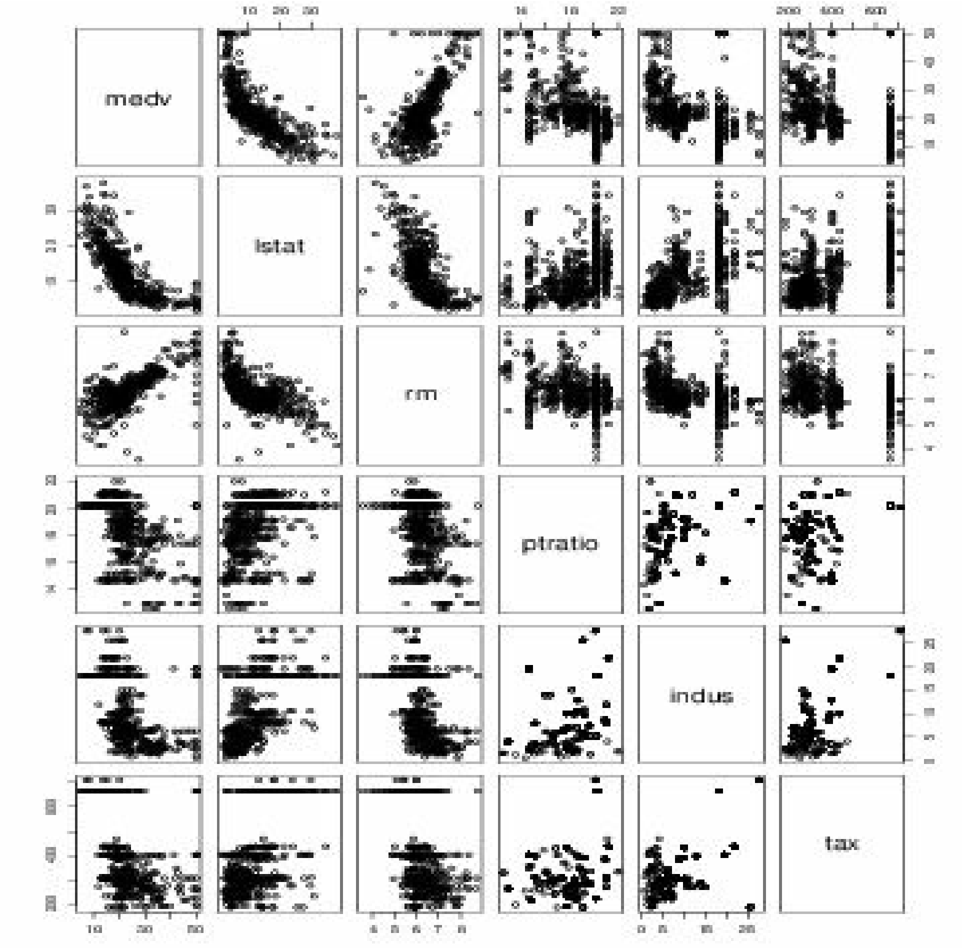

Figure 2: A scatter plot of medv and other variables suspected of being good at predicting medv. See the text for details.

Part (c): From the correlation matrix we see that rad (index of accessibility to radial highways) is the most correlated variable with crim.

Part (e): We find that 35 of the suburbs from 506 are bounded by the Charles river.

Part (f): We compute

```
> median(Boston$ptratio)
[1] 19.05
```

Part (g): We compute the suburb with the smallest home price and then the ranges of the predictors using the following code

```
print( which.min(Boston$medv) )
[1] 399
Boston[ which.min(Boston$medv), ]
       crim zn indus chas   nox    rm age    dis rad tax ptratio black lstat
399 38.3518  0  18.1    0 0.693 5.453 100 1.4896  24 666    20.2 396.9 30.59
    medv
399    5
sapply( Boston, range )
         crim  zn indus chas   nox    rm   age     dis rad tax ptratio  black
[1,]  0.00632   0  0.46    0 0.385 3.561   2.9  1.1296   1 187    12.6   0.32
[2,] 88.97620 100 27.74    1 0.871 8.780 100.0 12.1265  24 711    22.0 396.90
     lstat medv
[1,]  1.73    5
[2,] 37.97   50
```

One notices that the suburb with the smallest home price has values of its feature vector at the extream end of many of the features. This indicates that its attributes are perhaps the ”worst” in the feature.

Part (h): We can look at the records that have more than eight rooms per dwelling with the command Boston[Boston$rm > 8,]. We notice that these dwelling/suburbs have median home prices that are much greater than the average home price over all suburbs

```
print( range( Boston$medv ) )
[1]  5 50
> print( mean( Boston$medv ) )
[1] 22.53281
> print( mean( Boston[ Boston$rm > 8, ]$medv ) )
[1] 44.2
```

Chapter 3 (Linear Regression)

Conceptual Exercises

Exercise 1

Each of the *t*-values in the output of a linear regression corresponds to the null hypothesis $H_0 : \beta_i = 0$ when the other predictors are included in the model. A large *t*-value indicates that we can reject the null hypothesis with more confidence. A small *t*-value means we cannot reject the null hypothesis with confidence. From the table given in the text we would conclude that newspaper is not significant i.e. we have $\beta_{\text{newspaper}} = 0$ when the other predictors TV and radio are included in the model.

Exercise 2

Each technique starts by selecting the *k* closest samples in a neighborhood of our point *x*. For KNN classification we compute the majority vote of these nearby samples while for KNN regression we average the response *y* of these samples.

Exercise 3

Given the problem statement where *Y* is the staring salary after graduation (in thousands of dollars) then our model estimates that

$$Y = 50 + 20\text{GPA} + 0.07\text{IQ} + 35\text{Gender} + 0.01\text{GPA} * \text{IQ} - 10\text{GPA} * \text{Gender}.$$

Part (a): When we fix IQ and GPA in the model above and then consider what the model would predict for males and females we get

$$Y_{\text{male}} = 50 + 20\text{GPA} + 0.07\text{IQ} + 0.01\text{GPA} * \text{IQ}$$

$$Y_{\text{female}} = 50 + 20\text{GPA} + 0.07\text{IQ} + 0.01\text{GPA} * \text{IQ} + 35 - 10\text{GPA} .$$

Thus the difference between the average female and the average male starting salary is 35 − 10GPA and who makes more will depend on the sign of that expression. From the given four choices the third or (iii) will be true since if the GPA is small then females earn more while if the GPA is large then they would earn less. For example, if we have GPA = 3.8 then

$$Y_{\text{female}} - Y_{\text{male}} = 35 - 38 = -3,$$

and males earn more.

Part (b): This would be given by

$$\hat{Y} = 50 + 20(4.0) + 0.07(110) + 35(1) + 0.01(4.0)(110) - 10(4.0)(1.0) = 137.1.$$

Part (c): False. The absolute size of the linear regression coefficients depends on the dynamic range of the predictors and the response. Here IQ is in units of “hundreds” and the response *Y* is in units of “tens”. Thus the magnitude of the interaction coefficient between these two features may well be “small” since the two input ranges are somewhat large. The measure of importance of a coefficient is not determined by the coefficients absolute magnitude but by looking at the *t*-statistic of the interaction term GPA : IQ. This is the coefficients absolute size divided by its standard error.

Exercise 4

Part (a): We would expect the RSS to be smaller for the cubic regression since that functional form has more degrees of freedom to fit the training data set with and thus it will be able to do a better job.

Part (b): Since the *true* model is linear we would expect that on the test set the linear model would perform better than the cubic model.

Part (c): On the training set the cubic model will always have a lower RSS measurement since the procedure used to fit the coefficients in the cubic model is explicitly minimizing the RSS and has more degrees of freedom (four vs. two) do use in doing so.

Part (d): On the test set which model performs better depends on how non-linear the true underlying model is and how much data we were given to fit the models using. If the true underlying model is highly non-linear (like using horsepower to predict mpg in the Auto dataset) then we would expect the cubic model to perform better than the linear model. If however we were given only very few data points *n* then due to errors in fitting the cubic model might perform worse on the testing set than the linear model. This is because with only a few point to fit on the cubic model will have more variance than the linear model and could result in a worse out-of-sample fit.

Exercise 5 (the fitted values for linear regression without an intercept)

From the given expression for $\hat{\beta}$ we have

$$\hat{y}_i = x_i \hat{\beta} = \frac{x_i \sum_{i'=1}^{n} x_{i'} y_{i'}}{\sum_{j=1}^{n} x_j^2} = \sum_{i'=1}^{n} \left(\frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2} \right) y_{i'},$$

Thus to write the above in the form suggested $\hat{y}_i = \sum_{i'=1}^{n} a_{i'} y_{i'}$ we need to take

$$a_{i'} = \frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}.$$

Exercise 6

The least squares line is $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$. For this line to pass though the point $(\bar{x},\bar{y})$ we can put this point into the above equation and see if it is satisfied. Taking $x = \bar{x}$ the right-hand-side of this line gives

$$\hat{\beta}_0 + \hat{\beta}_1 \bar{x} = (\bar{y} - \hat{\beta}_1 \bar{x}) + \hat{\beta}_1 \bar{x} = \bar{y},$$

using the expression for $\hat{\beta}_0$. This shows that the least squares line passes through the point $(\bar{x},\bar{y})$.

Exercise 7

We get for R^2 the following

$$\begin{aligned} R^2 &= 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \\ &= 1 - \frac{\sum_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2}{\sum_i y_i^2} = 1 - \frac{\sum_i (y_i - \hat{\beta}_1 x_i)^2}{\sum_i y_i^2}\,. \end{aligned} \qquad (2)$$

When we use the fact that $\bar{x} = \bar{y} = 0$ in evaluating $\hat{\beta}_0$. Expanding the quadratic in the numerator of the above fraction we get

$$(y_i - \hat{\beta}_1 x_i)^2 = y_i^2 - 2 x_i y_i \hat{\beta}_1 - \hat{\beta}_1^2 x_i^2 .$$

If we sum both sides of this expression we get

$$\sum_i y_i^2 - 2\hat{\beta}_1 \sum_i x_i y_i - \hat{\beta}_1^2 \sum_i x_i^2 .$$

Using the known expression for $\hat{\beta}_1$ we have that the above is equal to

$$\sum_i y_i^2 - 2\left(\frac{\sum_i x_i y_i}{\sum_i x_i^2}\right) \sum_i x_i y_i - \left(\frac{\sum_i x_i y_i}{\sum_i x_i^2}\right)^2 \sum_i x_i^2 = \sum_i y_i^2 - \frac{\left(\sum_i x_i y_i\right)^2}{\sum_i x_i^2} .$$

If we put this expression into Equation 2 we get

$$R^2 = \frac{\sum_i y_i^2 - \sum_i y_i^2 + \frac{\left(\sum_i x_i y_i\right)^2}{\sum_i x_i^2}}{\sum_i y_i^2} = \frac{\left(\sum_i x_i y_i\right)^2}{\sum_i x_i^2 \sum_i y_i^2} .$$

Notice that this is the expression for Cor$(X,Y)^2$ given by the books equation 3.18 when $\bar{x} = \bar{y} = 0$ as we were to show.

Applied Exercises

Exercise 8

See the R code chap_3_prob_8.R where this problem is worked.

Part (a): The summary command for this fit gives

```
> summary(fit1)

Call: lm(formula = mpg ~ horsepower, data = Auto)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 39.935861   0.717499   55.66   <2e-16 ***
horsepower  -0.157845   0.006446  -24.49   <2e-16 ***

Residual standard error: 4.906 on 390 degrees of freedom
Multiple R-squared: 0.6059,     Adjusted R-squared: 0.6049
F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

Part (i): The *p*-value for the full fit is quite small indicating a predictive relationship exists between mpg and horsepower.

Part (ii): I'll take as a measure of the strength of the relationship to be the value of the R^2 for the given regression. From the summary command for this fit we see that we have R^2 = 0.6059 which says that 60% of the variance has been explained using this predictor.

Part (iii): From the estimate of β_1 above we see that $\beta_1 < 0$ indicating that as horsepower increases mpg will decrease.

Part (iv): We can answer these questions using the R function predict. We have

```
predict( fit1, data.frame(horsepower=98), interval = "confidence" )
       fit      lwr      upr
1 24.46708 23.97308 24.96108

predict( fit1, data.frame(horsepower=98), interval = "prediction" )
       fit     lwr      upr
1 24.46708 14.8094 34.12476
```

Part (b): See Figure 3 for a plot of the dataset with a least squares linear fit.

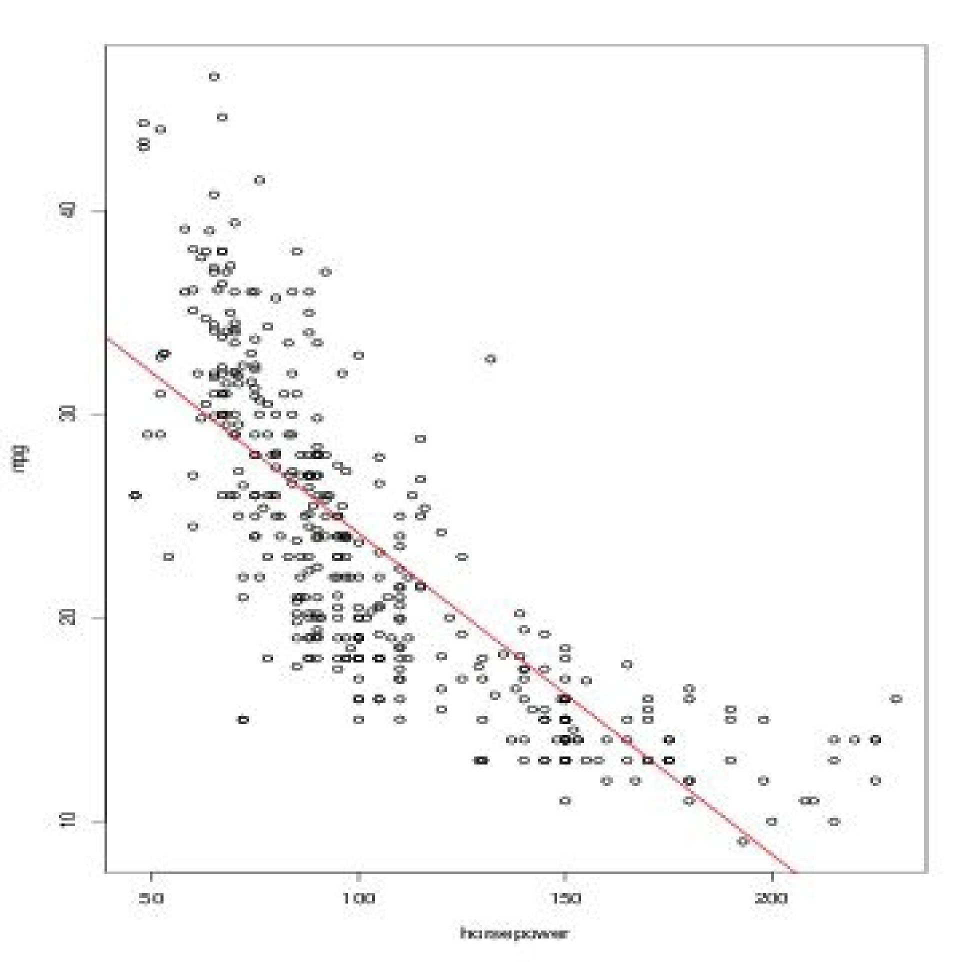

Figure 3: A linear fit of mpg as a function of horsepower.

Part (c): Calling plot on the output from a linear model gives four plots (that can be viewed sequentially by hitting the return key). The first is the residuals of the model as a function of the fitted values $\hat{y}$. There we see curvature in the scatterplot suggesting that a non-linear model might be needed. The second plot is a *qq*-plot which can be used to determine how well the residual are approximated by a normal curve. The plot for this data seems reasonable. The third plot can be used to determine if there is any horsepower dependence to the variance. This plot looks like there may be more variance for larger values of $\hat{y}$ that we could attempt to model using weighted least squares. The fourth plot expresses which points have large leverage (have the largest influence on the estimation of the least squares coefficient estimates). From that plot we see that the samples with indices 117 and 84 seem to have the largest influence on the estimations of β_0 and β_1.

Exercise 9

See the R code chap_3_prob_9.R where this problem is worked.

Part (c): When we do this we get

```
> summary(fit)

Call: lm(formula = mpg ~ ., data = Auto)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
cylinders     -0.493376   0.323282  -1.526  0.12780
displacement   0.019896   0.007515   2.647  0.00844 **
horsepower    -0.016951   0.013787  -1.230  0.21963
weight        -0.006474   0.000652  -9.929  < 2e-16 ***
acceleration   0.080576   0.098845   0.815  0.41548
year           0.750773   0.050973  14.729  < 2e-16 ***
origin         1.426141   0.278136   5.127 4.67e-07 ***

Residual standard error: 3.328 on 384 degrees of freedom
Multiple R-squared: 0.8215,     Adjusted R-squared: 0.8182
F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

Part (i): We see a large *F*-value and a very small *p*-value for the overall model fit. This indicates that there *is* a relationship between mpg and the other predictors.

Part (ii): From the above summary output the predictors with the most significance appear to be weight, year, and origin.

Part (iii): The coefficient of year is positive (with large significance) which indicates that as for larger years we have larger values of mpg. This implies perhaps that car manufactures worked harder as time went by to increase the mpg of the cars they produced.

Part (d): From the diagnostic plots produced by the command plot it looks like the points 323 and 327 have very large residuals. It looks like observation 14 is a point of high influence/leverage.

Part (e): We can use the R command update to add some interaction terms. Since all of the first order terms are included in the model we only need to add the interaction terms and thus the formula expression ":" is needed (the formula expression "*" adds both the first order terms *and* the interaction terms). To select the interaction terms to add we try to imagine which two terms might not act independently. Two such terms are horsepower and weight. The following abbreviated R command shows that this interaction term is important

```
> summary( update( fit, . ~ . + horsepower:weight ) )

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       2.876e+00  4.511e+00   0.638 0.524147
cylinders        -2.955e-02  2.881e-01  -0.103 0.918363
displacement      5.950e-03  6.750e-03   0.881 0.378610
horsepower       -2.313e-01  2.363e-02  -9.791  < 2e-16 ***
weight           -1.121e-02  7.285e-04 -15.393  < 2e-16 ***
acceleration     -9.019e-02  8.855e-02  -1.019 0.309081
year              7.695e-01  4.494e-02  17.124  < 2e-16 ***
origin            8.344e-01  2.513e-01   3.320 0.000986 ***
horsepower:weight  5.529e-05  5.227e-06  10.577  < 2e-16 ***

Residual standard error: 2.931 on 383 degrees of freedom
Multiple R-squared: 0.8618,     Adjusted R-squared: 0.859
F-statistic: 298.6 on 8 and 383 DF,  p-value: < 2.2e-16
```

Other combinations where the interaction is significant are acceleration:horsepower and acceleration:weight. Using the anova function shows that a model with the interaction terms is better (offers a statistically significant reduction in RSS) than the model without the interaction. From a pairs plot we would expect displacement, horsepower, weight, and acceleration to act somewhat the same manner with regard to interactions.

Part (f): Plots of the residuals vs. the fitted values indicate that a quadratic terms might be needed in modeling. Since we already has seen in some simpler cases (namely Figure 3.9) that a quadratic term in horsepower helps the regression. We can check that this coefficient is significant and that the resulting residuals vs. fitted values plot no longer show significant curvature. Incidentally, both square root and log transformations of horsepower also show significant non-zero coefficients when added. From a pairs plot we would expect displacement, horsepower, weight, and acceleration to act somewhat the same manner with regard to non-linearity.

Exercise 10

See the R code chap_3_prob_10.R where this problem is worked.

Part (a-b): A summary of the suggested fit is given by

```
Call: lm(formula = Sales ~ Price + Urban + US, data = Carseats)

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
Price       -0.054459   0.005242 -10.389  < 2e-16 ***
UrbanYes    -0.021916   0.271650  -0.081    0.936
USYes        1.200573   0.259042   4.635 4.86e-06 ***

Residual standard error: 2.472 on 396 degrees of freedom
Multiple R-squared: 0.2393,     Adjusted R-squared: 0.2335
F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

This indicates that the coefficient of Price is negative (and significant) showing that as the price increases sales decrease. The coefficient of UrbanYes is negative (but not significant). If it was significant we could conclude that an urban environment has less sales than a rural environment. The coefficient of USYes is positive indicating that stores in the US have increased sales over ones that are not located in the US.

Part (c): The model would be

$$\text{Sales} = \hat{\beta}_0 + \hat{\beta}_1 \text{Price} + \hat{\beta}_2 \text{UrbanYes} + \hat{\beta}_3 \text{USYes}.$$

Here if the value of Urban is Yes then the above indicator is one (otherwise it is zero) and if the value of US is Yes then the above indicator is one (otherwise it is zero).

Part (d): These are Price and USYes.

Part (e): This would give the model

```
summary( update( fit, . ~ . - Urban ) )

Call: lm(formula = Sales ~ Price + US, data = Carseats)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.03079    0.63098  20.652  < 2e-16 ***
Price       -0.05448    0.00523 -10.416  < 2e-16 ***
USYes        1.19964    0.25846   4.641 4.71e-06 ***

Residual standard error: 2.469 on 397 degrees of freedom
Multiple R-squared: 0.2393,     Adjusted R-squared: 0.2354
F-statistic: 62.43 on 2 and 397 DF,  p-value: < 2.2e-16
```

Part (f): The model in Part (a) has R^2 = 0.2393 (with $\hat{\sigma}$ = 2.472) and the model in Part (e) has R^2 = 0.2393 (with $\hat{\sigma}$ = 2.469). Thus these two models have the same value of R^2 but the second model has a smaller estimate of error about the linear fit.

Part (g): We can use the R command confint. Using the model from Part (e) we get

```
> confint( fit_2, level=0.95 )
              2.5 %      97.5 %
(Intercept) 11.79032020 14.27126531
Price       -0.06475984 -0.04419543
USYes        0.69151957  1.70776632
```

Part (h): We can use the plot command on the resulting fit object from Part (e). When we do that we find that the samples 69, 377, and 51 have relatively large residual values and the samples 26, 50, and 368 have large influence. When viewed with the other samples none of these points look extremely incorrect however.

Exercise 11

See the R code chap_3_prob_11.R where this problem is worked.

Part (a): For the requested linear fit we get

```
Call: lm(formula = y ~ x + 0)

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x   1.9939     0.1065   18.73   <2e-16 ***

Residual standard error: 0.9586 on 99 degrees of freedom
Multiple R-squared: 0.7798,     Adjusted R-squared: 0.7776
F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

From the above we see the estimate of β_1 is 1.9939 (quite close to the true value of 2) and note that it is estimated quite significantly (the p-value is quite small). We would reject the hypothesis $\beta_1 = 0$.

Part (b): In this case we would get

```
Call: lm(formula = x ~ y + 0)

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
y  0.39111    0.02089   18.73   <2e-16 ***

Residual standard error: 0.4246 on 99 degrees of freedom
Multiple R-squared: 0.7798,     Adjusted R-squared: 0.7776
F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

Notice that the estimate of β_1 is 0.39111 smaller than the expected value of 0.5 (but very close to the value of 0.4). In general we *don't* expect the slope coefficient of the regression (done in the opposite order) to be equal to the reciprocal of the slope coefficient of the regression done in the original order. There are several ways to see this. I will just present an argument based on simple linear regression without an intercept using some formulas from this chapter and the example data generation process from this problem but the conclusion holds in the more general case. First recall (from Exercise 5 in this chapter) that in simple linear regression without an intercept the slope coefficient β_1 is estimated using

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i'=1}^{n} x_{i'}^2}. \tag{3}$$

Thus for the regression of *Y* onto *X* since *X* is zero mean we have for many samples $\sum_{i'=1}^{n} x_{i'}^2 \sim \text{Var}(X) = 1$ since the samples of *X* are drawn from a standard normal Gaussian. In addition, for the numerator we have

$$\sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n} x_i(2x_i + \epsilon_i) = 2\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_i \epsilon_i \approx 2,$$

as the first sum becomes close to one and the second sum becomes close to zero as *n* gets large. These two facts combine to give us that $\hat{\beta}_1 \sim 2$ as we would expect from the given relationship between *X* and *Y* used to generate the data.

Lets see what happens if we now instead try to regress *X* onto *Y* . In Equation 3 the sum $\sum_{i=1}^{n} x_i y_i$ is symmetric in *x* and *y* and thus does not change but the denominator will change to $\sum_{i'=1}^{n} y_{i'}^2 \sim \text{Var}(Y)$. Thus for regression of *X* on *Y* we would get a estimate of slope given by

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i}{\text{Var}(Y)} = \frac{2}{\text{Var}(Y)}.$$

Since $Y = 2X + \epsilon$, we know that $\text{Var}(Y) = 4\text{Var}(X) + 1 = 5$ and our estimate of β_1 in this case becomes

$$\frac{2}{5} = 0.4,$$

which is the observed value found using R's lm function. If our data generation process had no noise i.e. $Y = 2X$ then $\text{Var}(Y) = 4\text{Var}(X) = 4$ and our slope estimate in this case becomes

$$\frac{2}{4} = 0.5,$$

the more “expected” value. More information on the phenomena of exchanging the dependent and independent variable in simple linear regression can be found by searching for ”the effect of switching response and explanatory variable in simple linear regression” and “the difference between the linear regressions of y on x vs. x on y” in a web search engine.

Part (c): Notice that the t-stat of the coefficient of β_1 is the same in both regressions.

Part (d): To begin we recall the books equation 3.38 given by

$$\hat{\beta} = \frac{\sum_{i=1}^{N} x_i y_i}{\sum_{i'=1}^{N} x_{i'}^2}, \tag{4}$$

and the standard error of $\hat{\beta}$ given by

$$\mathrm{SE}(\hat{\beta}) = \sqrt{\frac{\sum_{i=1}^{N}(y_i - x_i\hat{\beta})^2}{(n-1)\sum_{i'=1}^{n} x_{i'}^2}}.$$

Using these two expressions we can compute the *t*-statistic for $H_0 : \beta = 0$ given by

$$\frac{\hat{\beta}}{\mathrm{SE}(\hat{\beta})} = \frac{\left(\sum_i x_i y_i\right)\sqrt{n-1}\sqrt{\sum_i x_i^2}}{\left(\sum_i x_i^2\right)\sqrt{\sum_i (y_i - x_i\hat{\beta})^2}} = \frac{\sqrt{n-1}\left(\sum_i x_i y_i\right)}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i (y_i - x_i\hat{\beta})^2}}.$$

Using the expression for $\hat{\beta}$ in terms of x_i and y_i we can simplify some as

$$\begin{aligned} \sum_i (y_i - x_i\hat{\beta})^2 &= \sum_i (y_i^2 - 2x_i y_i\hat{\beta} + x_i^2\hat{\beta}^2) \\ &= \sum_i y_i^2 - 2\hat{\beta}\sum_i x_i y_i + \hat{\beta}^2 \sum_i x_i^2 \text{ now use } \hat{\beta} = \frac{\sum_i x_i y_i}{\sum_i x_i^2} \\ &= \sum_i y_i^2 - 2\frac{\left(\sum_i x_i y_i\right)^2}{\sum_i x_i^2} + \frac{\left(\sum_i x_i y_i\right)^2}{\sum_i x_i^2} = \sum_i y_i^2 - \frac{\left(\sum_i x_i y_i\right)^2}{\sum_i x_i^2}. \end{aligned}$$

When we multiply this by $\sum_i x_i^2$ we get

$$\sum_i x_i^2 \sum_i y_i^2 - \left(\sum_i x_i y_i\right)^2.$$

Putting this back into the above expression for the *t*-statistic we get

$$\frac{\sqrt{n-1}\sum_i x_i y_i}{\sqrt{\left(\sum_i x_i^2\right)\left(\sum_i y_i^2\right) - \left(\sum_i x_i y_i\right)^2}},$$

the expression we were to show.

Part (e): Note that the expression for the *t*-statistic given above is symmetric in the variables *x* and *y*. Thus regression of *y* onto *x* will give the same *t*-statistic as regression of *x* onto *y*.

Exercise 12

See the R code chap_3_prob_12.R where this problem is worked.

Part (a): This would happen if $\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i^2$.

Part (b-c): These are performed in the above R script.

Exercise 13

See the R code chap_3_prob_13.R where this problem is worked.

Part (a-c): Each vector is of length 100. The linear model has $\beta_0 = -1$ and $\beta_1 = 0.5$.

Part (e-f): Our linear model fit would have coefficients given by

```
Call: lm(formula = y ~ x)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.02373    0.04838  -21.16   <2e-16 ***
x            0.46253    0.04155   11.13   <2e-16 ***

Residual standard error: 0.4835 on 98 degrees of freedom
Multiple R-squared: 0.5584,     Adjusted R-squared: 0.5539
F-statistic: 123.9 on 1 and 98 DF,  p-value: < 2.2e-16
```

The estimate of β_0 and β_1 are quite good. If we plot this model with the true model we get the plot given in Figure 4. From there we see that the estimated linear model is quite close to the true linear model.

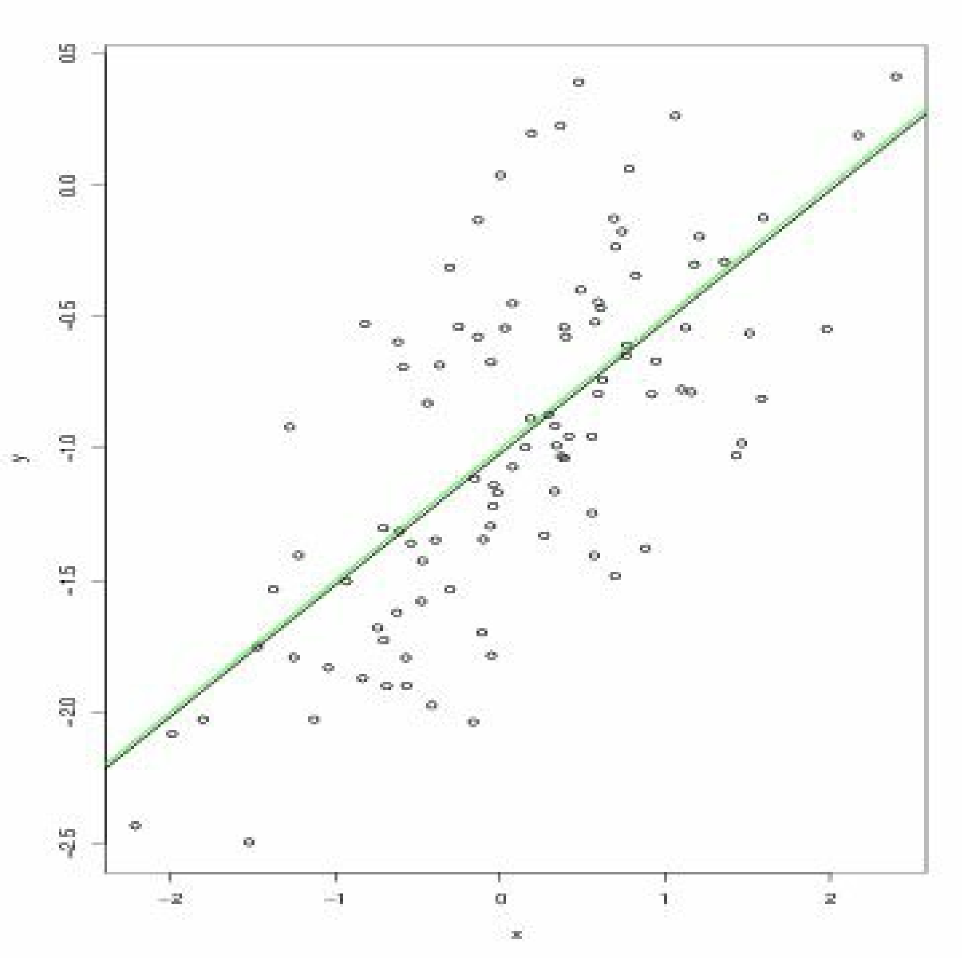

Figure 4: The estimated (in black) and true (in green) linear fit of Y as a function of X for Part (f) of Exercise 13.

Part (g): When we fit a quadratic model and look at the estimate of the quadratic term we get the following (single) line from the summary output

```
Coefficients:
          Estimate Std. Error t value Pr(>|t|)
I(x^2)    -0.05946    0.04238  -1.403    0.164
```

This indicates that there is not much statistical evidence for a quadratic expression.

Part (j): The difference confidence intervals for the original, the model with less noise, and the model with more noise are given by

```
# The original fit:
> confint( fit, level=0.95 )
```

```
                2.5 %     97.5 %
(Intercept) -1.1150804 -0.9226122
x            0.3925794  0.6063602
# With less noise:
> confint( lm( y ~ x ), level=0.95 )
                2.5 %     97.5 %
(Intercept) -1.0924783 -0.9454594
x            0.4789039  0.6422027
# With more noise:
> confint( lm( y ~ x ), level=0.95 )
                2.5 %     97.5 %
(Intercept) -1.1588167 -0.8859310
x            0.2648199  0.5679231
```

Notice that as the noise increases the widths of the intervals increase and as the noise decreases the widths of the intervals decrease. This is in line with what we would expect to happen.

Exercise 14

See the R code chap_3_prob_14.R where this problem is worked.

Part (a): The model in terms of x_1 and x_2 is given by

$$y = 2 + 2x_1 + 0.3x_2,$$

so $\beta_0 = 2$, $\beta_1 = 2$ and $\beta_2 = 0.3$. Our model is really just a model in terms of x_1 however. We can explicitly express it as such using

$$\begin{aligned} y &= 2 + 2x_1 + 0.3(0.5x_1 + \epsilon_1) + \epsilon_2 \\ &= 2 + 2x_1 + 0.15x_1 + \epsilon_3 = 2 + 2.15x_1 + \epsilon_3\,, \end{aligned}$$

where ϵ_3 is a noise term. The regression coefficients in this case are $\beta_0 = 2$ and $\beta_1 = 2.15$.

Part (b): The correlation between x_1 and x_2 is 0.8351212 and a scatterplot between x_1 and x_2 is given in Figure 5. There we see a strong linear correlation between x_1 and x_2.

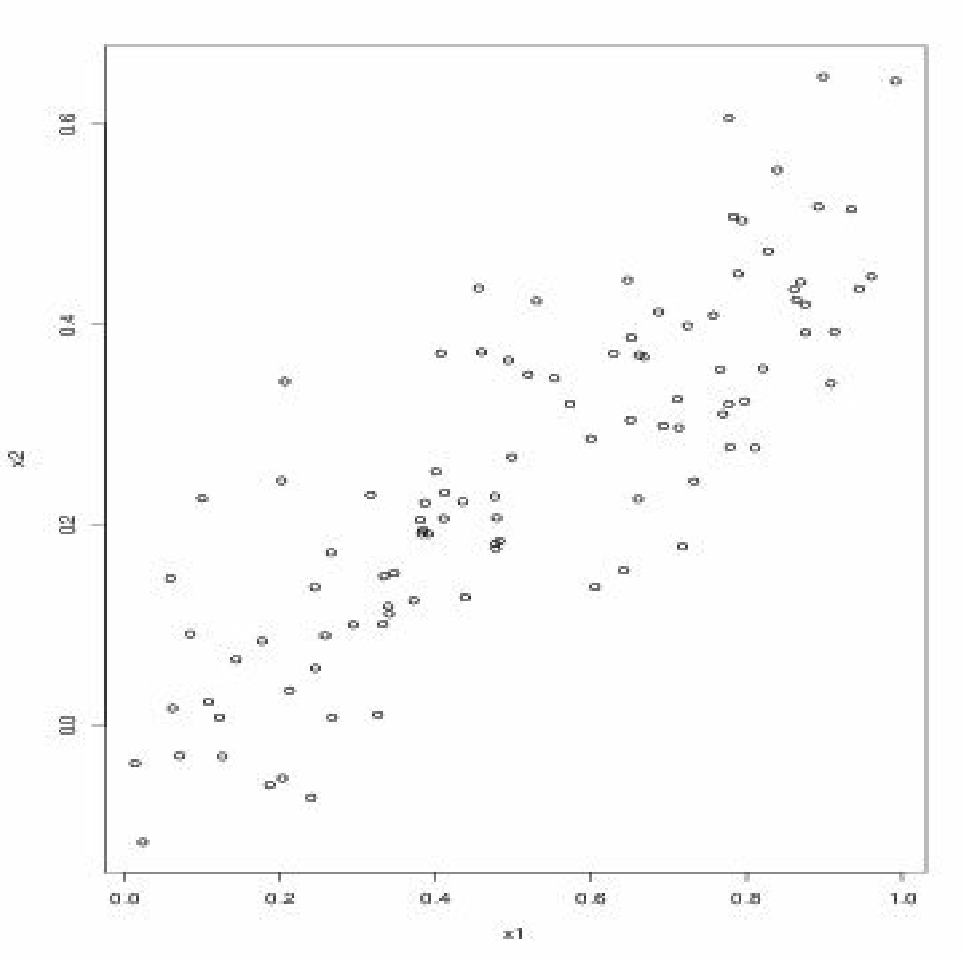

Figure 5: The scatterplot between the variables x_1 and x_2 for Part (b) of Exercise 14.

Part (c): A linear fit gives of y to x_1 and x_2 gives

```
Call: lm(formula = y ~ x1 + x2)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.1305     0.2319   9.188 7.61e-15 ***
x1          1.4396     0.7212   1.996   0.0487 *
x2          1.0097     1.1337   0.891   0.3754

Residual standard error: 1.056 on 97 degrees of freedom
Multiple R-squared: 0.2088,     Adjusted R-squared: 0.1925
F-statistic:  12.8 on 2 and 97 DF,  p-value: 1.164e-05
```

Notice that neither of the coefficients for x_1 or x_2 are highly significant. The estimate of the intercept term seems to reasonably match the truth, while the values found for the other two coefficients are not very close to the "truth" values. From the *p*-values given we can't reject the hypothesis $\beta_i = 0$ for either $i = 1$ or $i = 2$ very strongly.

Part (d): In this case the linear fit gives

```
Call: lm(formula = y ~ x1)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.1124     0.2307   9.155 8.27e-15 ***
x1            1.9759     0.3963   4.986 2.66e-06 ***

Residual standard error: 1.055 on 98 degrees of freedom
Multiple R-squared: 0.2024,     Adjusted R-squared: 0.1942
F-statistic: 24.86 on 1 and 98 DF,  p-value: 2.661e-06
```

We can now reject the hypothesis that $\beta_1 = 0$ quite easily.

Part (e): The linear model of *y* regressed on x_2 again has a significant value for β_1 allowing us to conclude that the model is useful.

Part (f): These results don't contradict each other since with the linear redundancy in x_1 and x_2 there are multiple ways (due to the redundancy) to combined these predictors and obtain the response *y*. These multiple ways make the standard errors of β_1 and β_2 larger and thus each coefficient become less significant.

Part (g): The different models can view the mismeasured point either as a high-leverage point or both as a high-leverage point and an outlier.

Exercise 15

See the R code chap_3_prob_15.R where this problem is worked.

Part (a): I find that every predictor (except chas) gives a significant simple linear regression model at the 1% level when used as a predictor of crim. The different models have different model *F*-statistics and *p*-values but taken individually each predictor is helpful in predicting crim. We can look at the results when we regress crim onto each feature and order the results by *f*-values. We find

```
   feature   f_values     p_values
3     chas   1.579364 2.094345e-01
1       zn  21.102782 5.506472e-06
5       rm  25.450204 6.346703e-07
```

```
10 ptratio  46.259453 2.942922e-11
6      age  71.619402 2.854869e-16
7      dis  84.887810 8.519949e-19
11   black  87.739763 2.487274e-19
13    medv  89.486115 1.173987e-19
2    indus  99.817037 1.450349e-21
4      nox 108.555329 3.751739e-23
12   lstat 132.035125 2.654277e-27
9      tax 259.190294 2.357127e-47
8      rad 323.935172 2.693844e-56
```

Thus the largest *f*-values (and perhaps the best models) correspond to the features tax and rad.

Part (b): Fitting a multiple linear regression model using all the variables indicates that dis and rad are the most significant while the others are less so (many perhaps are not significant at all).

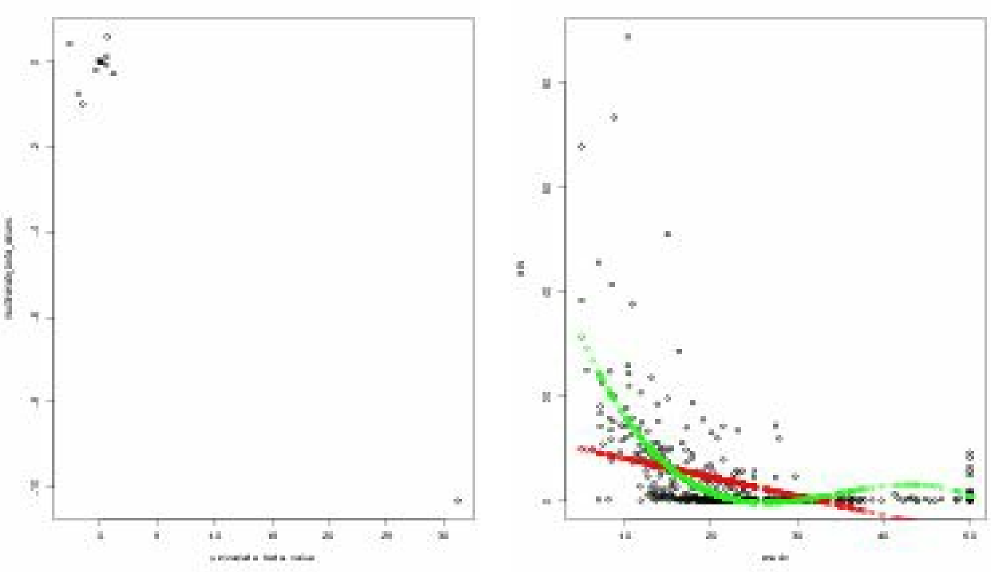

Figure 6: Left: Plots of the β_i coefficients in a multiple linear regression as a function of the β_1 coefficient in a simple linear regression with the same predictor. **Right:** The linear model (in red) and the cubic model (in green) for predicting crim as a function of medv. The feature medv had the cubic model that was the greatest improvement over its linear model.

Part (c): I would argue that the relationship between the coefficients of simple linear regression and multiple linear regression are not related in an easy to understand way. If we look at the plot in Figure 6 (left) we see that (excluding the “outlier” point that results from the simple linear regression using nox as a predictor) the rest of the points are clustered about a small circle with very little correlation between the value of the simple regression coefficient and the value of the multiple regression coefficient.

Part (d): One way to see if at least one non-linear term should be added to the linear regression is to look at a plot of the residuals vs. fitted values that is produced by calling the plot function on the output of the linear fit. If there is noticeable curvature in this plot one might gain some benefit by adding non-linear terms in one of the predictors to the model.

For the question asked in this part of the exercise (where we want to determine if a third order model is better than a first order model for any of the predictors), we can fit the third order model to the data and observe if the *f*-value of the cubic fit is sufficiently larger than the value we get from the simple linear regression fit. We can do this in R using the anova command. In the R code that accompanies this exercise we do this. We then order the results so that the single predictor with the largest ANOVA *f*-value is listed last and we get

```
   feature    f_values
11   black   0.4622222
12   lstat   3.3190437
8      rad   3.6732699
1       zn   4.8118205
5       rm   5.3088168
10 ptratio   8.4155300
9      tax  11.6400227
6      age  15.1400633
2    indus  31.9869602
4      nox  42.7581707
7      dis  46.4603654
13    medv 116.6340058
3     chas          NA
```

The predictor medv is the predictor that has the most improvement (in its fit) when moving to the cubic model. We can see how much better the cubic model fits than the linear model by looking at a scatter plot of the data with both the linear and the cubic fits. When we do that we get the plot shown in Figure 6 (right). There we can visually see that the cubic model looks to fit the data better than the linear model.

Chapter 4 (Classification)

Conceptual Exercises

Exercise 1

The books Eq. 4.2 is

$$p(X) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}. \tag{5}$$

The following manipulations derive the intended expression. We first multiply both sides by the denominator of the right-hand-side to get

$$(1+e^{\beta_0+\beta_1 x})p(X) = e^{\beta_0+\beta_1 x}.$$

Now get all the exponential terms on one side to get

$$p(X) = e^{\beta_0+\beta_1 x}(1-p(X)).$$

Now solve for the exponential to get

$$e^{\beta_0+\beta_1 x} = \frac{p(X)}{1-p(X)},$$

as we were to show.

Exercise 2

Before we start we recall that the books Eq. 4.12 is

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}, \tag{6}$$

and the books Eq. 4.13 is

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k). \tag{7}$$

Both expressions are valid for $1 \leq k \leq K$. First note that once x is given (i.e. observed) the denominator of Equation 6 is the same for all k. Thus the maximum over k selected does not depend on its numerical value. Thus we need only consider the maximum over k of the following expressions

$$\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right).$$

As each term like this has the same multiplier $\frac{1}{\sqrt{2\pi}\sigma}$ the numerical value of that multiplier won’t affect the selection of which of the K items is the largest and we can ignore it. Thus we need to find the maximum over *k* of

$$\pi_k \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right).$$

As the logarithm is a monotone increasing function of its argument the maximum in *k* of the above is still the maximum when we take the logarithm of the given expressions. Taking the logarithm gives

$$\log(\pi_k) - \frac{1}{2\sigma^2}(x-\mu_k)^2.$$

Expanding the quadratic in the above gives

$$\log(\pi_i) - \frac{1}{2\sigma^2}(x^2 - 2x\mu_k + \mu_k^2).$$

As the term $-\frac{1}{2\sigma^2}x^2$ is common to all expressions we can drop it and only seek to maximize

$$x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k),$$

which is the same seeking to maximize the books Eq. 4.13 or Equation 7 which is what we were to show.

Exercise 3

The densities of *X* in each class are given by the books Eq. 4.11. When σ^2 is different in each class, the expression for the posteriori probability $p_k(x)$ now takes the form

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K}\pi_l \frac{1}{\sqrt{2\pi}\sigma_l}\exp\left(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2\right)}.$$

Following the same arguments as in the previous problem (the denominator in the above expression has the *same* numerical value for every value of *k* and its value does not affect the decision rule) we would need to pick the *k* that maximizes the numerator or (dropping constant factors)

$$\frac{\pi_k}{\sigma_k}\exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right).$$

On taking logarithms and expanding we get

$$\log(\pi_k) - \log(\sigma_k) - \frac{1}{2\sigma_k^2}(x^2 - 2x\mu_k + \mu_k^2).$$

In this case we can't ignore the term $-\frac{x^2}{2\sigma_k^2}$ since it is different for different values of k. The above expression is quadratic in x i.e. the boundaries between classes in the feature space x are quadratic and not linear as they were when we had the same value for σ^2 in all classes i.e. $\sigma_1^2 = \sigma_2^2 = . . . = \sigma_K^2 = \sigma^2$.

Exercise 4

Part (a): Under an infinite number of training samples we would have 0.1 of the available observations to use in making a prediction.

Part (b): In this case we would have $0.1^2 = 0.01$ of the available observations to use in making a prediction.

Part (c): In this case we would have $0.1^{100} = 10^{-100}$ of the available observations to use in making a prediction.Note that this is a very small number and not likely to be very many samples in an a "real life" data set.

Part (d): Let x_0 be the point where we wish to make a prediction. Note that as p increases the average number of available observations that are within a given percent of x_0 gets very small very fast and thus there are very few training samples near x_0 to use.

Part (e): If our hypercube around x_0 needs to contain 10% of our training samples, then depending on p, it would need to have lengths l such that

when $p = 1$ need $l = 0.1$
when $p = 2$ need $l^2 = 0.1$ so $l = \sqrt{0.1} = 0.316227$
when $p = 100$ need $l^{100} = 0.1$ so $l = 0.1^{1/100} = 0.9772$.

Notice that when p is large l is so close to 1 that is hard to argue that the method we are using is *local* since we need to use training points almost the entire dimension of the hypercube away from x_0 to make our prediction.

Exercise 5 (LDA vs. QDA)

Part (a): Since LDA has fewer parameters to fit compared to QDA it will perform worse on the training data set. In other words it is easier for QDA to overfit the data. Since the true decision boundary is linear LDA will do better than QDA on the test set since it has less variance than QDA.

Part (b): The same argument as in Part (a) implies that QDA would have a smaller training set error. A non-linear boundary would be better fit by QDA and so we expect it to perform better on the test set. In this case LDA is a biased estimate of the true non-linear decision boundary.

Part (c): If the true decision boundary is linear then as *n* increase its decision boundary should limit to that found by LDA (since a linear surface is a specification of a quadratic surface) and thus its performance on the test set should increase more than LDA. If the true decision boundary is non-linear then QDA will perform better than LDA on the test data set. With more data QDA should obtain a more accurate estimate of the non-linear decision boundary and its performance should increase. In this case LDA is biased away from the true decision boundary and no amount of data will help its performance on test data. In both cases it looks like as *n* increase QDA performance to improve relative to that of LDA.

Part (d): False. QDA has more variance than LDA and when fitting to training data this variance will cause the decision boundary to be moved away from the truth by noise in the training data set. LDA has less variance and will likely be more accurate in its approximation of the linear decision boundary with the training data given than QDA.

Exercise 6

Part (a): In this case X_1 = 40 and X_2 = 3.5 so our estimate of the probability is given by

$$\log\left(\frac{P(\text{Gets A})}{1 - P(\text{Gets A})}\right) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 = -6 + 0.05(40) + 3.5 = -0.5.$$

Solving this for *P*(Gets A) we find *P*(Gets A) = 0.3775.

Part (b): We want *P*(Gets A) ≥ 0.5. This last expression means that1 − *P*(Gets A) ≤ 0.5 and thus

$$\frac{P(\text{Gets A})}{1 - P(\text{Gets A})} \geq \frac{0.5}{0.5} = 1.$$

Thus taking logarithms we want to find value of X_1 such that

$$\log\left(\frac{P(\text{Gets A})}{1 - P(\text{Gets A})}\right) \geq \log(1) = 0.$$

In order for this be true based on our assumed model of *P*(Gets A) we need

$$\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 \geq 0.$$

or putting in the values of $\hat{\beta}_i$ we have

$$-6 + 0.05 X_1 + 3.5 \geq 0.$$

Solving for X_1 we get X_1 ≥ 50 hours.

Exercise 7

From the statement of the problem we are told that

$$
\begin{aligned}
P(X|\text{Dividend=Yes}) &= \frac{1}{\sqrt{2\pi 6}} \exp\left(-\frac{1}{2(36)}(X-10)^2\right) \\
P(X|\text{Dividend=No}) &= \frac{1}{\sqrt{2\pi 6}} \exp\left(-\frac{1}{2(36)}X^2\right) \text{ and} \\
P(\text{Dividend=Yes}) &= 0.8\,.
\end{aligned}
$$

We want to compute *P*(Dividend=Yes|*X*). We can do this using Bayes' rule where we get

$$P(\text{Dividend=Yes}|X) = \frac{P(X|\text{Dividend=Yes})P(\text{Dividend=Yes})}{P(X|\text{Dividend=Yes})P(\text{Dividend=Yes}) + P(X|\text{Dividend=No})P(\text{Dividend=No})}.$$

When *X* = 4 we have (using R notation) that

$$
\begin{aligned}
P(X|\text{Dividend=Yes}) &= \mathit{dnorm}(4, 10, 6) = 0.0403 \\
P(X|\text{Dividend=No}) &= \mathit{dnorm}(4, 0, 6) = 0.05324\,.
\end{aligned}
$$

Using these we compute the requested expression

$$P(\text{Dividend=Yes}|X) = \frac{0.0403(0.8)}{0.0403(0.8) + 0.05324(0.2)} = 0.7517.$$

Exercise 8

Note that the 1-nearest-neighbor method will have a zero error rate on the training data set since it is "memorizing" the data. Using this fact, the test error rate for 1-nearest-neighbor must satisfy

$$\frac{1}{2}(0 + \text{test error}) = 0.18 \quad \text{so} \quad \text{test error} = 0.36.$$

Since the logistic regression algorithm reported a test error rate of 0.3 we would prefer that method since it has a smaller test error rate.

Exercise 9 (computing odds)

Recall that *odds* is defined as

$$\text{odds} = \frac{p}{1-p}, \qquad (8)$$

where *p* is the probability of some event happening. We can solve the above for *p* in terms of odds to get

$$p = \frac{\text{odds}}{1 + \text{odds}}.$$

Note the symmetry in these two formulas.

Part (a): If odds = 0.37 then using the above formula for *p* gives us *p* = $\frac{0.37}{1.37}$ = 0.27.

Part (b): If we have *p*(Default) = 0.16 then the above formula for odds gives odds = 0.1904762.

Applied Exercises

Exercise 10

See the R code chap_4_prob_10.R where this problem is worked.

Part (a): To study if this classification problem will be "easy" or "hard" I first looked at the correlation between the item we want to predict i.e. Direction and the inputs. Since Direction is a factor variable I first transformed it into a numerical variable where −1 represents a down move and +1 represents an up move. The correlations with this numeric variable are then given by

```
> Weekly.cor[9,]
        Year          Lag1          Lag2          Lag3
   -0.02220025    -0.05000380    0.07269634    -0.02291281
        Lag4          Lag5         Volume         Today
   -0.02054946    -0.01816827    -0.01799521    0.72002470
NumericDirection
    1.00000000
```

Notice that all of these (except for Today) are very small numbers. The variable Today is how the variable Direction was derived and so we expect a strong correlation between them. Since the actual percentage change over the given week is saved in the variable Today we can look if any of the variables are highly correlated with this variable (in case we wanted to try a regression technique to predict the numerical value of Today)

```
> Weekly.cor[8,]
        Year          Lag1          Lag2          Lag3
  -0.032459894    -0.075031842    0.059166717    -0.071243639
        Lag4          Lag5         Volume         Today
  -0.007825873    0.011012698    -0.033077783    1.000000000
NumericDirection
   0.720024704
```

Notice that only NumericDirection has a significant correlation. A pairs plot shows further that predicting Up and Down will be difficult.

Part (b): The summary command from the full logistic regression fit is given by

```
Call: glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Coefficients:
        Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1       -0.04127    0.02641  -1.563   0.1181
Lag2        0.05844    0.02686   2.175   0.0296 *
Lag3       -0.01606    0.02666  -0.602   0.5469
Lag4       -0.02779    0.02646  -1.050   0.2937
Lag5       -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
```

Notice that only Lag2 is slightly significant.

Part (c): The confusion matrix is given by

```
      truth
predicted Down  Up
   Down   54  48
   Up    430 557
```

From this it looks like when we predict Up we are correct more often than when we predict Down and in fact we don’t predict down very much only since

```
> table(y_hat)
y_hat
Down   Up
 102  987
```

states that we only predict down 102 times and predict up 987 times. The overall movement of the market was up during this time so our classifier may simply be reporting this fact.

Part (d-h): When we accumulate the different out-of-sample predictions from each of the different methods (after running the given R code) we are left with

```
[1] "LR (all features): overall fraction correct=   0.561065"
[1] "LR (only Lag2): overall fraction correct=   0.625000"
[1] "LDA (only Lag2): overall fraction correct=   0.625000"
[1] "QDA (only Lag2): overall fraction correct=   0.586538"
```

```
[1] "KNN (k=1): overall fraction correct=   0.509615"
[1] "KNN (k=1): overall fraction correct=   0.548077"
```

From this output it looks like logistic regression and LDA using only the feature Lag2 perform best on the out-of-sample data. One could study further the specific type of errors made by these two classifiers to determine if they are such that we in fact have a helpful algorithm.

Exercise 11

See the R code chap_4_prob_11.R where this problem is worked.

Part (b): We start by considering the correlation of the response mpg01 against all of the other predictors. We find

```
                  mpg01
cylinders     -0.7591939
displacement  -0.7534766
horsepower    -0.6670526
weight        -0.7577566
acceleration   0.3468215
year           0.4299042
origin         0.5136984
mpg01          1.0000000
```

Notice that many predictors seem very highly correlated with mpg01. Specifically the variables cylinders, displacement, and weight seem to have the strongest correlation with mpg01.

Part (c-f): Next we fit LDA, QDA, and logistic regression models on using this data. We find test sample results for each method given by

```
[1] "LDA: overall fraction correct=   0.938776"
[1] "QDA: overall fraction correct=   0.948980"
[1] "LR (all features): overall fraction correct=   0.928571"
```

Based on this simple study QDA seems to be the best classification method from the three tried.

Exercise 12

A few of these functions are implemented in the R code chap_4_prob_12.R.

Exercise 13

See the R code chap_4_prob_13.R where this problem is worked.

Part (b): Looking at the correlation of a binary representation of the feature crim with the other variables in the data set we get

```
> sort( Boston.cor['crim01',] )
        dis          zn       black        medv          rm        chas 
-0.61634164 -0.43615103 -0.35121093 -0.26301673 -0.15637178  0.07009677 
    ptratio       lstat       indus         tax         age         rad 
 0.25356836  0.45326273  0.60326017  0.60874128  0.61393992  0.61978625 
        nox      crim01 
 0.72323480  1.00000000 
```

From this simple analysis we might select the features that are most highly correlated with crim01 to use in our models. Specifically for this exercise we will use nox, rad, and dis. We then split our data set into two parts the training and testing. Using the training data set we learn models for logistic regression, LDA, QDA, and KNN. We then predict the performance on the test data set. When we do that we get error rates given by

```
[1] "LR: overall fraction correct=   0.850394"
[1] "LDA: overall fraction correct=   0.811024"
[1] "QDA: overall fraction correct=   0.811024"
[1] "KNN (k=1): overall fraction correct=   0.921260"
[1] "KNN (k=3): overall fraction correct=   0.905512"
```

From this data it looks like the best prediction method is given by 1-nearest-neighbor.

Chapter 5 (Resampling Methods)

Conceptual Exercises

Exercise 1

We can use the properties of the variance to write the expression we seek to minimize as

$$\mathrm{Var}\left(\alpha X + (1-\alpha)Y\right) = \alpha^2 \mathrm{Var}\left(X\right) + (1-\alpha)^2 \mathrm{Var}\left(Y\right) + 2\alpha(1-\alpha)\mathrm{Cov}\left(X, Y\right). \quad (9)$$

To minimize this with respect to *α* we can take the derivative of the above with respect to *α*, set the result equal to zero, and then solve for *α*. We find the first derivative of the above given by

$$2\alpha \mathrm{Var}\left(X\right) + 2(1-\alpha)(-1)\mathrm{Var}\left(Y\right) + 2(1-2\alpha)\mathrm{Cov}\left(X, Y\right).$$

If we set this equal to zero and then solve for *α* we get

$$\alpha = \frac{\text{Var}(Y) - \text{Cov}(X,Y)}{\text{Var}(X) + \text{Var}(Y) - 2\text{Cov}(X,Y)},$$

the expression we were to show.

Exercise 2

Part (a): For any of the bootstrap samples to not be the *j*th sample from the original dataset will happen with probability $\frac{n-1}{n}$ = 1 −$\frac{1}{n}$, since there are *n* − 1 samples in the original dataset that exclude the *j*th from a total of *n* samples.

Part (b): This is the same as Part (a) or $\frac{n-1}{n}$.

Part (c): To have the *j*th sample from the original dataset not in the bootstrap sample will happen if its not the first, second, third etc. all the way to the *n*th. This happens with the probability from Part (a-b) to the *n*th power or

$$\left(1 - \frac{1}{n}\right)^n.$$

Part (d): This would be the probability complement of the probability in Part (c) or

$$1 - \left(1 - \frac{1}{n}\right)^n = 0.67232.$$

Thus for a small samples size *n* = 5 there is a decent probability that the *j*th sample is *in* a given bootstrap sample.

Part (e): When *n* = 100 the calculation above gives the value 0.633968.

Part (f): When *n* = 100 the calculation above gives the value 0.632305. Based on these few sample values this seems to be a very slowly decreasing function of *n*.

Part (g): Note that from the limit

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = e^{-1} = 0.3678794,$$

the formula we are plotting for large *n* limits to the value of 1 − 0.3678794 = 0.6321206. Because of this we don't need to plot the given expression for *n* as large as suggested to get an idea of the curves behavior. We can do the plotting with the simple R script

```
ns = 1:2000
prob_j_in_sample = 1 - ( 1 - 1/ns )^ns
plot( ns, prob_j_in_sample, ylim=c(0.6,0.7), type='l' )
grid()
```

When we run this code we get the plot given in Figure 7.

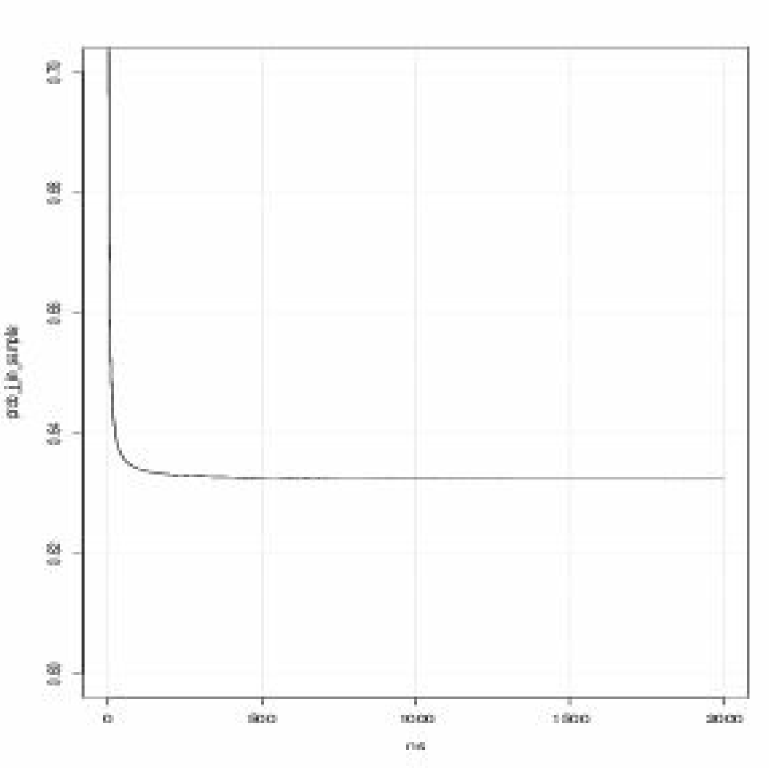

Figure 7: A plot of the probability sample *j* is included in a given bootstrap sample. Note that this curve asymptotes to a value around 0.632.

Part (h): Running the give R code we get the value of 0.6263, close to the value calculated above.

Exercise 3

Part (a): *k*-fold cross validation is implemented by dividing the entire dataset into *k* parts of equal size. Each of the *k* parts is called a cross-validation “fold”. We then fit our model *k* times ignoring the data in one of the *k* folds. Once we have the model fit we can use the data held out to estimate our models performance on the held-out fold. This procedure then gives *k* estimates of our models performance on unseen data, that we can use to obtain an estimate (via averaging) of the performance on unseen data.

Part (b): Each of the three techniques discussed has more or less computational requirements in that the learning procedure needs to be fit several times (more or less times) depending on the validation method used.

In the validation set approach we have only one training and testing set obtained by dividing the original dataset into two pieces. Thus this approach has a risk of *over* estimating the test set error as the training set size is smaller than (by a factor of two) the total amount of data *n*. This is because many learning procedures perform better on the test set when given more data.

In comparing *k*-fold cross-validation with LOOCV we note that LOOCV has less bias since it is using almost all of the training data to build our model. A drawback with LOOCV is that all of the training sets used for building the model are almost identical and are therefore highly correlated. Since the variance of the mean obtained from samples that are highly correlated is larger than the variance of the mean obtained from samples that are independent the variance of LOOCV will be larger than that of *k*-fold cross-validation. This argument (along with the computation one) make *k*-fold cross-validation with $k = 5$ or $k = 10$ the preferred validation approach.

Exercise 4

We could apply the techniques of this chapter, namely applying a LOOCV type technique we would have $n - 1$ estimates of Y (less computationally intensive methods could be devised). The variance of these estimates of Y could be used to estimate the variance of our prediction.

Applied Exercises

Exercise 5

See the R code chap_5_prob_5.R where this problem is worked.

Part (b): Using the validation set approach we find a validation error rate given by 0.0268.

Part (c): Three more validation set estimates give values 0.0236, 0.026, and 0.0304. We note that these values are very close together indicating that our method (in this case) may not have much variance.

Part (d): When we add the “student” predictor we get a validation error of 0.0272. Notice that this is the range of the output given above indicating that it is not clear (from this analysis) whether or not this predictor is helpful in predicting default.

Exercise 6

See the R code chap_5_prob_6.R where this problem is worked.

Part (a): The summary command when we fit using all of the data gives

Call:

```
glm(formula = default ~ income + balance, family = "binomial", data = Default)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
```

Notice that all of the coefficients are significant and have standard errors as given.

Part (b-c): When we run the boot function with the boot.fn written to estimate the standard errors of the intercept, the coefficient of income, and the coefficient of balance we get

```
        original        bias     std. error
t1* -1.154047e+01 -3.395100e-02 4.403472e-01
t2*  2.080898e-05  2.572090e-07 4.979363e-06
t3*  5.647103e-03  1.396819e-05 2.331213e-04
```

Part (d): The standard errors given via the bootstrap and from the output of the glm call are quite similar.

Exercise 7

See the R code chap_5_prob_7.R where this problem is worked.

Part (a-c): I get that the first sample was classified as “Up” while the truth in this case should have been classified as “Down”. Thus the first sample was classified in error.

Part (d-e): When we run the given R script we compute the leave one out error rate to be 0.449954.

Exercise 8

See the R code chap_5_prob_8.R where this problem is worked.

Part (a): In this data set we have n = 100 and p = 2 (assuming predictors of x and x^2).

Part (b): A scatter plot of the data is given in Figure 8

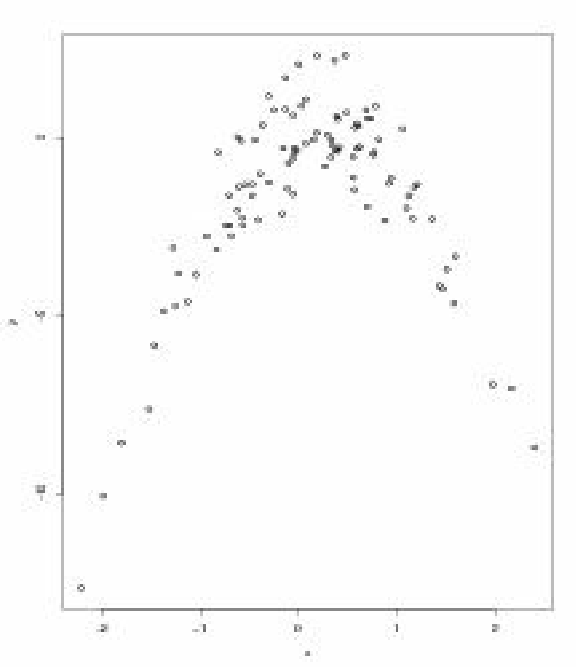

Figure 8: The scatter plot *x* vs. *y* for the variables given in Problem 8.

Part (c-d): When we perform the requested LOOCV we get the following output

```
[1] "Model (i): cv output=   7.288162"
[1] "Model (ii): cv output=   0.937424"
[1] "Model (iii): cv output=   0.956622"
[1] "Model (iv): cv output=   0.953905"
```

These will be the *same* if we change the random seed since LOOCV is a deterministic procedure.

Part (e): The quadratic model has the smallest LOOCV error, which is what is expected since the data was generated from a quadratic model.

Exercise 9

See the R code chap_5_prob_9.R where this problem is worked.

Part (a-b): I compute $\hat{\mu}$ = 22.53281, and SE(μ) = 0.4088611.

Part (c): The output from boot function gives

```
Bootstrap Statistics :
    original      bias    std. error
```

```
t1* 22.53281 -0.01884625   0.4105924
```

Notice that this is slightly larger than the estimate computed in Part (b).

Part (d): We would compute a 95% confidence interval is approximately given using the following R code

```
mu_hat + 2 * c(-1,+1) * 0.413101
```

which gives the range of values for μ of (21.70660, 23.35901). The output from the R command t.test gives (among other things)

```
95 percent confidence interval:
 21.72953 23.33608
```

Notice that the bootstrap confidence interval is slightly larger than the one obtained from t.test.

Part (e-f): For the median I get the value of 21.2 and a standard error using the bootstrap of 0.3713003.

Part (g-h): For these two statistics I compute 12.75 with a standard error of 0.5103471. Notice that this standard error is the largest we have observed thus far.

Chapter 6 (Linear Model Selection and Regularization)

Conceptual Exercises

Exercise 1

Part (a): As best subset is searching over the largest possible set of models with *k* predictors it will have the lowest training set error of the three choices.

Part (b): Which of the three sets will have smallest test error is hard to tell without testing each on a validation data set.

Part (c):

- **Part (i):** True. In forward stepwise selection in moving from the model with *k* predictors to the one with *k* +1 we add the predictor which reduces the RSS the most.
- **Part (ii):** True. In backwards stepwise selection we start with the model of *k* + 1 predictors and remove the single predictor which (when removed) leaves us with the smallest RSS.
- **Part (iii):** False. The predictors selected in forward and backwards selection depend on the predictors selected (or deleted) at earlier stages. The predictors one gets from each procedure could be different for a given *k* depending on the selection (deletion) order.
- **Part (iv):** False for the same reason as in Part (iii).
- **Part (v):** False. For each *k*, the best subset prediction procedure selects the subset that has the lowest RSS on the training set. As shown in Table 6.1 (in going from *k* = 3 to *k* = 4) these selected predictors at each stage don't have to be nested.

Exercise 2

Part (a): The lasso with non-zero λ, since it has a constraint on the size of the total budget $\sum_{j=1}^{p} |\beta_j|$ allowed, is less flexible than least squares. This means that it will give improved prediction accuracy when the increase in bias is less than its decrease in variance.

Part (b): The ridge regression technique and the lasso technique are very similar and the answer to this part is the same as in Part (a).

Part (c): A non-linear method will be able to fit more complex function mappings $Y = f(X)$ and thus will be a more flexible method than linear least squares which assumes linearity. The non-linear procedure will give better prediction accuracy when its increase in variance (due to having to fit more free parameters) is less than its decrease in bias (due to its more flexible nature being able to fit more complex functions).

Exercise 3

Part (a): When *s* is zero we have a highly constrained model (the coefficients of β_j must all be zero) and thus will have a relatively high RSS value. As we let *s* increase we move to more flexible models (eventually obtaining the full least squares model) and we expect the training RSS to steadily decrease as *s* increase.

Part (b): The test RSS will involve a bias variance trade off and thus should decrease initially and then eventually start to increase. This behavior is seen in the examples in this book for example in Figure 6.8 and 6.9 (when the plots are viewed from right to left since as the parameter *s* increases the value of λ decreases).

Part (c): The variance of our fit will steadily increase as *s* is increased as we are allowing more and more flexible models.

Part (d): The squared bias of our fit will steadily decrease as *s* is increases and we move towards the full least squares model.

Part (e): The irreducible error is not modeled by our fitting procedure (and predictors) and thus will remain constant regardless of the value of *s*.

Exercise 4

Note that this is the "same" optimization problem as Problem 3 but with an L_2 penalty (rather than an L_1 penalty) applied to the β_j values.

Part (a): When λ is very large our problem is highly constrained (the coefficients β_j must all be zero). This means that our technique will have high bias (be far from the assumed true linear model) and have low variance (will have estimated coefficients independent of the given dataset). When λ is zero, our problem is equivalent to least squares and has lower bias but higher variance. Thus as λ increases from 0 the training RSS will start small and increase as λ increases.

Part (b): The test RSS should decrease initially and then then increase in a "U" like shape. This behavior is seen in the examples presented in this chapter (for example Figure 6.5).

Part (c): As λ increases the variance will decrease as we are further constraining our model (see Figure 6.5).

Part (d): As λ increases the (squared) bias will increase (again see Figure 6.5).

Part (e): The irreducible error is not able to be modeled by our fitting procedure (and predictors) and thus will remain constant regardless of the value of λ.

Exercise 5

Part (a-b): The data matrix *X*, where rows represent data samples and columns represent features, for this problem *X* takes the form

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{11} \\ 1 & x_{22} & x_{22} \end{bmatrix}.$$

Where we have used the fact that $x_{11} = x_{12}$ and $x_{21} = x_{22}$. From the condition that $x_{11} + x_{22} = 0$ (or $x_{22} = -x_{11}$) we see that *X* looks like

$$X = \begin{bmatrix} 1 & x_{11} & x_{11} \\ 1 & -x_{11} & -x_{11} \end{bmatrix}.$$

In the above representation of *X* we can see that the two right-most columns of *X* are *equal* i.e. the correlation between these two features is one, the largest value possible. From the discussion in the section on ridge regression in the book we have that if the columns of *X* have been centered to have mean zero (as the ones above are) then β_0 given by

$$\hat{\beta}_0 = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i = 0.$$

In this case from the books equation 6.5 the ridge regression coefficients are given by fixing a value for λ and minimizing the following objective function

$$(y_1 - (\beta_1 x_{11} + \beta_2 x_{12}))^2 + (y_2 - (\beta_1 x_{21} + \beta_2 x_{22}))^2 + \lambda(\beta_1^2 + \beta_2^2).$$

Since $x_{12} = x_{11}$ and $x_{21} = x_{22}$ the above is equal to

$$(y_1 - (\beta_1 + \beta_2)x_{11})^2 + (y_2 - (\beta_1 + \beta_2)x_{22})^2 + \lambda(\beta_1^2 + \beta_2^2).$$

Next using $x_{11} = -x_{21} = -x_{22}$ we get for our objective $\mathcal{L}(\beta_1,\beta_2)$

$$\mathcal{L} \equiv (y_1 - (\beta_1 + \beta_2)x_{11})^2 + (y_2 + (\beta_1 + \beta_2)x_{11})^2 + \lambda(\beta_1^2 + \beta_2^2).$$

To optimize this objective with respect to the unknowns β_1 and β_2 we take the derivatives and set the results equal to zero to get

$$\frac{\partial \mathcal{L}}{\partial \beta_1} = 2(y_1 - (\beta_1 + \beta_2)x_{11})(-x_{11}) + 2(y_2 + (\beta_1 + \beta_2)x_{11})x_{11} + 2\lambda\beta_1 = 0$$

$$\frac{\partial \mathcal{L}}{\partial \beta_2} = 2(y_1 - (\beta_1 + \beta_2)x_{11})(-x_{11}) + 2(y_2 + (\beta_1 + \beta_2)x_{11})x_{11} + 2\lambda\beta_2 = 0\,.$$

Notice that these the second equation is equivalent to the first equation with $\beta_2 \rightarrow \beta_1$ and vice versa. Thus if we find a solution (β_1,β_2) to the first equation then (β_2,β_1) will be a solution to the second equation. This makes us conclude that $\hat{\beta}_1 = \hat{\beta}_2$.

Part (c-d): In this case, using the books equation 6.7 the lasso coefficients are given by fixing a value for *λ* and minimizing the following objective function $\mathcal{L}(\beta_1,\beta_2)$

$$\mathcal{L} \equiv (y_1 - (\beta_1 + \beta_2)x_{11})^2 + (-y_1 + (\beta_1 + \beta_2)x_{11})^2 + \lambda(|\beta_1| + |\beta_2|).$$

We have used the the previous parts to simplify some and used the fact that $y_2 = -y_1$. The first two terms can be combined to give

$$\mathcal{L} = 2(y_1 - (\beta_1 + \beta_2)x_{11})^2 + \lambda(|\beta_1| + |\beta_2|).$$

In the above expression we see that if both values of β_i are the same sign (either positive or negative) the above has many solutions for the two values of β_i on a line i.e $\beta_1 + \beta_2 = c$ where the value of *c* provides the minimal value of $\mathcal{L}$. For example, if $\beta_i > 0$ for $i = 1, 2$ then $\mathcal{L}$ is equivalent to

$$\mathcal{L} = 2(y_1 - (\beta_1 + \beta_2)x_{11})^2 + \lambda(\beta_1 + \beta_2).$$

Taking $v \equiv \beta_1 + \beta_2$ we can take the derivative of $\mathcal{L}$ with respect to *v*, set the result equal to zero, and solve for *v* to get the minimal value for $\mathcal{L}$. We find this value of *v* given by

$$v = \frac{4y_1x_{11} - \lambda}{4x_{11}^2}.$$

Then any two values of β_1 and β_2 that sum to *v* will provide a minimum value for $\mathcal{L}(\beta_1,\beta_2)$.

Exercise 6

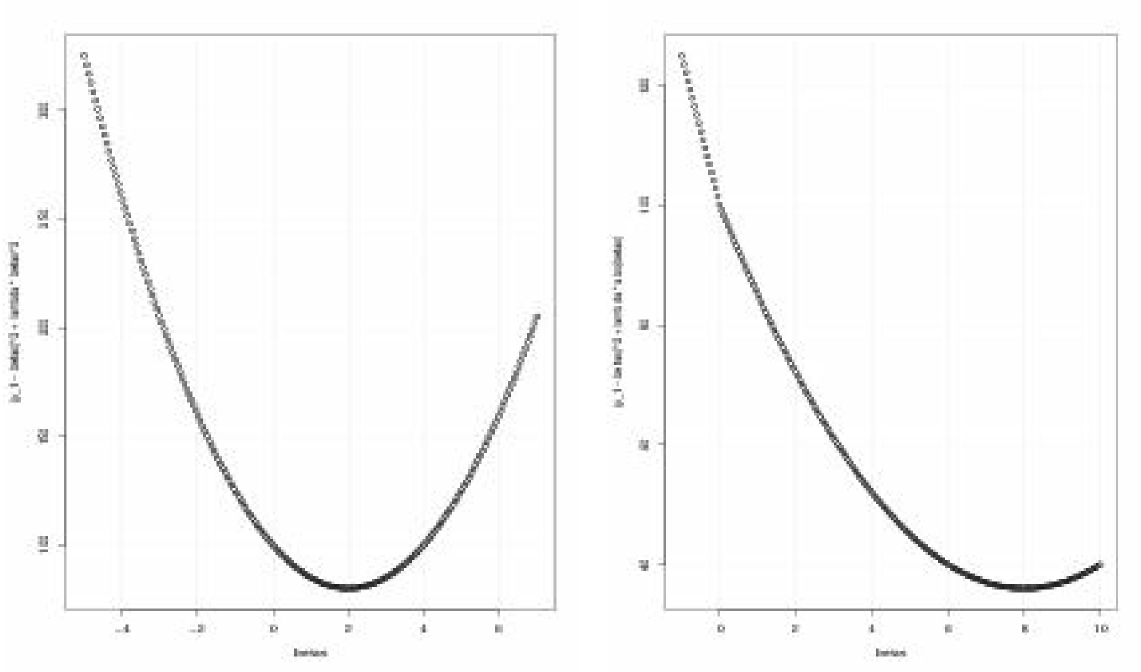

Figure 9: Left: Plots of the objective function for ridge regression. **Right:** Plots of the objective function for the lasso. Note that the range of *x*-axis in each plot is different.

Part (a): The books equation 6.12 is given by

$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2. \tag{10}$$

When *p* = 1 this becomes

$$(y_1 - \beta_1)^2 + \lambda \beta_1^2.$$

Following the problem we pick y_1 = 10 and λ = 4 and plot the above optimization objective as a function of β_1. When we do this we get the plot given in Figure 9 (left). Note that in this case the books equation 6.14 gives

$$\hat{\beta}_1^R = \frac{y_1}{1 + \lambda} = 2, \tag{11}$$

which is the location of the minimum of the plot.

Part (b): The books equation 6.13 is given by

$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p}|\beta_j|. \tag{12}$$

When p = 1 this is

$$(y_1 - \beta_1)^2 + \lambda|\beta_1|.$$

Again we pick y_1 = 10 and λ = 4 and plot the above optimization objective as a function of β_1. When we do this we get the plot given in Figure 9 (right). In this case the books equation 6.15 gives

$$\hat{\beta}_1^L = 10 - 2 = 8,$$

which is the location of the minimum of the plot.

Exercise 7 (the Bayesian connection)

Part (a): Our residuals assuming the given model for *Y* are

$$\epsilon_i = y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j.$$

Then the likelihood is defined as

$$L = p(\{\epsilon_i\}_{i=1}^N|\beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2}{2\sigma^2}\right\}.$$

Part (b): To apply Bayes rule for the estimation of β we recall that

$$p(\beta|\{\epsilon_i\}_{i=1}^N) \propto p(\{\epsilon_i\}_{i=1}^N|\beta)p(\beta).$$

Then if each β_i is distributed according to a double exponential we have

$$p(\beta_i) = \frac{1}{2b}\exp\left(-\frac{|\beta_i|}{b}\right),$$

so that using L for the likelihood introduced above and the independence of β_i we get

$$\begin{aligned} p(\beta|\{\epsilon_i\}_{i=1}^N) &\propto L \times \left(\frac{1}{(2b)^p}\exp\left(-\frac{1}{b}\sum_{j=1}^{p}|\beta_i|\right)\right) \\ &= \frac{1}{(2b)^p(2\pi)^{n/2}\sigma^n} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 - \frac{1}{b}\sum_{j=1}^{p}|\beta_j|\right\} \end{aligned}$$

The mode of this distribution are the values of β_i that make this probability density as large as possible or the negative of the argument in the exponential as small as possible. This is equivalent to the lasso estimate.

Part (d): In this case the prior density of β_i changes and we have

$$p(\beta) = \frac{1}{(2\pi)^{p/2}\sigma^p} \exp\left\{ -\frac{1}{2} \sum_{j=1}^{p} \beta_j^2 \right\}.$$

Using this expression and following the same steps as in earlier parts gives the ridge regression estimate for β.

Applied Exercises

Exercise 8

See the R code chap_6_prob_8.R where this problem is worked.

The summary command on the output from the regsubsets output gives the following

```
Selection Algorithm: exhaustive
          X   X2  X3  X4  X5  X6  X7  X8  X9  X10
1  ( 1 )  " " " " "*" " " " " " " " " " " " " " "
2  ( 1 )  "*" " " "*" " " " " " " " " " " " " " "
3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
4  ( 1 )  " " "*" "*" " " "*" " " "*" " " " " " "
5  ( 1 )  " " "*" "*" "*" "*" " " " " " " "*" " "
6  ( 1 )  " " "*" "*" " " "*" "*" " " "*" "*" " "
7  ( 1 )  "*" "*" "*" " " "*" "*" " " "*" "*" " "
8  ( 1 )  " " "*" "*" " " "*" "*" "*" "*" "*" "*"
9  ( 1 )  "*" "*" "*" " " "*" "*" "*" "*" "*" "*"
10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

which shows the sequence of models selected for each value of $k \in \{1, \dots, 10\}$. If we plot the C_p, *BIC*, and adjusted R^2 for the *in-sample* training data we get the first three plots given in Figure 10. Note that these metrics selects $k = 2$, $k = 2$, and $k = 6$ as the optimal number of predictors to use. From the output of the summary command on the output of regsubsets the three predictors selected when $k = 2$ would be X, and X^3 which is close but incorrect given the model used to generate the data

$$Y = 1 - 0.1X + 0.05X^2 + 0.75X^3 + \epsilon.$$

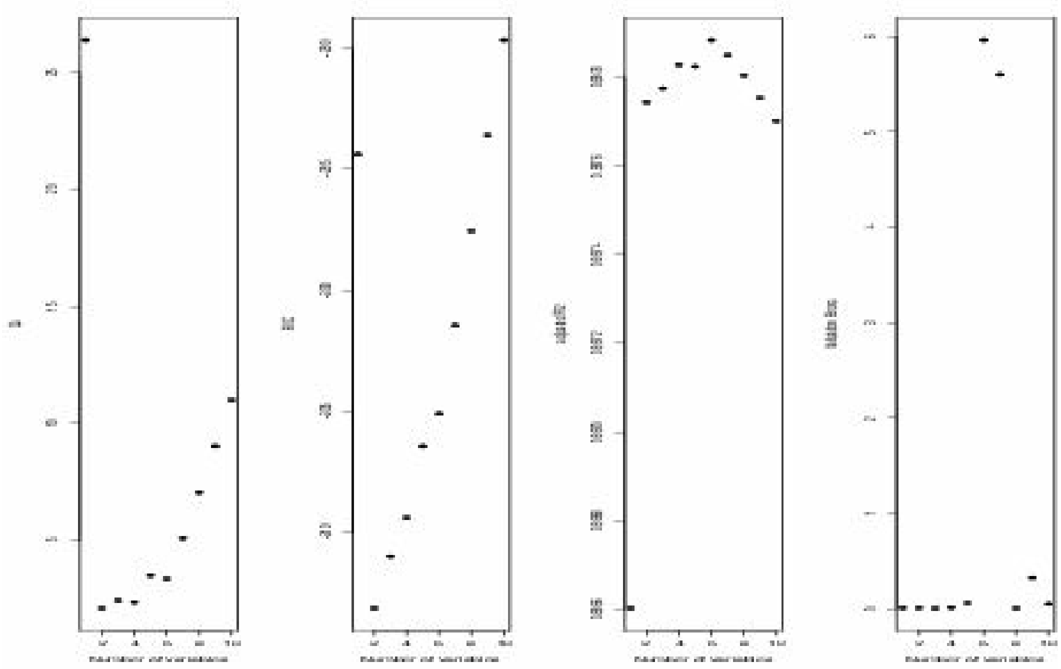

Figure 10: Best subset selection for Problem 8. From left to right as a function of the number of variables in the model: plots of C_p, *BIC*, the adjusted R^2 for the in-sample data and the MSE validation error.

To see how these in-sample results will hold up out-of-sample we compute the MSE on the validation set in the right-most plot in Figure 10. The lowest validation MSE is when *k* = 3 but the validation MSE error for *k* ≤ 4 are all very similar. If we take the model with the lowest validation error we would get coefficients give by

```
(Intercept)          X          X2          X3
 1.01506706 -0.11855671  0.02753875  0.76721123
```

These values match well with the true coefficients.

Part (b): We will now present the same analysis but using forward stepwise selection instead of best subset selection. The summary command on the forward selection results give

```
Selection Algorithm: forward
          X   X2  X3  X4  X5  X6  X7  X8  X9  X10
1  ( 1 )  " " " " "*" " " " " " " " " " " " " " "
2  ( 1 )  "*" " " "*" " " " " " " " " " " " " " "
3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
4  ( 1 )  "*" "*" "*" "*" " " " " " " " " " " " "
5  ( 1 )  "*" "*" "*" "*" "*" " " " " " " " " " "
```

```
6  ( 1 )  "*" "*" "*" "*" "*" " " " " " " "*" " "
7  ( 1 )  "*" "*" "*" "*" "*" " " " " "*" "*" " "
8  ( 1 )  "*" "*" "*" "*" "*" "*" " " "*" "*" " "
9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" "*" " "
10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

Plots of the in-sample values of C_p, BIC, and the adjusted R^2 for forward selection are given in the first three plots in Figure 11. There using these three in-sample metrics we see that the best subset is of size k = 2, k = 2, and k = 7. A validation data set suggests that k = 3 is the optimal model with same coefficients found earlier.

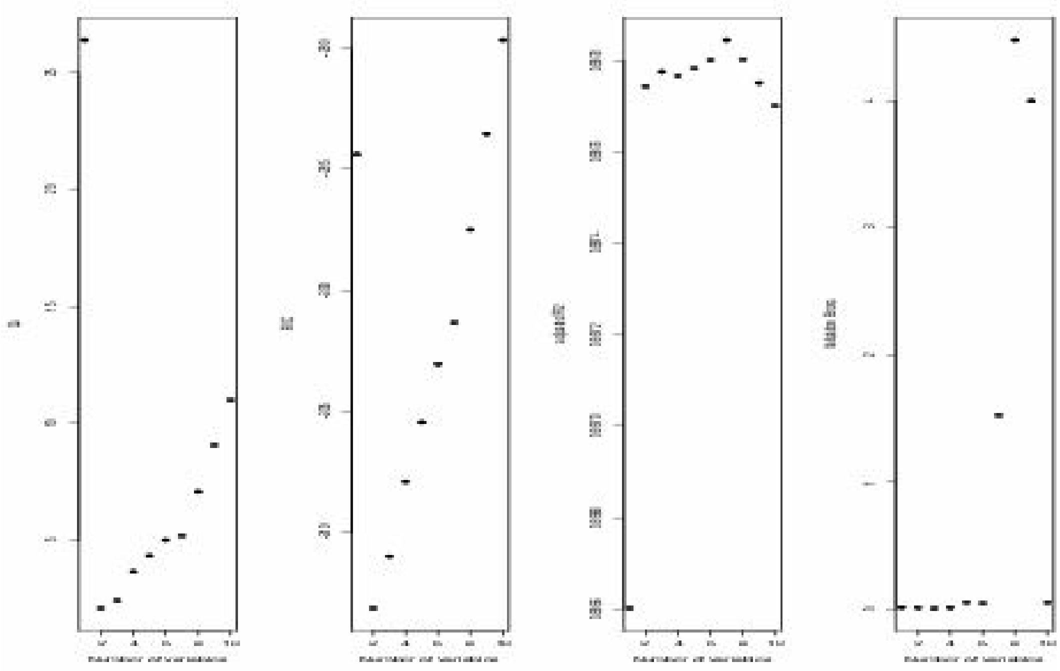

Figure 11: Forward selection for Problem 8. From left to right as a function of the number of variables in the model: plots of C_p, BIC, the adjusted R^2 for the in-sample data and the MSE validation error.

Following the same procedure for backwards selection we get the plots shown in Figure 12. We find suggested optimal model sizes of k = 4, k = 3, and k = 7. The validation set again selects for the optimal model a subset of size k = 3.

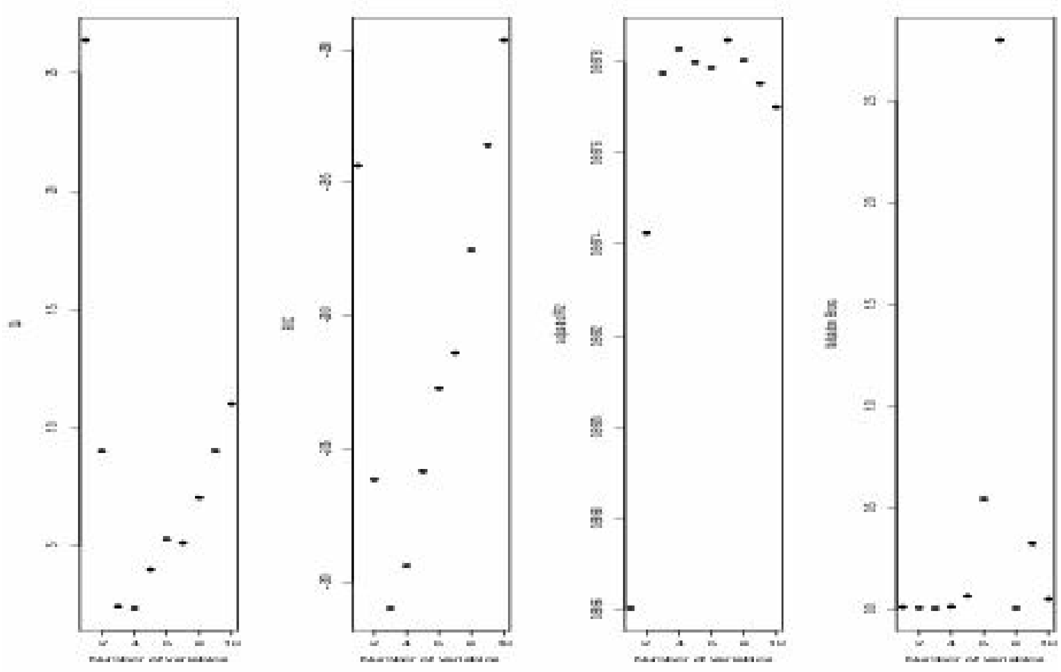

Figure 12: Plots of C_p, *BIC*, and the adjusted R^2 for the in-sample data for Problem 8 when using backward selection. The MSE validation on the validation set as a function of the number of variables.

One thing to note is that the C_p and *BIC* metrics (when using the in-sample) for all of the selection techniques data gave values of *k* that were quite close to optimal (i.e. $k \approx 3$). This is in contract to using the adjusted R^2 metric which gave values of *k* that were quite far from optimal in many cases $k \approx 7$. All of these metrics were inferior to the validation data set approach which found the correct model each time.

Part (c): We next use the lasso procedure to fit the given training data set. Using the cross validation method provided in the glmnet package we get Figure 13. From the output of cv.out we see that the one standard error optimal value of λ is given by 0.03186245. With this optimal value of λ the coefficients that accompany this model are given by

```
(Intercept) 1.026020037
X           .
X2          0.004179690
X3          0.664868098
X4          0.003606719
X5          0.011344145
X6          .
X7          .
```

```
X8          .
X9          .
X10         .
```

Notice that only the β_0 and β_3 coefficients are estimated with any degree of accuracy. Only by increasing the data set size (via $n \approx 100000$) do we start to have the lasso correctly estimate all coefficients.

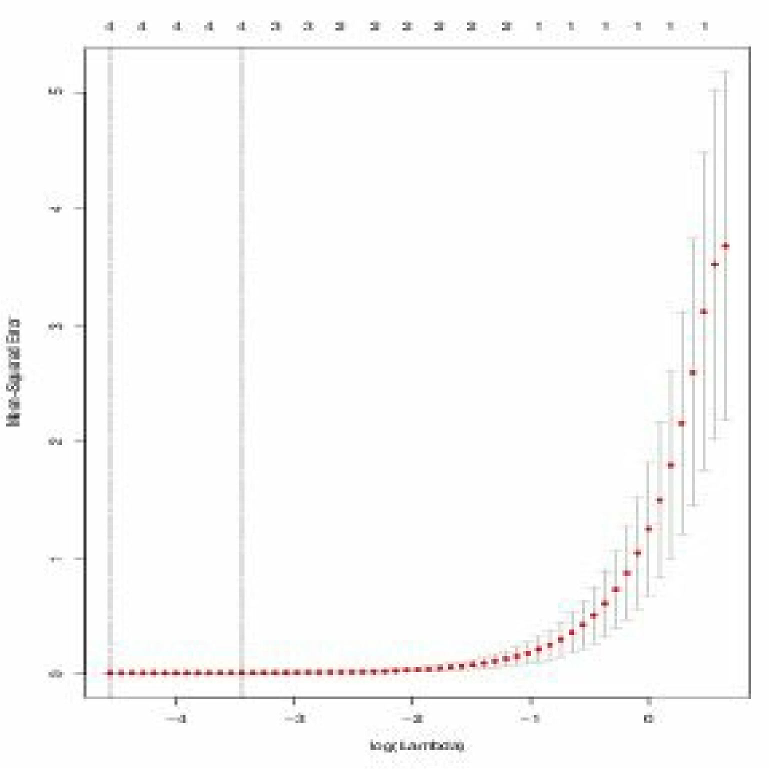

Figure 13: Plots of cross-validated MSE errors for various values of log(λ).

Part (f): For this part we will repeat much of the analysis performed in the preceding parts but for response $Y = 1.0 + 2.5X^7 + \epsilon$. We will compare the results obtained when we fit our data using best subset selection and the lasso. Notice that each of these methods ends with a set of k selected predictors from among the possible $X, X^2, \ldots, X^{10}$ but the lasso does a more restrictive search in model space and avoids doing the exponential amount of work required when one fits all possible models. For each method as a summary we present

- the final polynomial model selected (power and coefficients values)
- and the expected error rate under the selected model

For best subset selection we will use the procedure regsubsets on a training set to select the models used for each value of *k* and then use a validation data set to select (and return performance statistics) for the final (and hopefully optimal) model. For the lasso method we will use the cross validation method to return both the optimal model size, its coefficients, and the expected MSE error.

- For best subset selection the validation error is smallest for $k = 1$ (a single predictor), the selected feature is X^7 and the model coefficients are

  ```
  (Intercept)         X7
    0.9840339   2.5000261
  ```

  Notice that these are very close to the true values of β_0 and β_7. The MSE prediction error is 0.01151953 a relatively small number.

- For the lasso the optimal value of λ selected was 39.03716 which gives model coefficients

  ```
  (Intercept) 4.9082683
  X           .
  X2          .
  X3          .
  X4          .
  X5          .
  X6          0.2193258
  X7          2.3360006
  X8          .
  X9          .
  X10         .
  ```

  The estimate of β_0 is not very accurate but the estimate of β_7 is more correctly estimated. The predicted MSE error in this case is given by 1528.524.

Notice the difference in magnitude between the lasso predicted MSE error and the one found by best subset selection. I believe this difference is due to the fact that large powers of *X* can produce very large numbers and so a model that has even a small coefficients amount of a large power monomial X^p can produced an estimated $\hat{Y}$ that is far from the true *Y* and result in a large MSE.

Exercise 9

See the R code chap_6_prob_9.R where this problem is worked.

When we run that code we get the following (partial) output.

```
[1] "Linear model test MSE= 1615966.966"
[1] "Ridge regression test MSE= 3238522.483"
[1] "Lasso regression test MSE= 2018489.732"
[1] "PCR (with ncomp=    17) test MSE= 1615966.966"
[1] "PLS (with ncomp=    10) test MSE= 1601425.747"
```

Notice that when using principal component regression (PCR) cross-validation suggested we use the full model which gave a MSE equal to that of least squares. For this application the smallest MSE on the test data when we use partial least squares.

Exercise 10 (increasing the number of features)

See the R code chap_6_prob_10.R where this problem is worked.

Part (a): For this problem I choose to take one-half of the entries in β exactly equal to zero. This hopefully will make the results from the best subset selection easier to verify/validate. The exact numerical value for the true β was taken to be (zeros are represented with a dot)

```
[1] "True values for beta (beta_0-beta_20):"
 [1]  1.262954285   .           .           .           0.414641434
 [6] -1.539950042   .           .          -0.005767173  2.404653389
[11]  .             .          -1.147657009 -0.289461574  .
[16] -0.411510833   .          -0.891921127  0.435683299  .
[21] -0.224267885
```

Part (b-c): We use the R command regsubsets and then extract the optimal subsets of each size k. We can then evaluate the MSE on both the training and testing data sets as a function of the number of predictors. When we do that we get the plot given in Figure 14. There we see that as a function of the number of predictors k, the training MSE steadily decreases. As a function of k the testing MSE decreases initially and then stays flat for most subset sizes larger than ten.

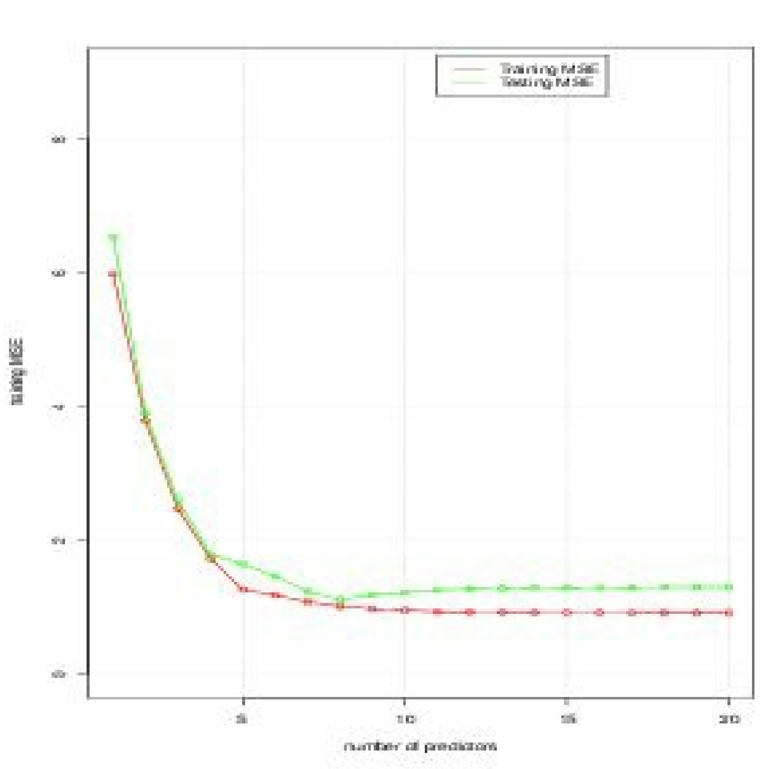

Figure 14: Plots of training (red) and testing (green) MSE errors as a function of the number of predictors.

The value of *k* that gives the smallest testing MSE is *k* = 8 and has a set of coefficients given by

```
(Intercept)         X4          X5          X9         X12         X13 
  1.3210455   0.4132288  -1.5880109   2.4639393  -1.1455077  -0.2815162 
        X15         X17         X18 
 -0.3589236  -1.0286318   0.6142240 
```

We notice a few things

- Best subset selection with a validation set did not find the optimal subset size *k* = 10 but the MSE for the model found and the model with *k* = 10 have about the same numerical values indicating the predictive accuracy of the model selected is close to the more optimal *k* = 10 model.
- When the true value of β_i is “large” the estimated $\hat{\beta}_i$ value is close to the true values. For example the estimates for β_0, β_4, β_5 etc. are close to the true values.
- When the true value of β_i is “small” best subsets might not include that predictor (at all) as a feature.

- The “errors” made by best subset selection (when it includes a predictor that is not really in the model for example) can be made worse if the random predictors chosen to fill the data matrix *X* are more correlated, since in general that makes the estimated value of $\hat{\beta}_i$ more noisy.

Part (g): In Figure 15 we make the suggested plot. Notice the dip in error values for *k* = 8 and the steady increase in error in the estimate of *β* as the subset size increases.

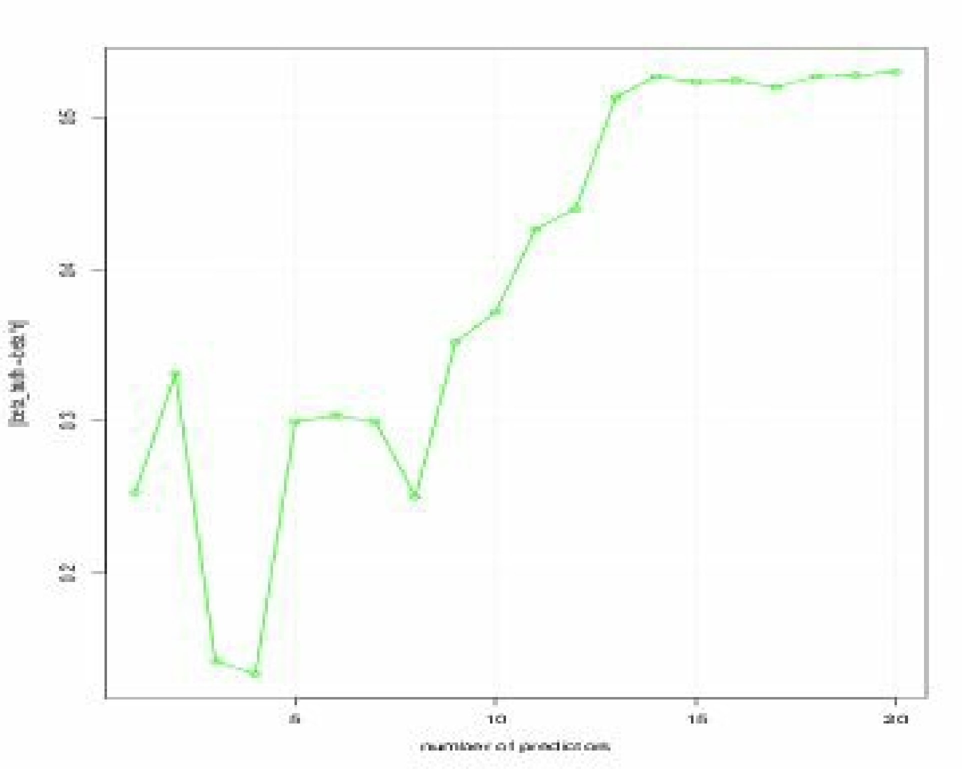

Figure 15: Plots of $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^k)^2}$ as a function of *k* the subset size.

Exercise 11

See the R code chap_6_prob_11.R where this problem is worked. In that R code we use the following methods to predict crim: linear regression using all features, ridge regression, the lasso, principle component regression, and partial least squares regression. In the R code we selected any parameters of the method using cross-validation.

When we run that code we get the following (partial) output.

```
[1] "Linear model test MSE=     34.996"
[1] "Ridge regression test MSE=     63.826"
[1] "Lasso regression test MSE=     63.572"
[1] "PCR (with ncomp=     3) test MSE=     40.049"
[1] "PLS (with ncomp=     5) test MSE=     35.258"
```

What is interesting in this example is that the linear model with no feature selection performed best on the test data set. The performance of PLS and PCR was very close to that of the linear model and used far fewer degrees of freedom. The lasso result was interesting in that it selected a model that used only the predictor rad: the index of accessibility to radial highways.

Chapter 7 (Moving Beyond Linearity)

Conceptual Exercises

Exercise 1

Part (a): If $x \leq \xi$ then $f(x)$ is given by

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3.$$

To equal $f_1(x)$ on this domain we need to have

$$a_1 = \beta_0, \quad b_1 = \beta_1, \quad c_1 = \beta_2, \quad \text{and} \quad d_1 = \beta_3. \tag{13}$$

Part (b): If $x > \xi$ then $f(x)$ is given by

$$\begin{aligned} f(x) &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \xi)^3 \\ &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3) \\ &= (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2) x + (\beta_2 - 3\beta_4 \xi) x^2 + (\beta_3 + \beta_4) x^3 \; . \end{aligned}$$

Thus to have this equal $f_2(x)$ we need to have

$$a_2 = \beta_0 - \beta_4 \xi^3, \quad b_2 = \beta_1 + 3\beta_4 \xi^2, \quad c_2 = \beta_2 - 3\beta_4 \xi, \quad \text{and} \quad d_2 = \beta_3 + \beta_4. \tag{14}$$

Part (c): If $x = \xi$ then based on Equation 13 we have

$$f_1(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3 \tag{15}$$

and based on Equation 14 we have

$$\begin{aligned} f_2(\xi) &= (\beta_0 - \beta_4\xi^3) + (\beta_1 + 3\beta_4\xi^2)\xi + (\beta_2 - 3\beta_4\xi)\xi^2 + (\beta_3 + \beta_4)\xi^3 \\ &= \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 , \end{aligned}$$

when we expand and simplify. Note that $f_1(\xi) = f_2(\xi)$ as it should.

Part (d): Now the first derivative of f_1 and f_2 are given by

$$\begin{aligned} f_1'(x) &= b_1 + 2c_1x + 3d_1x^2 \\ f_2'(x) &= b_2 + 2c_2x + 3d_2x^2 . \end{aligned}$$

Replacing b_i, c_i, and d_i with their expressions in terms of β_i we get

$$\begin{aligned} f_1'(\xi) &= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 \\ f_2'(\xi) &= (\beta_1 + 3\beta_4\xi^2) + 2(\beta_2 - 3\beta_4\xi)\xi + 3(\beta_3 + \beta_4)\xi^2 = \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 . \end{aligned}$$

Thus we see $f'_1(\xi) = f'_2(\xi)$ as it should.

Part (e): Now the second derivative of f_1 and f_2 are given by

$$\begin{aligned} f_1''(x) &= 2c_1 + 6d_1 \\ f_2''(x) &= 2c_2 + 6d_2 . \end{aligned}$$

Replacing b_i, c_i, and d_i with their expressions in terms of β_i we get

$$\begin{aligned} f_1''(\xi) &= 2\beta_2 + 6\beta_3\xi \\ f_2''(\xi) &= 2(\beta_2 - 3\beta_4\xi) + 6(\beta_3 + \beta_4)\xi = 2\beta_2 + 6\beta_3\xi , \end{aligned}$$

so $f_1''(\xi) = f_2''(\xi)$.

We have shown that the expression for $f(x)$ is a piecewise cubic polynomial with the two cubic polynomials on the left and right of the point $x = \xi$. These two polynomials at $x = \xi$ are continuous, have continuous first derivatives, and have continuous second derivatives.

Exercise 2

Part (a): As $\lambda = \infty$ to optimize we will try to make the expression $\int g^{(0)}(x)^2dx$ as small as possible (i.e. zero). In order to do this we need $g = 0$ i.e. it is the zero function.

Part (b): With $\lambda = \infty$ to optimize we will try to make the expression $\int g^{(1)}(x)^2dx$ as small as possible (i.e. zero). In order to do this $g(x)$ will need to be a constant $g(x) = \beta_0$.

Part (c): With $\lambda = \infty$ we want to make $\int g^{(2)}(x)^2dx$ as small as possible so $g^{(2)}(x) = 0$ and $g(x)$ is a linear function i.e. $g(x) = \beta_0 + \beta_1x$.

Part (d): As in the previous parts, we must have $g^{(3)}(x) = 0$ so $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2$.

Part (e): If $\lambda = 0$ and $m = 3$ we want to pick g to minimize only $\sum_{i=1}^{n}(y_i - g(x_i))^2$ and so will select a $g(x)$ that interpolates y_i at each x_i as closely as possible.

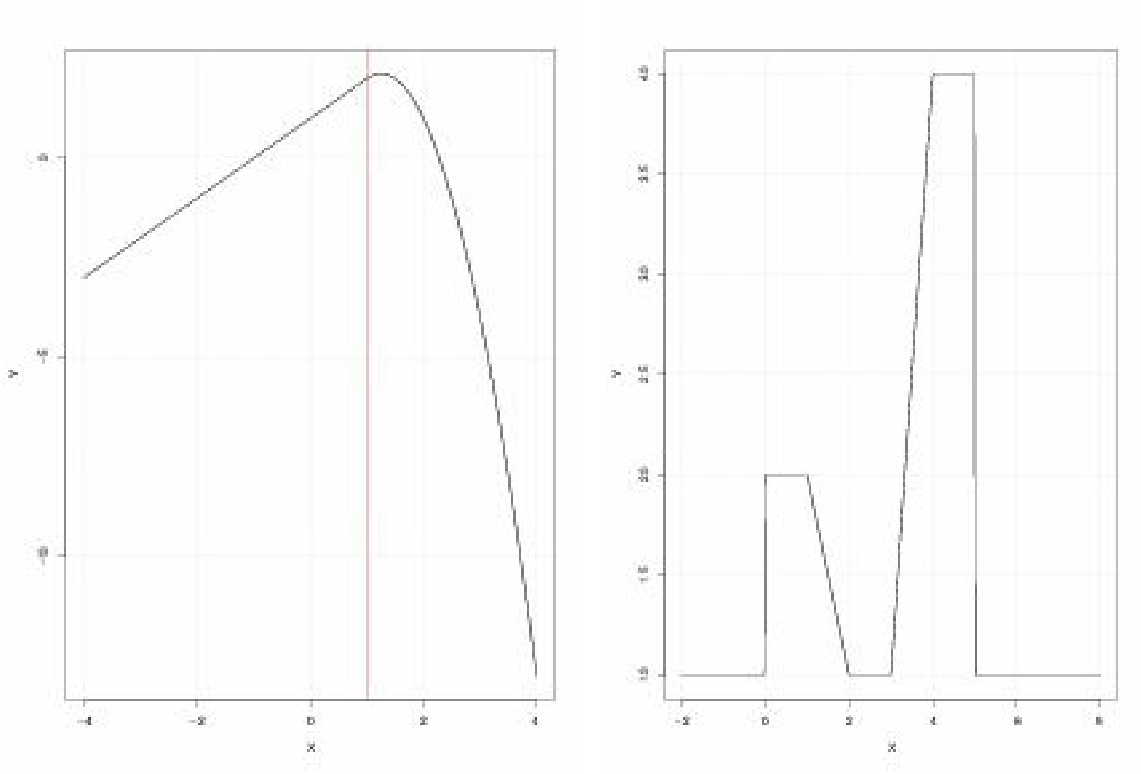

Figure 16: Left: The estimated curve for Y as a function of X for Exercise 3. **Right:** The estimated curve for Y as a function of X for Exercise 4.

Exercise 3

For the given values of β_i and functions $b_i(X)$ we have

$$
\begin{aligned}
Y &= \beta_0 + \beta_1 X + \beta_2 (X-1)^2 I(X \geq 1) \\
&= 1 + X - 2(X-1)^2 I(X \geq 1)\,.
\end{aligned}
$$

When we plot this function we get the plot given in Figure 16 (left). Notice that the quadratic expression can be seen as curvature in the plot when $X \geq 1$.

Exercise 4

For the given values of β_i and functions $b_i(X)$ we have

$$
\begin{aligned}
Y &= 1 + 1(I(0 \le X \le 2) - (X-1)I(1 \le X \le 2)) \\
&\quad + 3((X-3)I(3 \le X \le 4) + I(4 \le X \le 5)) \,.
\end{aligned}
$$

When we plot this function we get the plot given in Figure 16 (right).

Exercise 5

Part (a): Note that as $\lambda \to \infty$ the minimization procedure will try to make the regularization terms (the integral terms with a λ in them) as small as possible eventually requiring $g^{(3)}(x) \to 0$, for $\hat{g}_1$ and $g^{(4)} \to 0$ for $\hat{g}_2$. As $g^{(3)}(x) \to 0$ we have $g(x)$ going to a second order polynomial and as $g^{(4)}(x) \to 0$ we have $g(x)$ going to a third order polynomial. Since a third order polynomial has more degrees of freedom than a second order polynomial it will be able to fit the data better and will have a smaller RSS. Thus $\hat{g}_2$ should have the smaller training RSS.

Part (b): One cannot answer this question without computing the performance of $\hat{g}_1$ and $\hat{g}_2$ on the test data set. If a person could answer this question in general there would be no reason to discuss the two techniques, we would always use the one that has the better test set error.

Part (c): When $\lambda = 0$ then $\hat{g}_1$ and $\hat{g}_2$ will be the same functions so they will have equal training and test RSS.

Applied Exercises

Exercise 6

See the R code chap_7_prob_6.R where this problem is worked.

Part (a): Note that for this part of the problem we could implement *k*-fold cross validation ourselves (i.e. “by hand” using lm) but it is easier to use the function cv.glm since this later function has the ability to perform cross validation for us.

Using the cv.glm function we obtain the plot given in Figure 17 (left). From that plot we see that the smallest value of the error estimate is given with a ten degree polynomial. Since this seems to be somewhat large (a tenth order polynomial is quite “wiggly”) and the cross-validation curve is rather flat for all degrees larger than two we should probably select a smaller degree polynomial. Based on this logic I choose $d = 4$. If we had access to the standard errors from the cross-validation results we could perhaps use the one-standard error rule to algorithmically select a polynomial degree.

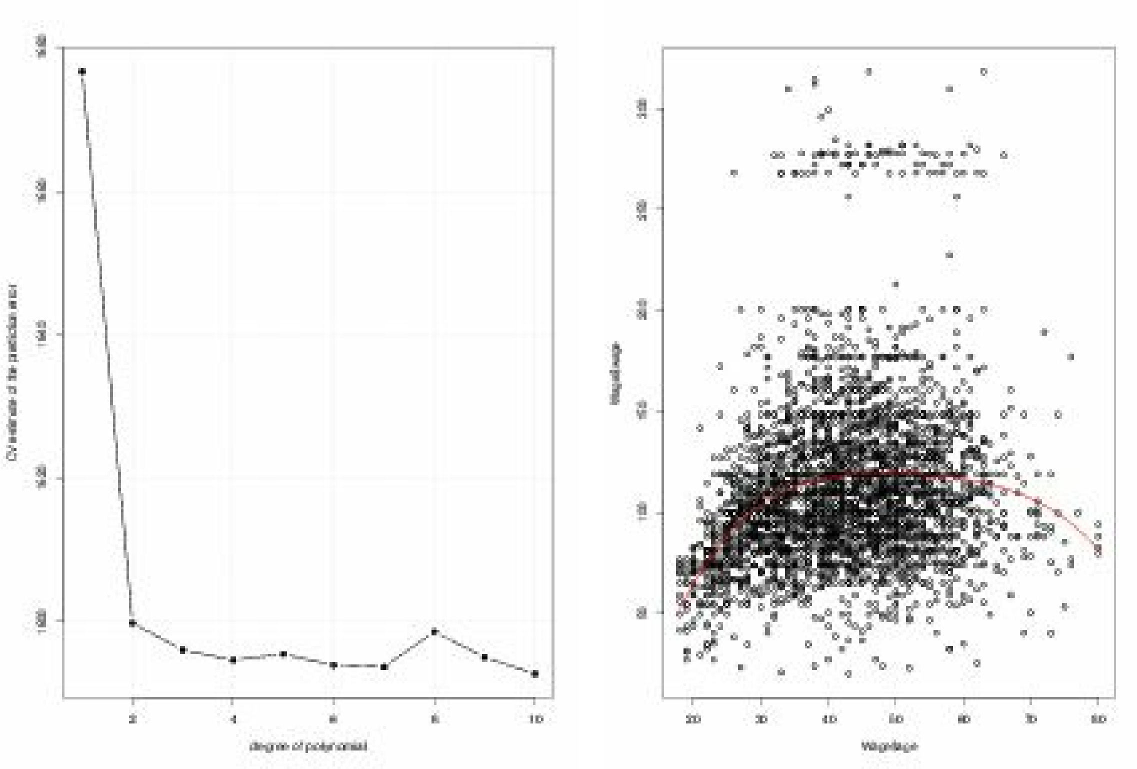

Figure 17: Left: The cross-validated error curve. **Right:** The data and a fourth order polynomial fit.

We can then build a model using all of the data and a fourth order polynomial in age to predict wage. Fitting the model the summary command on the output gives

```
Call: glm(formula = wage ~ poly(age, me), data = Wage)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     111.7036     0.7287 153.283  < 2e-16 ***
poly(age, me)1  447.0679    39.9148  11.201  < 2e-16 ***
poly(age, me)2 -478.3158    39.9148 -11.983  < 2e-16 ***
poly(age, me)3  125.5217    39.9148   3.145  0.00168 **
poly(age, me)4  -77.9112    39.9148  -1.952  0.05104 .
```

Notice that the fourth coefficient is not that significant. This indicates that perhaps a third order polynomial would perhaps be a better choice (more on this later). Next we plot the non-linear model just obtained with the data to visually observe how well the model fits. When we do that we get the plot given in Figure 17 (right). In that plot we see that the polynomial fit enables the model to curve downwards at the two ends of the domain.

We can use the anova command to compare polynomial models of various orders. We do this by building a sequence of nested models and then calling the anova command on them. The output of this command gives

```
> anova(m0,m1,m2,m3,m4,m5)
Analysis of Variance Table

Model 1: wage ~ 1
Model 2: wage ~ poly(age, 1)
Model 3: wage ~ poly(age, 2)
Model 4: wage ~ poly(age, 3)
Model 5: wage ~ poly(age, 4)
Model 6: wage ~ poly(age, 5)
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1   2999 5222086
2   2998 5022216  1    199870 125.4443 < 2.2e-16 ***
3   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
4   2996 4777674  1     15756   9.8888  0.001679 **
5   2995 4771604  1      6070   3.8098  0.051046 .
6   2994 4770322  1      1283   0.8050  0.369682
```

From this command we can conclude that starting with constant fit adding a linear term and then a quadratic term decrease the resulting RSS significantly (note the three stars). Adding a cubic term reduces the RSS somewhat (note the two stars) and so on. Since the fourth order polynomial is only “better” than the third order polynomial at the 10% significance level we might decide to use a third order polynomial instead. This is the same conclusion obtained above.

Part (b): This is much the same as the previous part but now we will use R’s cut function to produce constant regions. The cross-validated MSE error curve (with standard errors) is given in Figure 18 (left). Here the *x*-axis corresponds to the number of bins the domain age is cut up into. The smallest MSE is given by using ten bins to discretize age. In the plot we also present the one-standard-errors of the MSE. Using the one-standard-error rule we would select the model which is simpler and which has an average MSE error value that is within one-standard-error from this smallest value. One-standard-error above the ten bin MSE result is drawn as a horizontal line in red. The one-standard-error rule would select a model that cut the domain into three bins. When we plot this model (with the data) we get the plot given in Figure 18 (right).

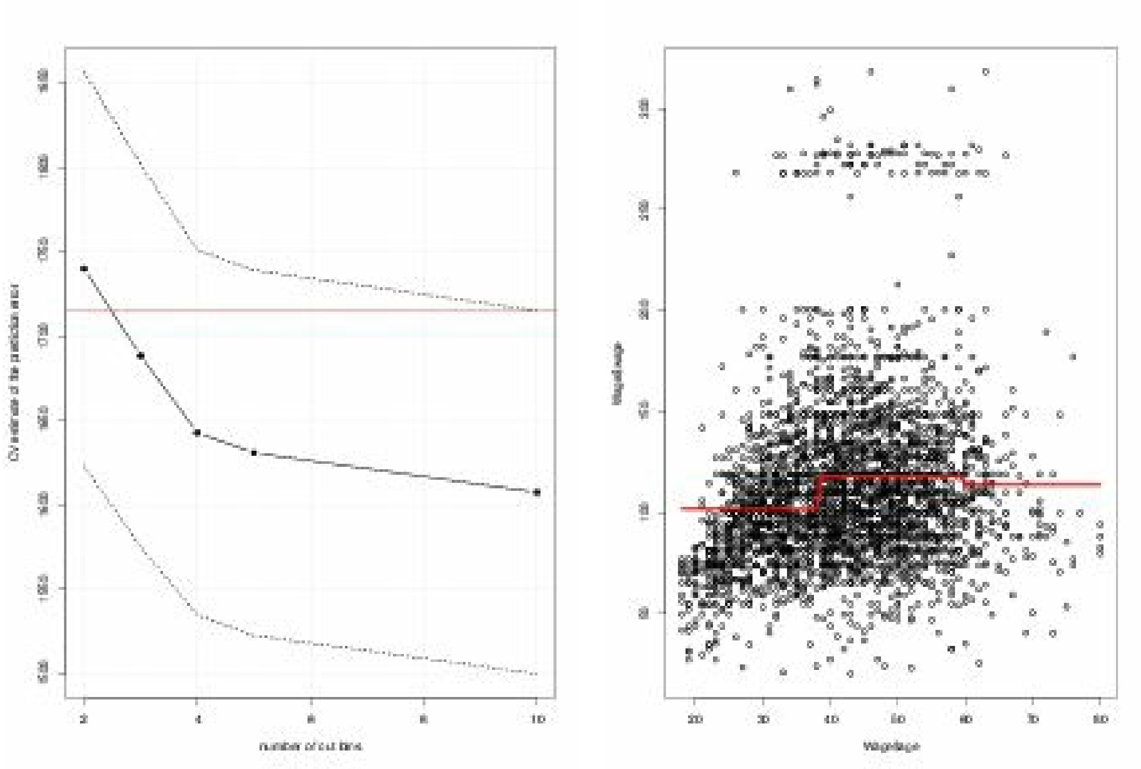

Figure 18: Left: The cross-validated error curve with standard errors. **Right:** The data and the optimal stepwise constant function selected using the one-standard-error rule.

Exercise 7-8

As these two problems are rather open ended I'll only outline how one would proceed and leave the details for the interested reader. For the Wage data set one could start by assuming a linear model that used all the predictors to predict wage. Some of these predictors will probably not be useful in predicting wage either due to their lack of usefulness or due to the inclusion of predictive features that mask their importance. The summary command on the initial linear model will help determine which predictors can be dropped from the analysis using something like a backwards stepwise selection procedure. In addition, a validation set (or cross-validation) can help select a more optimal (i.e. smaller) model with the most meaningful predictors. Once a linear model is built we can use residual plots (see [1]) to graphically assess if any of the predictors enter the model in a non-linear way. One type of residual plot is a scatter plot of the residuals (or standardized residuals) for the vertical axis and the model predictions $\hat{y}$ on the horizontal axis (often with an overlaying smoothing like from a lowess curve). Any observed curvature in this scatter plot represents a non-linear relationship between Y and at least one of the predictors X_i that could be modeled. Another type of residual plot presents the residuals on the vertical axis and a single predictor X_i on the horizontal axis and again we look for curvature. The feature with the most curvature in the residual plots could then be modeled using non-linear transformations like discussed in this chapter. This process could be repeated as many times as needed until all statistically significant curvature has been modeled.

Exercise 9

See the R code chap_7_prob_9.R where this problem is worked.

Part (a): See the plot given in Figure 19 (left). The data is fit quite well with this third order polynomial as the plot and the output of the summary command shows:

```
Call: lm(formula = nox ~ poly(dis, 3), data = Boston)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared: 0.7148,     Adjusted R-squared: 0.7131
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

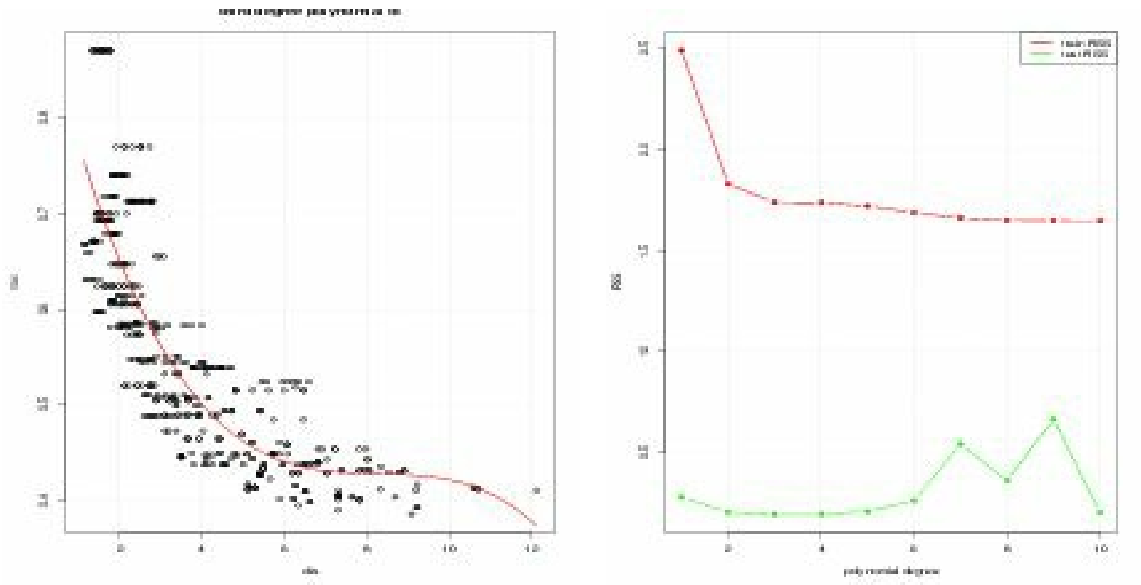

Figure 19: Left: A third degree polynomial fit (in red) of nox using the feature dis from the Boston dataset. **Right:** Cross-validation to suggest the degree of polynomial to use in predicting nox from dis.

Part (b-c): See the plot given in Figure 19 (right) where we perform cross-validation on the degree of the polynomial to use in predicting nox from dis. The training RSS steadily decreases as we add more degrees of freedom (the polynomial degree increases). The testing RSS decreases initially (to a low at a degree of around three) and then begins to increase again as the degree of the polynomial gets larger. From this plot a degree three polynomial seems to be optimal.

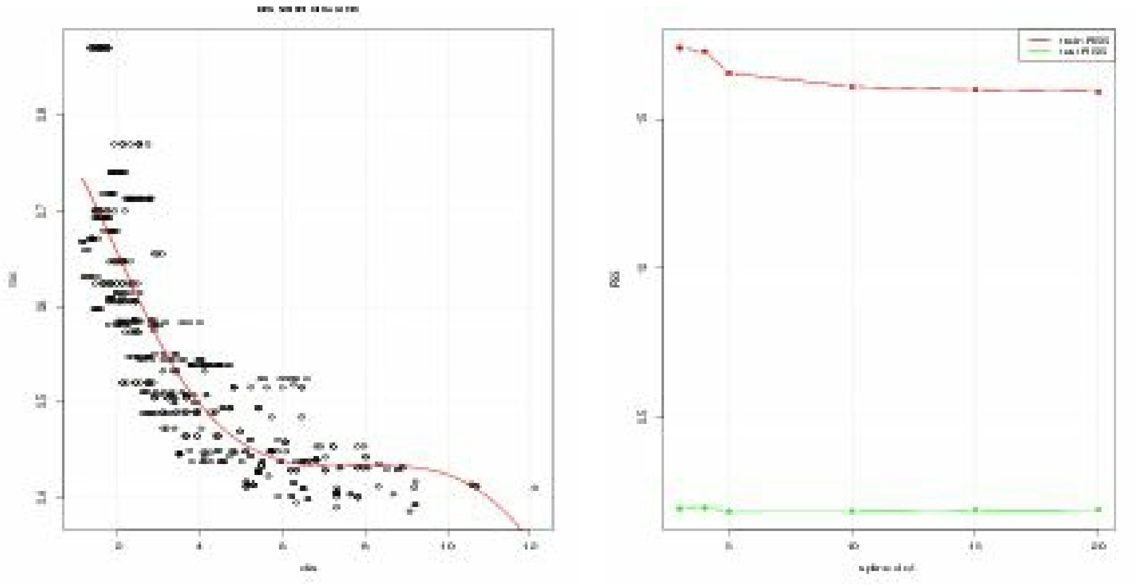

Figure 20: Left: A spline fit with four degrees of freedom to predict nox using the feature dis from the Boston dataset. **Right:** Cross-validation to suggest the spline degree of freedom to use in predicting nox from dis.

Part (d-f): In using the bs function I choose to specify the splines degree of freedom via cross-validation. This will place knots at the percentile points of the data with a large degrees of freedom corresponding to more knots in the domain. In Figure 20 (left) I plot the data along with a spline with the requested four degrees of freedom. In Figure 20 (right) I plot the cross-validate training and testing RSS as a function of the number of degrees of freedom. The test RSS seems to be very flat for many of the degrees of freedom tested. If one had to select a minimum of the testing RSS it might be around the value of five.

Exercise 10

See the R code chap_6_prob_10.R where this problem is worked.

Part (a): We first split the given data into three parts: a training set, a validation set, and a test set. We then apply the R command regsubsets on the training set to get the best set of predictors for subsets of every size 1 ≤ k ≤ p. To determine which subset size k is the optimal we then compute for each subset size the validation set mean square error. When I do this I'm getting the plot given in Figure 21 (left). The best subset size on the validation dataset seems to be k = 9. On the test set this model gives a MSE of 4427377. Note that we could "avoid" the need for a validation set by using the metrics: adjusted R^2, BIC, or AIC as discussed in Chapter 6 to select the optimal subset size k. Experience with some of the problems in Chapter 6 has shown that using a validation set will generally result in better performance than these other metrics. Because the next part of this problem will fit a GAM on the features found to be most important in forward stepwise selection to make our lives easier we'll consider the case where the subset size is k = 3 or only three predictors are selected as important in predicting Outstate. In that case forward stepwise selection selects the predictors (and coefficient estimates) to be

```
 (Intercept)   PrivateYes   Room.Board      Expend
-588.3851111 3053.1751313    1.2124240    0.3568496
```

This model has a MSE given by 5097180.

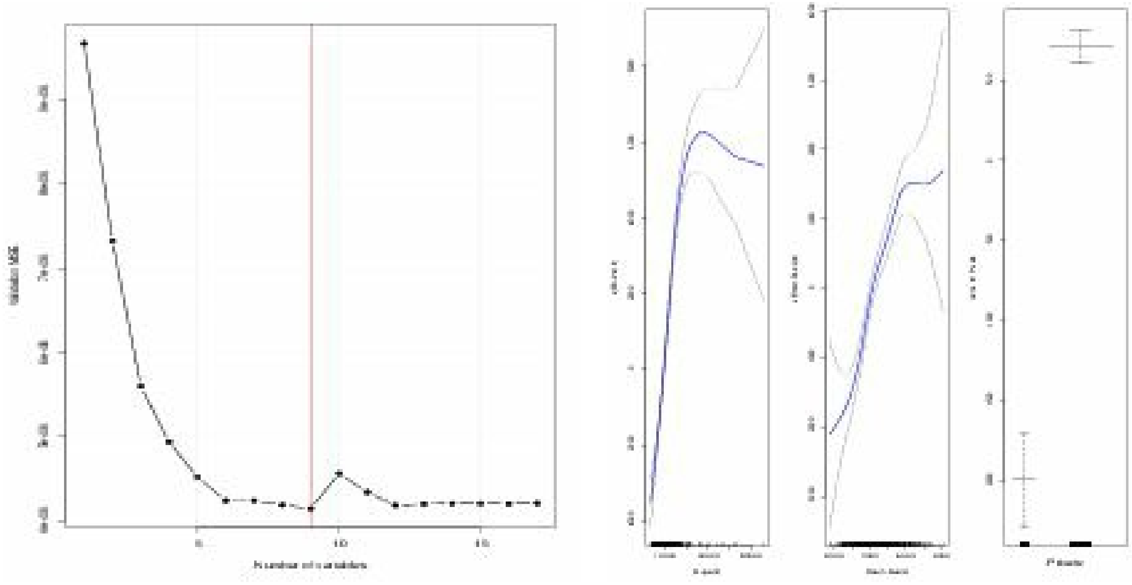

Figure 21: Left: Validation MSE for each subset of size k ≤ p selected using forward selection. The subset size k with the smallest MSE is located with a vertical red line. **Right:** The GAM plot output.

Part (b): Next we fit a GAM to model Outstate using the three predictors found above. A plot of the output from the plot command applied to the gam output is given in Figure 21 (right). We see that both Expend and Room.Board seem to be non-linear (especially Expend). The MSE on the test data when using this model is 4570160 which is smaller than the MSE on the test data set for the optimal $k = 3$ subset. This might be an indication that the GAM is helping improve prediction accuracy.

Exercise 11

See the R code chap_7_prob_11.R where this problem is worked.

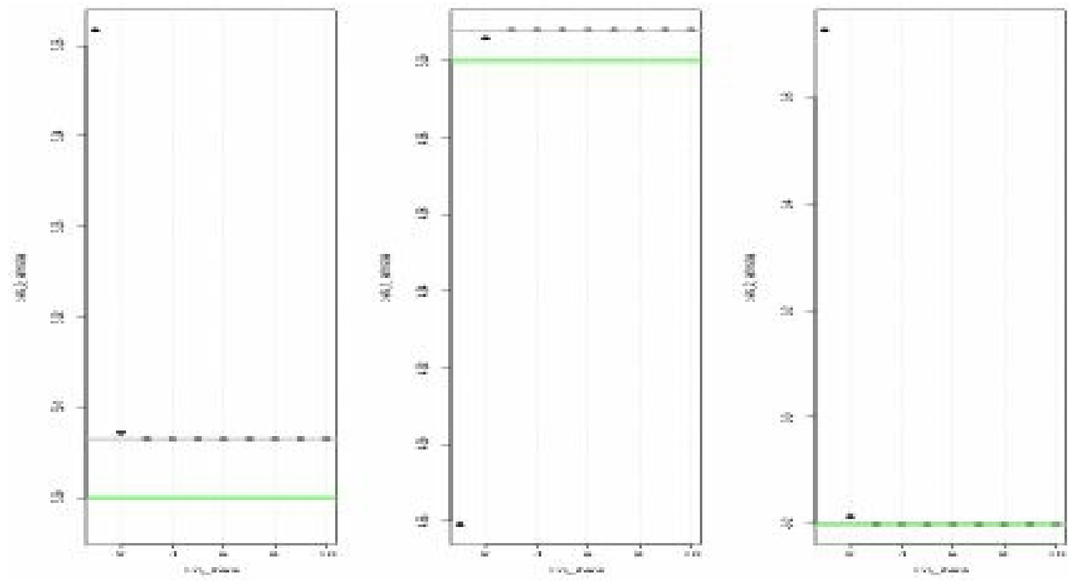

Figure 22: Results from back-fitting when $p = 2$ for Exercise 11 in estimating the coefficients β_0, β_1, and β_2. The green horizontal lines correspond to the true values of β_i and the gray horizontal line is the estimate of β_i obtained from the normal equations.

See the plot in Figure 22 (left) for a plot of the iterations of the back-fitting procedure on the two dimensional data suggested. For each coefficient β_0, β_1, and β_2 the back-fitting iterations are plotted as black dots, the true values of β_i are plotted as a horizontal green line and the estimate of β_i obtained from using the lm command i.e. solving the normal equations are plotted as a horizontal gray line. We see that the back-fitting iterations converge the the *same* values as the solution of the normal equations. Notice that we only plot 50 iterations for after that point the estimate $\hat{\beta}_i$ don't perceptibly change from iteration to iteration.

Exercise 12

See the R code chap_7_prob_12.R where this problem is worked.

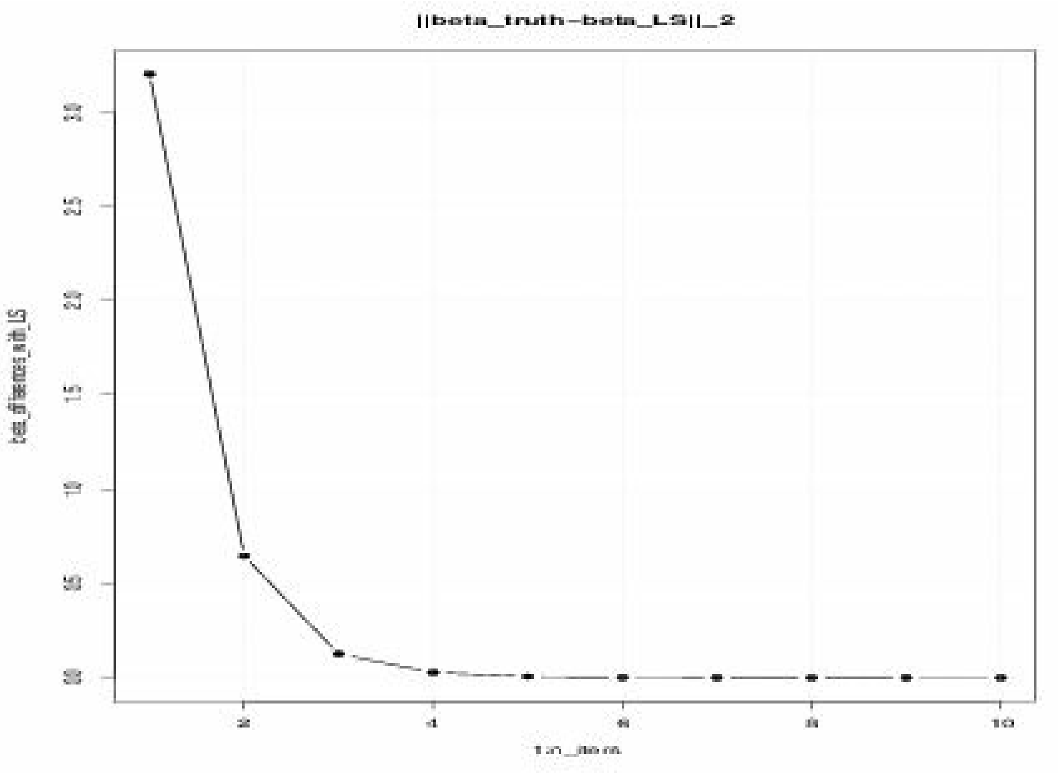

Figure 23: Results from back-fitting when *p* = 100 for Exercise 12. We plot the difference between the estimates of β_i given by the normal equations and the estimate given by back-fitting.

See the plot in Figure 23 for a plot of the iterations of the back-fitting procedure when *p* = 100. Notice that only around *ten* iterations are needed to converge to the coefficient estimates. These coefficient estimates are equal to the least squares ones (the same ones that are produced by solving the normal equations and output by the R command lm).

Chapter 8 (Tree-Based Methods)

Conceptual Exercises

Exercise 2

Studying algorithm 8.2 (Boosting for Regression Trees) when we are boosting stumps (*d* = 1 splits) each function $\hat{f}^{\,b}$ that we add to the total approximation $\hat{f}$ will only depend on *one* component of the input vector *X* since it involves the test $X_j < t$ for some threshold value *t*. Even if different components *j* are selected at each iteration when we have finished building the full model $\hat{f}$ we can separately sum all the functions $\hat{f}^{\,b}$ that involve the variable X_1, then separately sum all the $\hat{f}^{\,b}$ that involve the variable X_2 etc. Once we have done that we have written $\hat{f}$ that is the sum of functions f_j that depend only on the *j*-th component of *X* i.e. we have shown $\hat{f}$ is an additive model.

Exercise 3

See Figure 24 for the desired plot. Note that all of these error metrics are smallest when the class decision is more uniform (i.e. closer to 0 or 1).

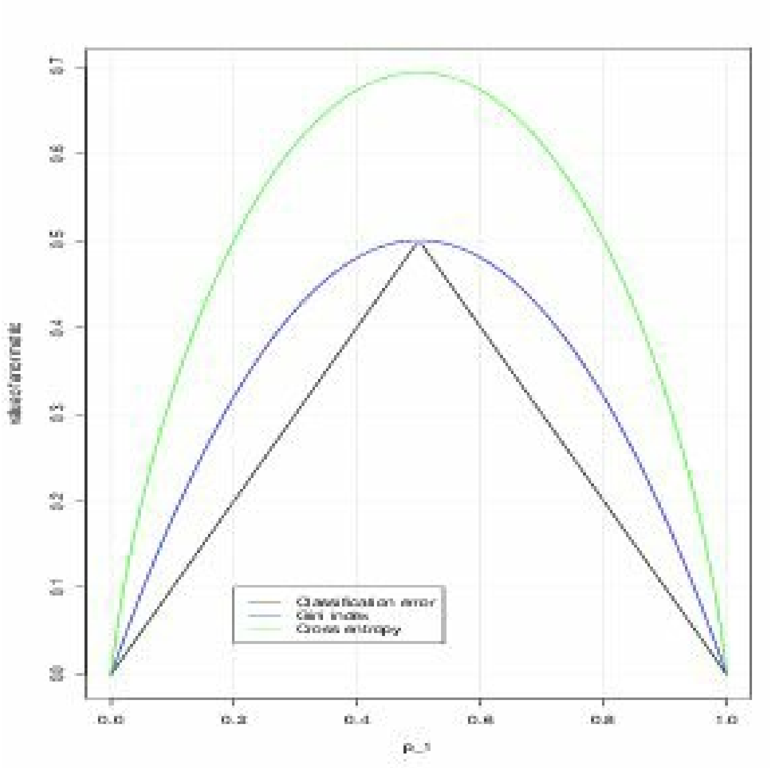

Figure 24: Various error metrics for classification as a function of p_{m1}.

Exercise 5

For the given probabilities we have four votes for class “not red” and six votes for class “red”. The majority voe would then classify this sample as “red”. An averaging approach would average the two classifications

```
p_red = mean( c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75) )
p_not_red = mean( 1-c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75) )
```

This gives 0.45 and 0.55 for the probability of “red” and “not red” respectively. Thus in the average case we would classify our sample as “not red”.

Applied Exercises

Exercise 7

See the R code chap_8_prob_7.R where this problem is worked.

For this problem we use the randomForest library with the Boston dataset to predict medv from the other features given in the dataset. The randomForest command has a mtry parameter which specifies the number of predictors to consider at each split as it grows each subsequent tree. We tested three values for mtry the values of *p*, $\frac{p}{2}$ and $\sqrt{p}$. For each of these values of mtry we generate many random forests with a different number of trees using the training set data. We then use these random forests to estimate the prediction error on the test dataset. Plots of the learning curves we get for the test data are given in Figure 25. Notice that all three settings give qualitatively the same behavior and that after the curves have reached their lowest level adding more trees does not further decrease the testing MSE. This last fact is the observation that it is relatively hard to overfit using random forests.

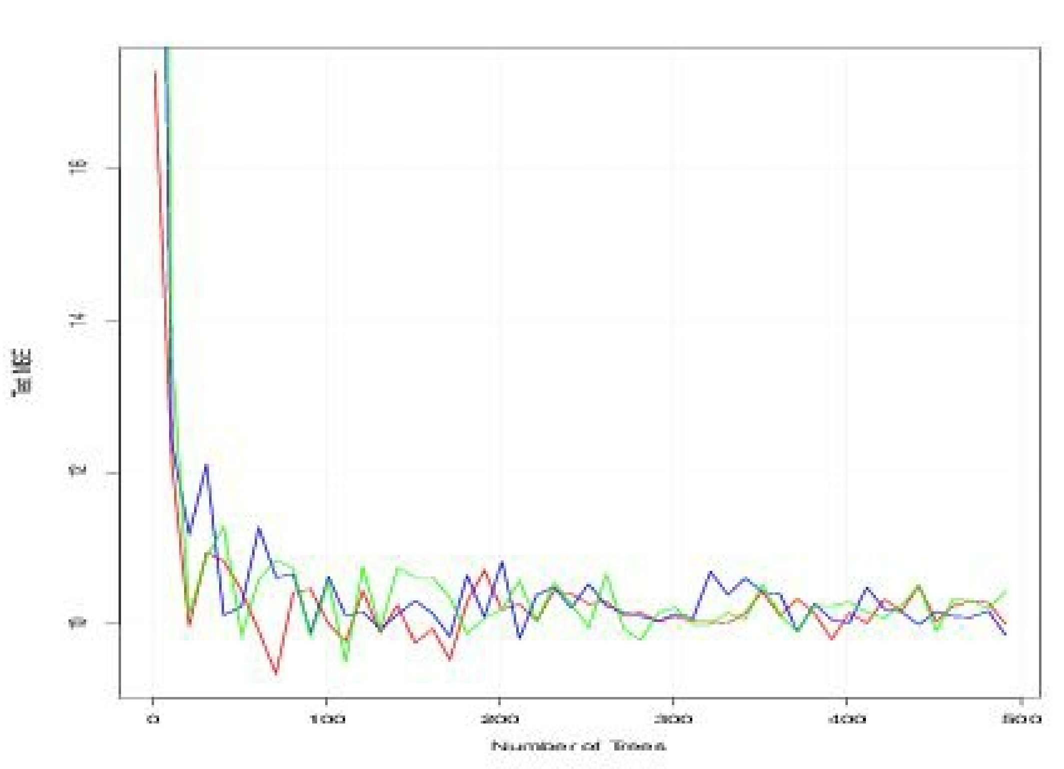

Figure 25: The test MSE for three different values for mtry and a number of trees.

Exercise 8

See the R code chap_8_prob_8.R where this problem is worked.

Part (a-b): When we plot the original tree (grown on the training data) we get Figure 26. Notice that when reading a tree like this the text label at each split represents denotes the samples that flow along the *left* branch as we go down the tree. Thus in Figure 26 our top split has the label ShelveLoc: Bad, Medium meaning that all samples that go down the left branch of the initial split have their ShelveLoc field set to either Bad or Medium and the samples that flow down the right branch of that split have their ShelveLoc variable set at Good. On this initial (unpruned) tree our MSE on the test dataset is 4.477452.

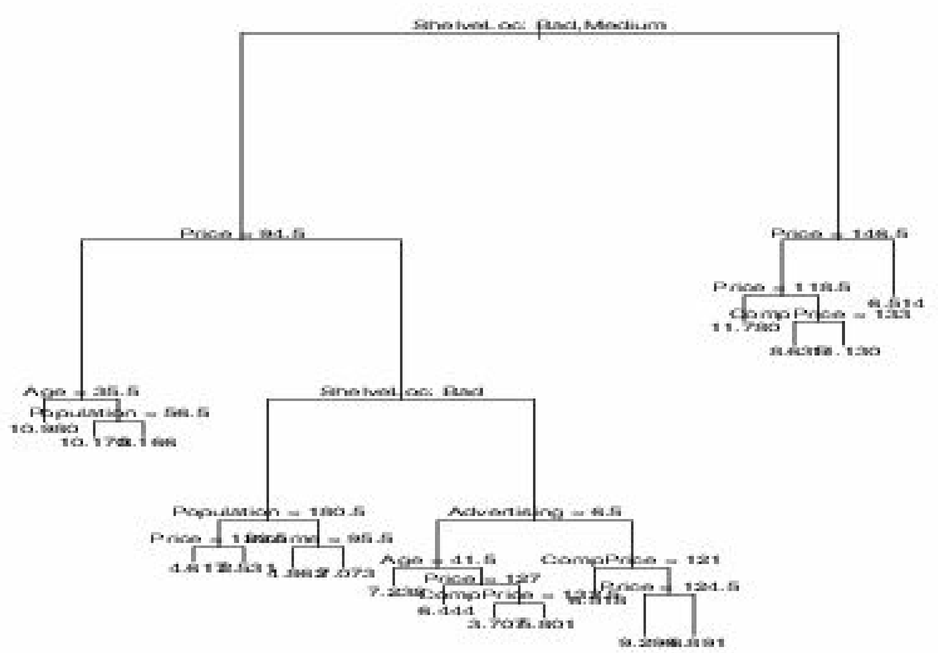

Figure 26: The initial tree built from the training data. Note that no pruning has been done on this tree.

Part (c): Using the prune.tree command and plotting the deviance as a function of tree size gives Figure 27 (left). In that plot it looks like the cross-validated deviance is smallest for the largest tree size. We note that the deviance is rather flat for all tree sizes larger than about six. To select the simplest model that performs near optimal we will then consider a tree with $k = 6$ terminal nodes. This tree is shown in Figure 27 (right). Based on how the splits are obtained the features involved in the top most splits are the most important in predicting Sales. For this problem we see that the top two features are ShelveLoc and Price.

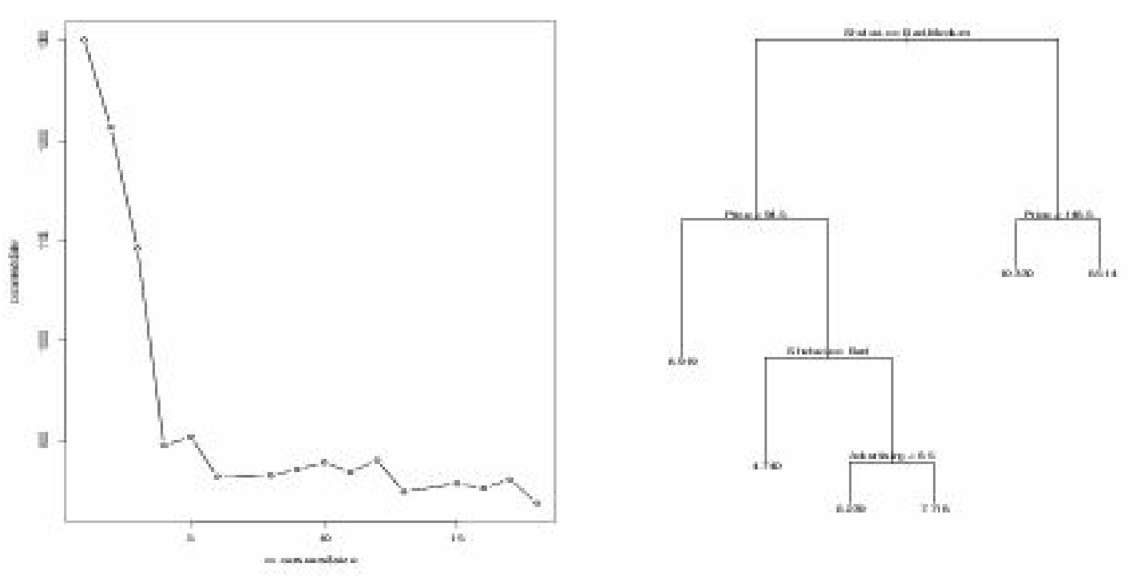

Figure 27: Left: Output from the prune.tree command. **Right:** The tree pruned to have *k* = 6 leaf nodes.

Using this pruned tree on the test dataset gives a MSE of 5.208115. This is slightly above what we got when we used the full tree on the test dataset.

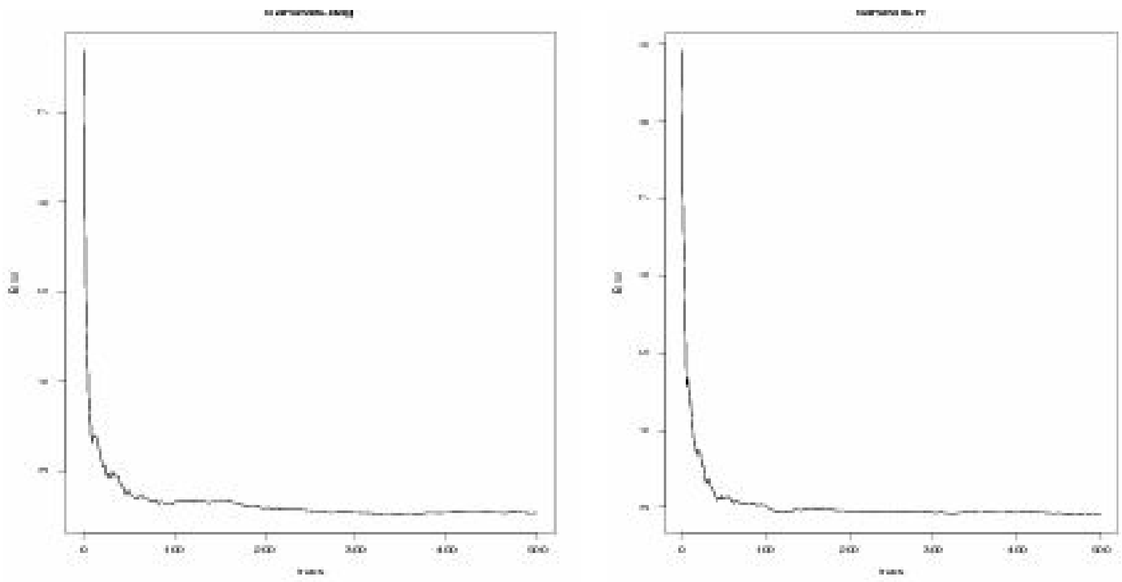

Figure 28: Left: Learning curves for bagging. **Right:** Learning curves for random forests.

Part (d): Next we use bagging of trees to predict Sales. Using the plot function in the randomForest library we get Figure 28 (left). There we see that after about 200 trees the test error stops decreasing and is mostly constant as we add more trees. The MSE on the test data for this method is 2.915448 (the smallest value we have seen thus far).

The importance function on the bagging output gives

```
              %IncMSE IncNodePurity
Population  -2.3274347      56.23009
Urban       -0.9579637       7.59014
Income       1.7408774      59.98804
US           3.6411800      10.82022
Education    3.7973472      42.11271
Advertising 13.4146209      89.73669
Age         15.5046254     112.29879
CompPrice   22.7283190     130.11907
ShelveLoc   50.4850425     351.43732
Price       57.0800460     378.61700
```

The largest values of %IncMSE are from Price and ShelveLoc indicating that they are the two most important variables affecting Sales.

Part (e): Next we use random forests to predict Sales in the Carseats dataset. Again using the plot function in the randomForest library we get Figure 28 (right). There we see that after about 100 trees the test error stops decreasing and is unchanging as we add more trees. This is effectively the same qualitative plot we got from bagging results. The MSE on the test data for this method is 3.701603 which is larger than the bagged result but smaller than the two regression tree models considered (the original and the pruned trees).

The importance function on the bagging output gives

```
              %IncMSE IncNodePurity
Urban       -1.9313442      9.993526
Population  -1.5276568     91.356145
Income       0.1595107     91.115688
Education    1.8069254     60.203614
US           2.8760212     15.125488
Advertising 11.4166337    114.286415
CompPrice   11.8018196    116.301059
Age         11.9500572    146.522301
ShelveLoc   35.0778886    254.106774
Price       36.8943346    293.183959
```

This again indicates that Price and ShelveLoc are the two most important variables affecting Sales.

Exercise 9

See the R code chap_8_prob_9.R where this problem is worked.

Part (a-b): Following the given directions in this exercise we find that the summary command on the resulting tree gives

```
Classification tree:
tree(formula = Purchase ~ ., data = OJ[train, ])
Variables actually used in tree construction:
[1] "LoyalCH"       "SalePriceMM"   "ListPriceDiff" "PctDiscMM"
Number of terminal nodes:  8
Residual mean deviance:  0.7414 = 587.2 / 792
Misclassification error rate: 0.1575 = 126 / 800
```

This output gives the requested information.

Part (c): A partial output when we view the tree in R is given by

```
> tree.OJ
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

  7) LoyalCH > 0.764572 272   92.12 CH ( 0.95956 0.04044 ) *
```

This means that the final step to get to the seventh terminal node was to require that LoyalCH be greater than 0.764572, there were 272 observations in that branch, the deviance was 92.12, the prediction was CH with probabilities of 0.95956 for CH and 0.04044 for MM.

Part (d): The tree is presented in Figure 29.

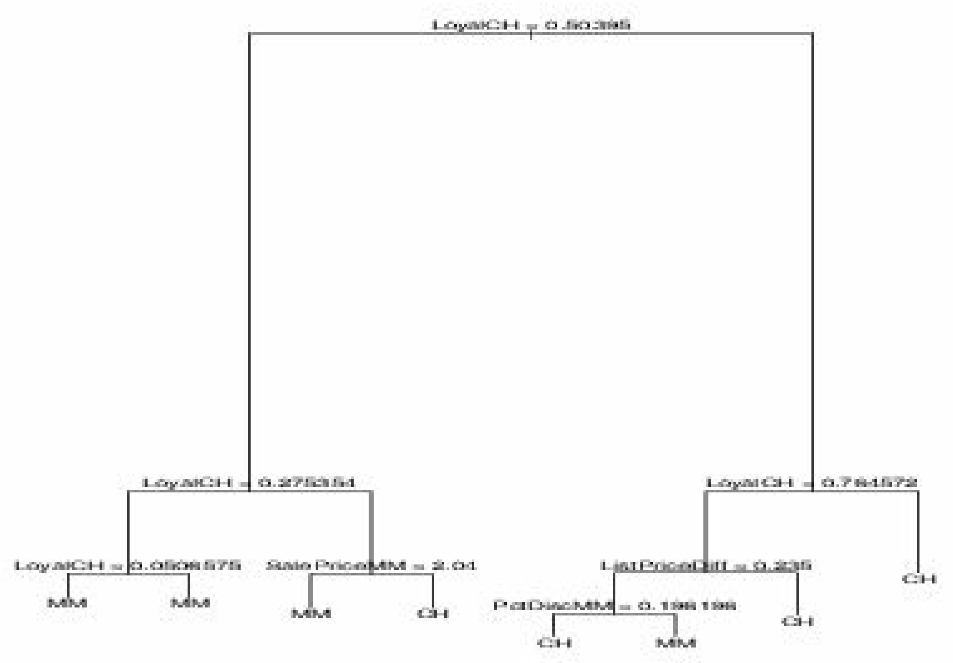

Figure 29: The initial tree built from the training data. Note that no pruning has been done on this tree.

Part (e): The confusion table for the test data is given by

```
y_hat  CH  MM
   CH 142  27
   MM  20  81
```

this gives a misclassification error rate of 0.1740741.

Part (f-i): The cross-validation results for the tree is given in Figure 30 (left). From that plot we can see that the smallest deviance value corresponds to a *k* = 5 node tree. The pruned tree for this many terminal nodes is given in Figure 30 (right).

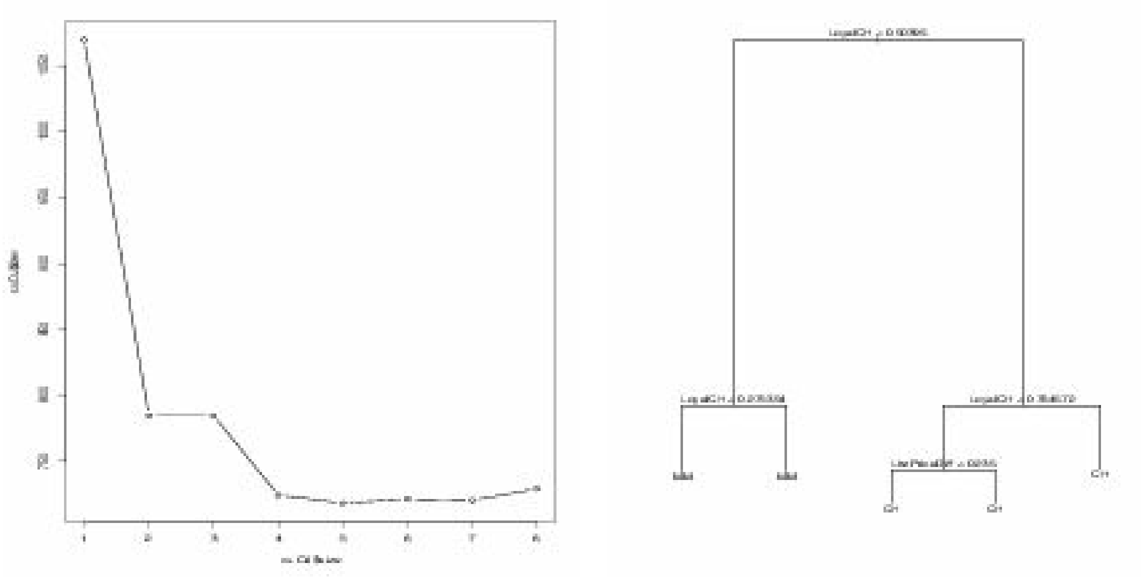

Figure 30: Left: Output from the prune.tree command. **Right:** The tree pruned to have *k* = 6 leaf nodes.

Part (j-k): On the training set the pruned tree has an error rate of 0.19875. On the test set has an error rate of 0.2037037. Both the training and testing error rates of the pruned tree is larger than in the unpruned tree. This is understandable since the cross-validated results show that for trees with $k \geq 4$ the deviance is about constant. The differences in deviances above could be within statistical noise.

Exercise 10

See the R code chap_8_prob_10.R where this problem is worked.

Part (a-d): A plot of the training MSE (in red) and the test MSE (in green) as a function of the shrinkage parameter λ are given in Figure 31.

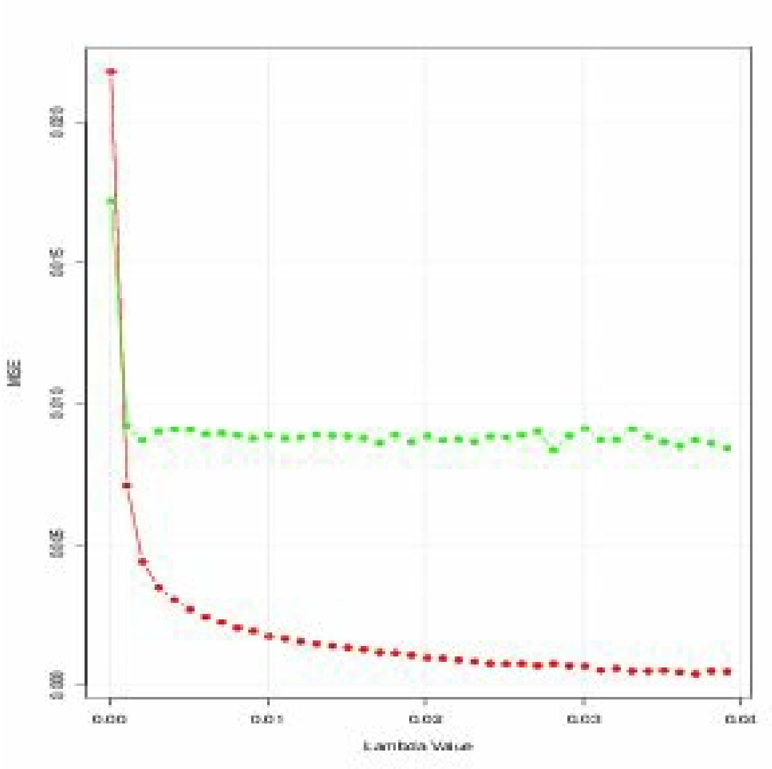

Figure 31: The MSE for the training (red) and the test (green) for predicting Salary in the Hitters dataset.

Part (e): The testing MSE error for several of the regression methods tried were

```
[1] "regression boosting test MSE:"
[1] 0.2756909
[1] "linear regression test MSE:"
[1] 0.4917959
[1] "lasso regression test MSE:"
[1] 0.4382486
[1] "ridge regression test MSE:"
[1] 0.4382486
```

Note that regression boosting is the best method tested.

Part (f): Running the summary command on the boosted output we get the following (truncated to save space)

```
> summary( boost.hitters )
             var    rel.inf
CAtBat       CAtBat 27.9843348
```

```
CHits        CHits  9.7163571
CRBI          CRBI  9.3564948
CWalks      CWalks  8.4641693
CRuns        CRuns  7.2820304
Years        Years  5.4674280
PutOuts    PutOuts  4.5604591
Walks        Walks  4.3724216
AtBat        AtBat  4.0692012
CHmRun      CHmRun  3.6759612
Assists    Assists  3.1511172
```

From this output we see that CAtBat (the number of times a player is at bat during his career) seems to be the most important variable in predicting Salary.

Part (g): Using the randomForest command on this dataset gives a test MSE of 0.2172229 which is slightly better than that from using boosting.

Exercise 11

See the R code chap_8_prob_11.R where this problem is worked.

Part (a-b): The summary command on the output of the gbm gives the following partial output

```
> summary( boost.caravan )
              var    rel.inf
PPERSAUT PPERSAUT 9.66288888
MKOOPKLA MKOOPKLA 7.91434045
MGODGE     MGODGE 5.53600913
PBRAND     PBRAND 5.14107349
MOPLHOOG MOPLHOOG 4.87826510
MBERMIDD MBERMIDD 4.49116608
MINK3045 MINK3045 3.44802622
MOSTYPE   MOSTYPE 3.17889172
MAUT2       MAUT2 3.00517585
MBERARBG MBERARBG 2.97523350
MGODPR     MGODPR 2.95870181
MSKC         MSKC 2.52642801
```

Thus we see that the two features PPERSAUT and MKOOPKLA seem to be the most informative.

Part (c): A confusion matrix for the testing data looks like

```
will_buy    0    1
```

```
    0 4356  251
    1  177   38
```

Here the two rows correspond to the predictions to not-buy (0) and to buy (1), while the two columns correspond to the true not-buy / buy labels. We see that from the 177 + 38 = 215 people we predict will make a purchase in fact only 38 do. This is the fraction (precision) 0.1767442. We find that logistic regression has a confusion matrix given by

```
will_buy    0    1
    0 4183  231
    1  350   58
```

which has a precision of 0.1421569 which is less (worse) than that obtained using gradient boosting. We note however that the summary command on the logistic regression object shows many of the variables to have no statistical significance. In fact two variables AWERKT and AZEILPL can’t have their coefficients estimated. Some form of subset selection should be performed in order to possibly have better out-of-sample results using logistic regression.

Exercise 12

As this is very much like the previous problems but only using different datasets the previous problems provide templates that one can follow to perform the analysis requested here.

Chapter 9 (Support Vector Machines)

Conceptual Exercises

Exercise 1

Each of these expressions is a line in X_1-X_2 space. Evaluating the expression for the line for a (X_1,X_2) pair that is not on the line will tell you if the half-space where the evaluation point falls is the “positive” or the “negative” half-space.

Exercise 2

The given expression (when viewed as an equality) is a circle with a center at $X_1 = -1$ and $X_2 = 2$ and a radius $\sqrt{4} = 2$. The points where this expression is positive/negative are points outside/inside the given circle. Expanding each quadratic expresses the left-hand-side as a linear expression in X_1, X_1^2, X_2, and X_2^2.

Exercise 3

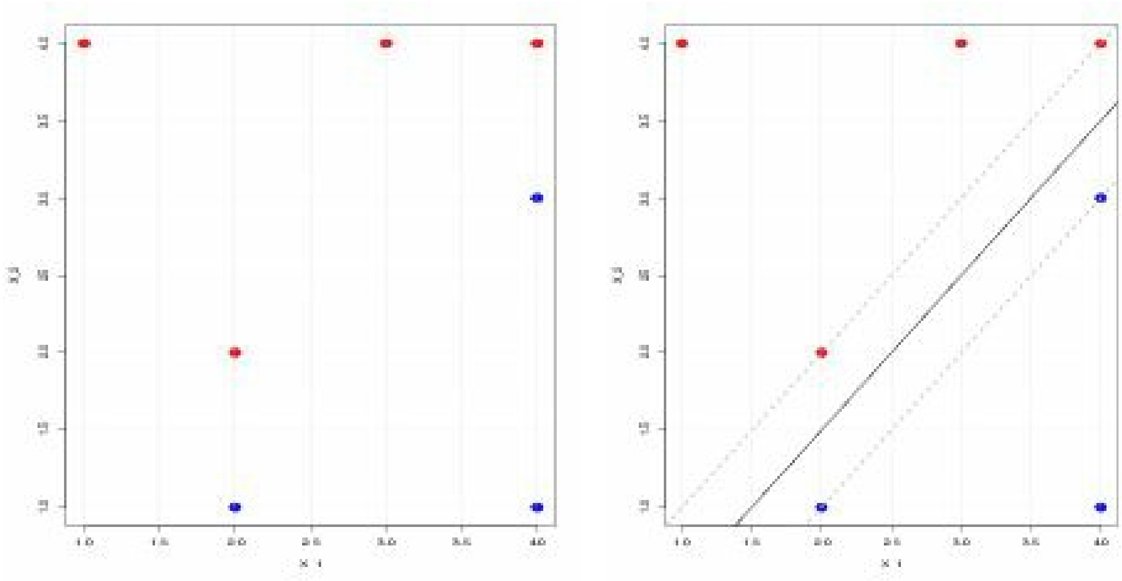

Figure 32: Left: A plot of the raw data in exercise 3. **Right:** The data in exercise 3 with the optimal separating hyperplane (in black) and two lines (in gray) that go through the support vectors on each side of the separating hyperplane.

Part (a-b): See the plot given in Figure 32 (left). From the given data we expect the optimal separating hyperplane to pass through the points (2, 1.5) and (4, 3.5). In Figure 32 (right) we draw this line which has an equation given by

$$0.5 - x_1 + x_2 = 0.$$

Part (c): The classification rule can be read from Figure 32 (right) i.e. if a point falls above the given line meaning that

$$0.5 - x_1 + x_2 > 0,$$

we classify the point as "red" while if our point is below the given line meaning that

$$0.5 - x_1 + x_2 < 0,$$

we classify the point as "blue".

Part (d): The margin is the perpendicular distance between either of the two gray lines and the black line.

Part (e): The support vectors are the four points that pass though the gray lines. These points are

$$(2, 1), (2, 2), (4, 3), (4, 4).$$

Part (f): The seventh point is located at (4, 1) which is far from the separating hyperplane and not close to any of the supporting vectors which determine the separating hyperplane. As such small movements in its location won't change the separating hyperplane.

Part (h): The two classes will no longer be linearly separable if we select a side of the separating hyperplane and place a point on that side with the a classification different from what the original separating hyperplane would predict.

Applied Exercises

Exercise 4

This is very much like Exercise 5 below.

Exercise 5

See the R code chap_9_prob_5.R where this problem is worked.

Using the suggested data set I found convergence issues in fitting the non-linear logistic regression model. Attempting to increase the number of points from 500 to 5000 did not seem to help this issue. I'll assume that we can ignore the error messages produced by glm.fit and proceed to answer the given questions using what results we do obtain. Since we know the true distribution of the classification labels we can compare the decision region produced by glm with the known truth. It appears that glm (even with the error message) is finding the correct decision boundary (see Figure 34 (right)) and thus ignoring these error messages is probably acceptable.

Part (a-b): See Figure 33 for a plot of the initial data. Notice that the points are colored according to their class label. We see the hyperbolic decision boundary that was used to produce data for the two classes.

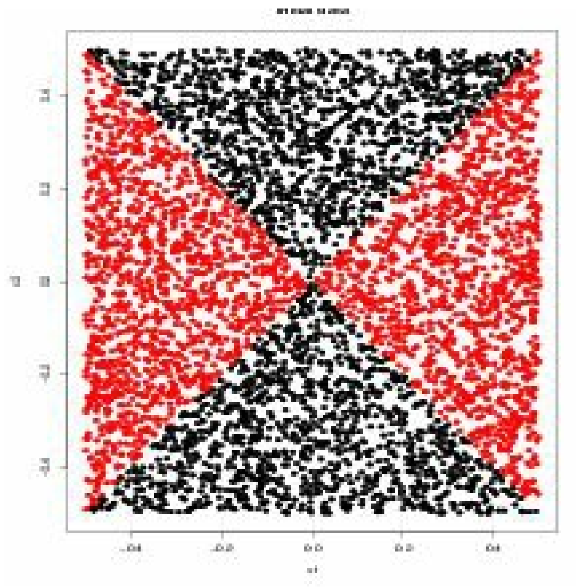

Figure 33: A scatter plot of the original data for Exercise 5. The two classes are color coded red and black.

Part (c-f): A logistic regression model with only X_1 and X_2 as features gives the training classifications shown in Figure 34 (left). Notice that in this case we obtain a linear decision boundary that matches the true decision boundary quite poorly. A logistic regression model with X_1, X_2, X_1^2, X_2^2, and X_1X_2 as features give the training classifications shown in Figure 34 (right). Notice that in this case the decision boundary looks quite close to the known true decision boundary.

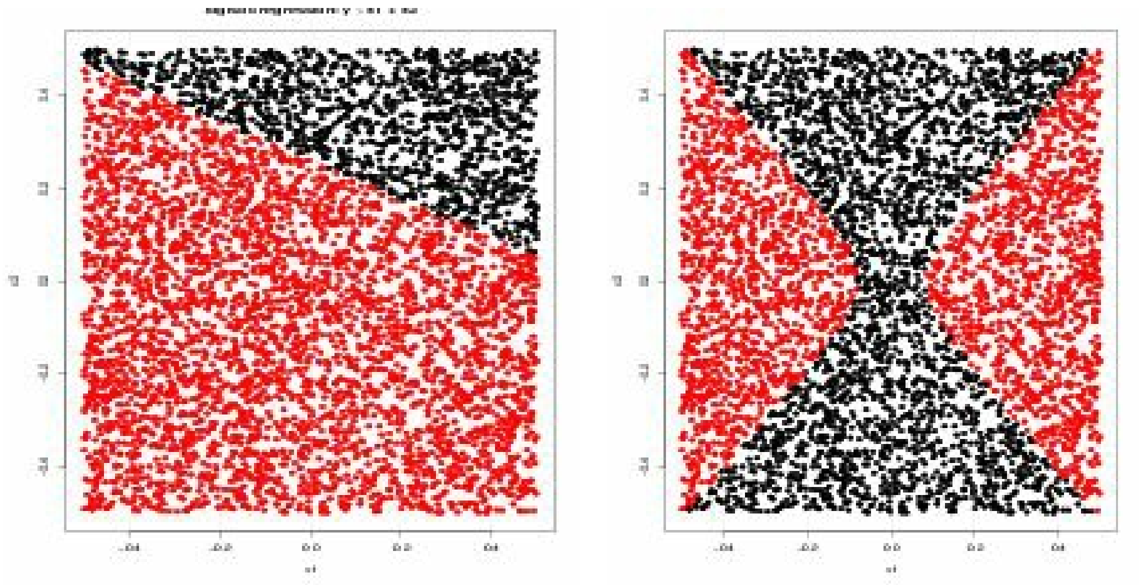

Figure 34: Left: Logistic regression using X_1 and X_2 as features. **Right:** Logistic regression using X_1, X_2, X_1^2, X_2^2, and X_1X_2 as features.

Part (g-h): In the first part of this problem we use a support vector machine with a *linear* kernel. We know that this will produce a linear classification region and will therefore not be optimal given the training data. To determine the numerical value of the cost parameter in the svm function call we use the tune function. When we do that and then use the resulting classifier to predict the training data we get the plot given in Figure 35 (left). Notice that the linear SVM is predicting all of the samples to be of the same class and thus we expect it to have an error rate equal to that of the class with the smaller a-priori probability which in this case is class 0 of 0.4904. Following this we fit a non-linear SVM using the “radial” kernel and also use the tune function to specify both the value of cost and gamma. When we do this and then use the resulting classifier to predict the training data we get the plot given in Figure 35 (right).

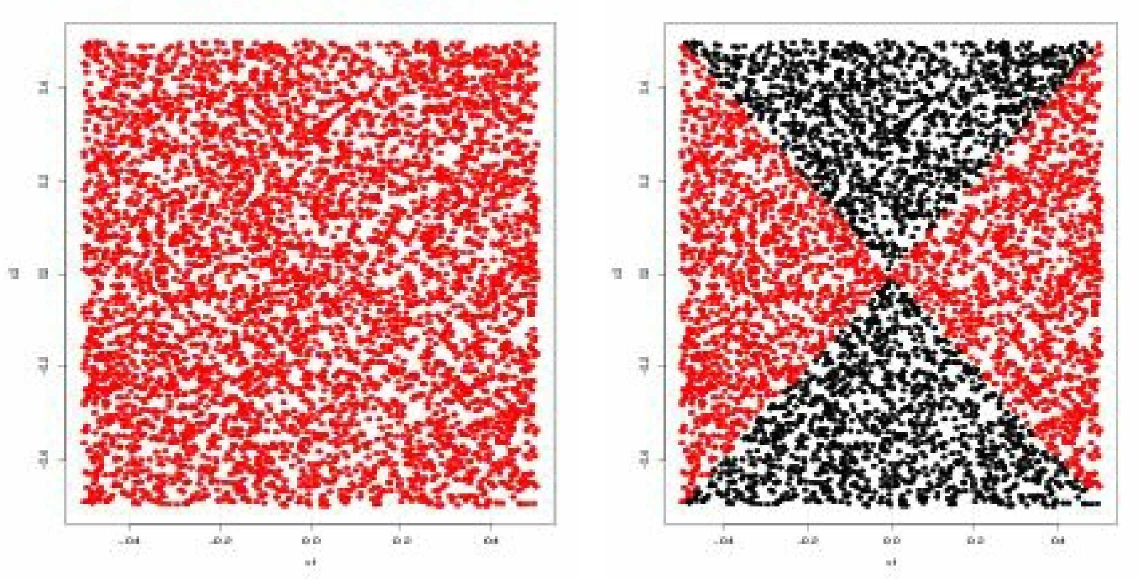

Figure 35: Left: Using a support vector machine to classify the training data with a linear kernel. **Right:** Using a support vector machine to classify the training data with a non-linear kernel.

Part (i): As a summary of the results obtained with the classifiers thus far we present the training error rate for each of the classifiers. We find

```
[1] "Linear logisitic regression training error rate=   0.395800"
[1] "Non-linear logistic regression training error rate=   0.035800"
```

```
[1] "Linear SVM training error rate=   0.490400"
[1] "Nonlinear SVM training error rate=   0.001400"
```

We see that both techniques that result in linear classification boundaries give similar training error rate. The two techniques that result in non-linear boundaries also give similar error rates. One can also see visually (using the plots above) that the two non-linear techniques are estimating the correct hyperbolic decision boundary while the linear techniques are not.

Exercise 6

See the R code chap_9_prob_6.R where this problem is worked.

Part (a): We use the code given in the R lab session in this chapter to generate data that is just barely linearly separable. A plot of this initial data (color coded by classification label) is given in Figure 36.

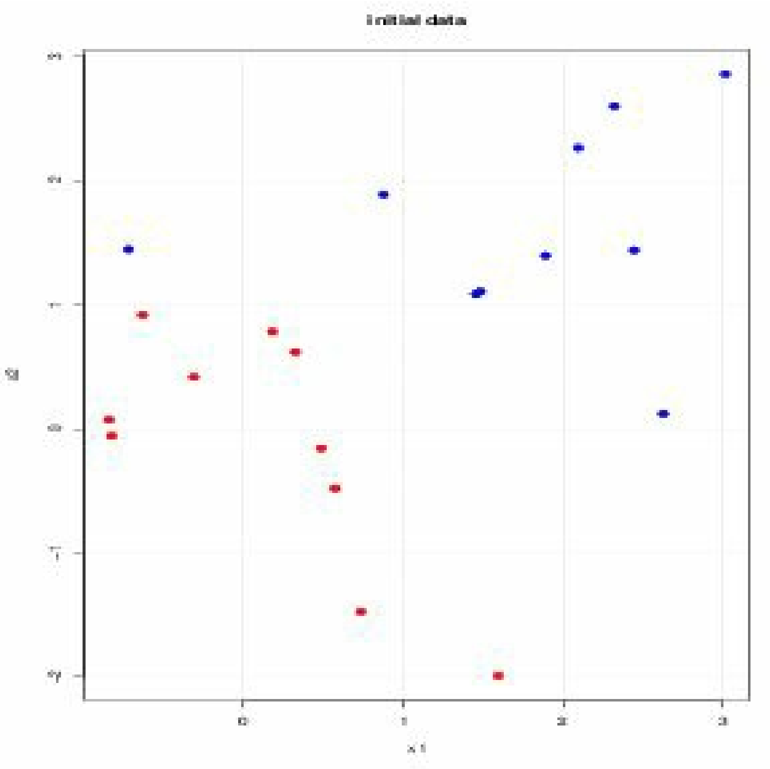

Figure 36: The data for Exercise 6.

Part (b): Using the tune function to train a SVM linear classifier we find that for the cost value given by 0.1 we get the best cross-validated error and dispersion. The summary command on the output of the tune function gives (truncated output)

```
- best performance: 0.05

- Detailed performance results:
   cost error dispersion
1 1e-03  0.50  0.4714045
2 1e-02  0.50  0.4714045
3 1e-01  0.05  0.1581139
4 1e+00  0.05  0.1581139
5 5e+00  0.10  0.2108185
6 1e+01  0.10  0.2108185
7 1e+02  0.10  0.2108185
8 1e+03  0.10  0.2108185
```

The error column in the above output is the training classification error rate. Given that we have *n* = 20 training samples we have 0.05(20) = 1 misclassified training samples. Applying the optimal value of cost found above on a test data set we get a test error rate of 10%.

Exercise 7

See the R code chap_9_prob_7.R where this problem is worked.

Part (a-b): We use the tune function to train a linear SVM classifier over various values of the cost parameter. When we do this we get the following partial output

```
- best performance: 0.09192308

- Detailed performance results:
   cost      error dispersion
1 1e-03 0.13269231 0.06908289
2 1e-02 0.09192308 0.04997918
3 1e-01 0.09711538 0.05248522
4 1e+00 0.09711538 0.06157796
5 5e+00 0.10467949 0.06447191
6 1e+01 0.10467949 0.06447191
7 1e+02 0.10211538 0.06161265
8 1e+03 0.10211538 0.06161265
```

Notice that the cross-validated error decreases initially and then increases as the value of the parameter cost increases thus giving an optimal value of 0.1. We can use the plot function to produce scatter plots of the given data and the classified training samples. For example in Figure 37 we display two plots of the SVM classifier.

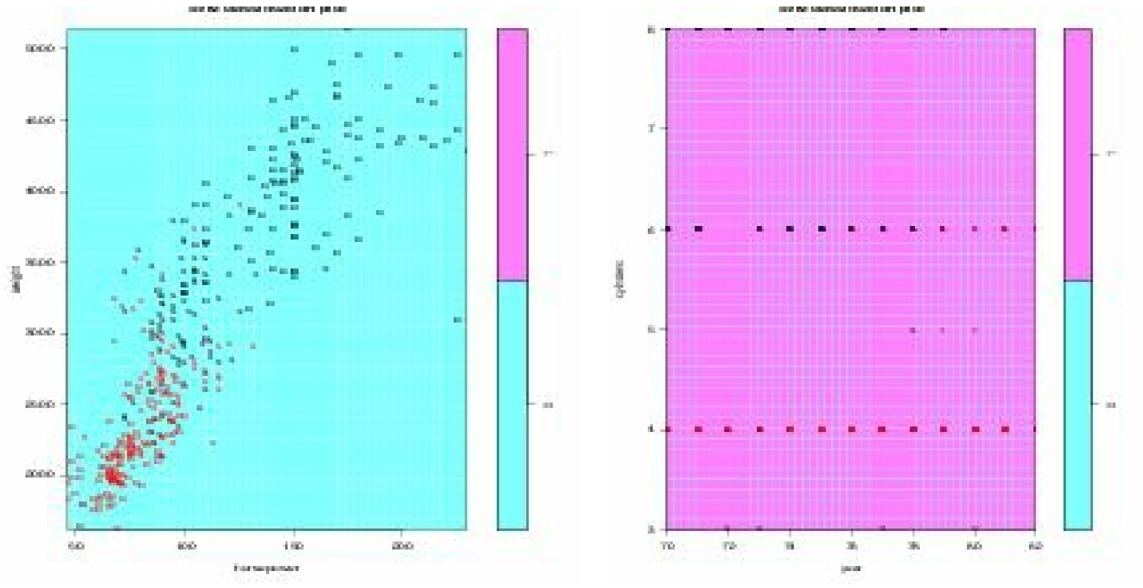

Figure 37: Using a support vector machine with a linear kernel to classify mpg from the Auto data set into “small” and “large” categories. **Left:** Plots of the SVM decision in the horsepower vs. weight space. **Right:** Plots of the SVM decision in the year vs. cylinders space. Note that support vectors are plotted with the “x” symbol while data points that are not support vectors are plotted with the “o” symbol. The black points are points where the mpg feature is greater than the median and the red points are ones where the mpg feature is less than the median.

Part (c): We next fit a non-linear SVM (using first a radial and then a polynomial kernel) on this problem to see if the predicted in-sample performance can be improved. Again we use the tune function to specify values for either gamma (when our kernel is radial) and degree (when our kernel is polynomial). Using the plot command on the optimal tune results in each case gives similar looking figures to the ones presented above. If we compare the classification performance on the training data using the three methods we get the following

```
kernel=linear     - best performance: 0.08173077
kernel=radial     - best performance: 0.07128205
kernel=polynomial - best performance: 0.07403846
```

From which we see that all three methods give comparable performance on the training data. One would need to split the data into a validation set or use cross-validation to estimate which method would be better on testing data.

Exercise 8

See the R code chap_9_prob_8.R where this problem is worked.

Part (a-b): The summary command on the suggested fit gives

```
Call: svm(formula = Purchase ~ ., data = OJ, kernel = "linear", cost = 0.01)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  560

 ( 279 281 )

Number of Classes:  2

Levels:
 CH MM
```

Part (c): We can produce a table of the predictions on the training data set

```
         truth
predicted  CH  MM
       CH 430  75
       MM  61 234
```

which has an error rate of 0.170000. A table of predictions on the testing data set gives

```
         truth
predicted  CH  MM
       CH 146  25
       MM  16  83
```

which has an error rate of 0.151852.

Part (d): The (truncated) output from the tune function gives

```
- best performance: 0.1663551

- Detailed performance results:
   cost     error dispersion
1 1e-03 0.2355140 0.04174920
2 1e-02 0.1728972 0.03531397
```

```
3 1e-01 0.1710280 0.03116822
4 1e+00 0.1663551 0.02636037
5 5e+00 0.1672897 0.02836431
6 1e+01 0.1700935 0.02405017
7 1e+02 0.1700935 0.02445036
8 1e+03 0.1747664 0.02415084
```

We see an initial decrease in the error metric followed by an increase. The value of cost that gives the minimal value of error is the value of 1.

Part (e): The training and testing error rates using the optimal value of cost and a linear kernel are given by

```
[1] "Linear SVM training error rate (optimal cost=1)=   0.165000"
[1] "Linear SVM testing error rate (optimal cost=1)=   0.148148"
```

Part (f): The training and testing error rates using an optimal value of cost (and gamma) under the radial kernel are given by

```
[1] "Radial SVM training error rate (optimal)=   0.133750"
[1] "Radial SVM testing error rate (optimal)=   0.144444"
```

Part (g): The training and testing error rates using the optimal value of cost (and degree) under the polynomial kernel are given by

```
[1] "Polynomial SVM training error rate (optimal)=   0.165000"
[1] "Polynomial SVM testing error rate (optimal)=   0.155556"
```

Note the book did not request us to optimize our prediction accuracy over the parameters gamma (when using a radial kernel) or degree (when using a polynomial kernel) but since it is easy to use tune to do this I did.

From the above test error rates it looks like the radial kernel gives the smallest testing error rate.

Chapter 10 (Unsupervised Learning)

Conceptual Exercises

Exercise 1

Part (a): We begin by expanding the quadratic on the right-hand-side of the given expression as

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}^2 - 2x_{ij}x_{i'j} + x_{i'j}^2).$$

Next we reorder the sums and perform some simplifications as follows

$$\begin{aligned}
&= \frac{1}{|C_k|}\sum_{j=1}^{p}\left(\sum_{i'\in C_k}\sum_{i\in C_k} x_{ij}^2 - 2\sum_{i,i'\in C_k} x_{ij}x_{i'j} + \sum_{i\in C_k}\sum_{i'\in C_k} x_{i'j}^2\right)\\
&= \frac{1}{|C_k|}\sum_{j=1}^{p}\left(|C_k|\sum_{i\in C_k} x_{ij}^2 - 2\sum_{i,i'\in C_k} x_{ij}x_{i'j} + |C_k|\sum_{i'\in C_k} x_{i'j}^2\right)\\
&= \frac{1}{|C_k|}\sum_{j=1}^{p}\left(2|C_k|\sum_{i\in C_k} x_{ij}^2 - 2\sum_{i,i'\in C_k} x_{ij}x_{i'j}\right)\\
&= 2\sum_{j=1}^{p}\left[\sum_{i\in C_k} x_{ij}^2 - \sum_{i\in C_k}\left(\sum_{i'\in C_k}\frac{1}{|C_k|}x_{i'j}\right)x_{ij}\right] = 2\sum_{j=1}^{p}\left[\sum_{i\in C_k} x_{ij}^2 - \sum_{i\in C_k}\bar{x}_{kj}x_{ij}\right]\\
&= 2\sum_{j=1}^{p}\left[\sum_{i\in C_k} x_{ij}(x_{ij}-\bar{x}_{kj})\right] = 2\sum_{j=1}^{p}\left[\sum_{i\in C_k}(x_{ij}-\bar{x}_{kj}+\bar{x}_{kj})(x_{ij}-\bar{x}_{kj})\right]\\
&= 2\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2 + 2\sum_{j=1}^{p}\sum_{i\in C_k}\bar{x}_{kj}(x_{ij}-\bar{x}_{kj}).
\end{aligned}$$

Notice that we can write this second term in the above as

$$2\sum_{j=1}^{p}\bar{x}_{kj}\left(\sum_{i\in C_k}(x_{ij}-\bar{x}_{kj})\right) = 2\sum_{j=1}^{p}\bar{x}_{kj}\left(\sum_{i\in C_k}x_{ij} - |C_k|\bar{x}_{kj}\right) = 2\sum_{j=1}^{p}\bar{x}_{kj}\left(\sum_{i\in C_k}x_{ij} - \sum_{i\in C_k}x_{ij}\right) = 0.$$

Thus we have shown that

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2. \qquad (16)$$

Part (b): The K-means clustering algorithm seeks to minimize summing the left-hand-side of Equation 16 for $k = 1, 2, \ldots K-1, K$. It does this by alternately selecting the cluster members (i.e. the sets C_k) and then the cluster centroids $\bar{x}_k$. One can think of this as iteratively minimizing first the $\sum_{i\in C_k}$ part of the above expression followed by the $\sum_{j=1}^{p}(x_{ij} - \bar{x}_{kj})^2$ part.

Exercise 2

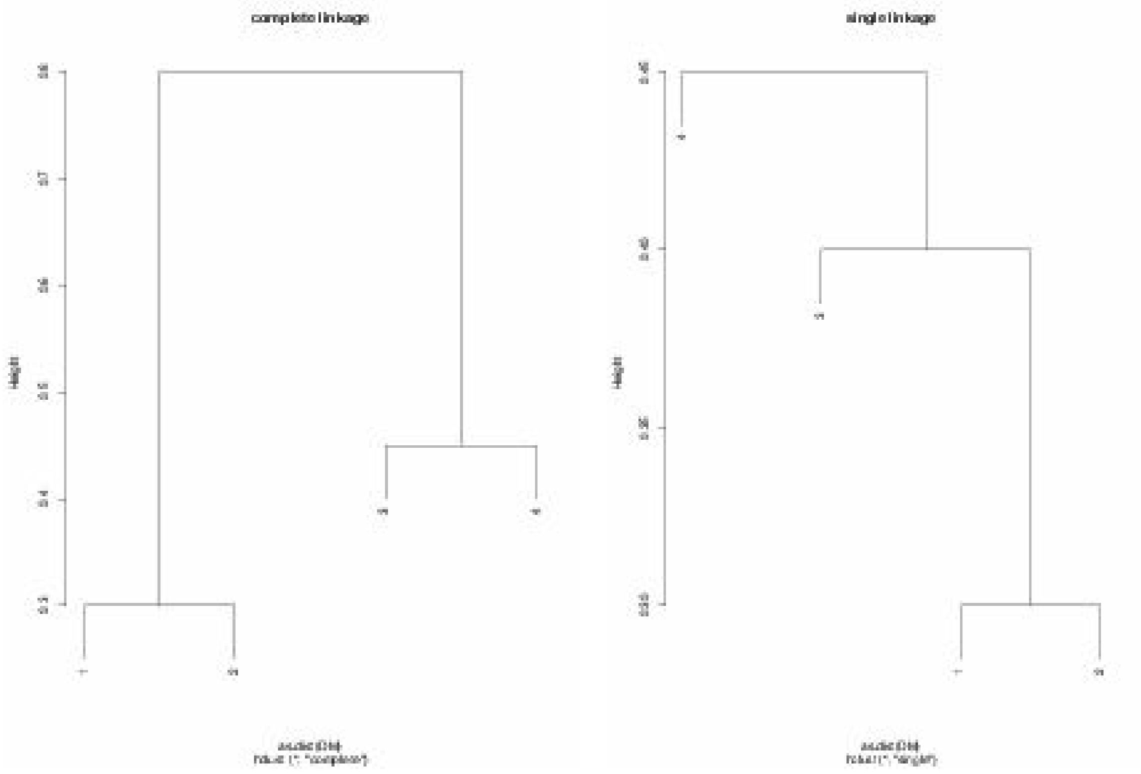

Figure 38: Left: The data from Exercise 2 clustered with complete linkage. **Right:** The data from Exercise 2 clustered with single linkage.

Note we could do these plots and calculations "by hand" but it is easier to use the hclust function on the given dissimilarity matrix. To do this we need to convert the dissimilarity matrix into an argument that hclust can understand. We can do this with the as.dist command. Calling plot on the output then gives a nice dendrogram. When we do this we get the two plots shown in Figure 38.

Part (a-b): See Figure 38 (left) and (right) respectively.

Part (c-d): Under complete linkage cutting the dendrogram to produce two clusters would place elements 1 and 2 in one cluster and elements 3 and 4 in the other cluster. Under single linkage cutting the tree to get two clusters would place elements 1, 2, and 3 in one cluster and element 4 in its own cluster. One can verify these results using the cutree command on the output of hclust.

See the R code chap_10_prob_2.R where this problem is worked.

Exercise 3

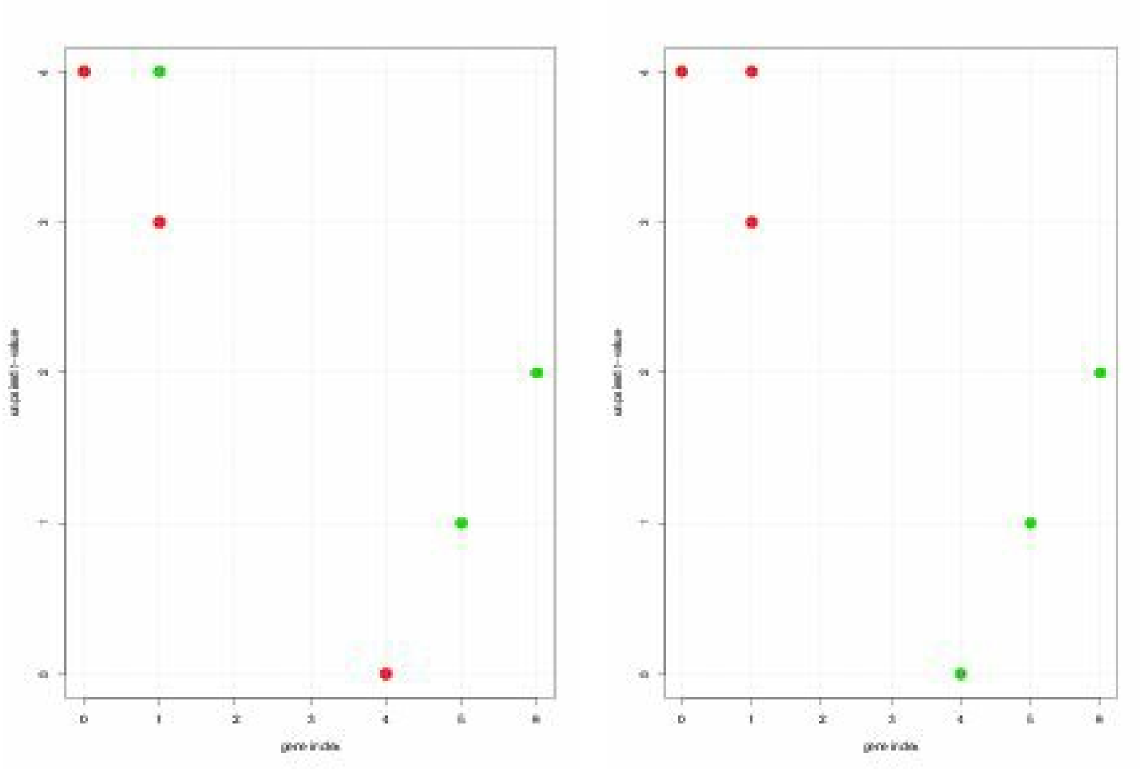

Figure 39: Left: The original cluster assignment. **Right:** The final cluster assignment.

Part (a-f): See the R code chap_10_prob_3.R where this problem is worked. When we run that program we get the two plots given in Figure 39. On this problem the *K*-means algorithm converged in very few iterations.

Exercise 4

Part (a): When performing complete linkage hierarchical clustering we compute all pairwise dissimilarities between the points in each cluster and then assign the dissimilarity between the two clusters to be the *largest* pairwise dissimilarity. In single link clustering the dissimilarity between two clusters is the *smallest* of all the pairwise dissimilarities. Thus we expect the two clusters {1, 2, 3} and {4, 5} to join later in the dendrogram than they would under single linkage.

Part (b): In this case as the two clusters fused only consist of single points they should fuse at the same level in both dendrogram trees.

Applied Exercises

Exercise 7

See the R code chap_10_prob_7.R where this problem is worked.

Note that for this problem we have to scale each *observation* to have mean zero and variance one. This is different from what we might otherwise do in machine learning where often we scale each *feature* to have mean zero and variance one. This means that we have to take the transpose of the USArrests dataframe at several places to be operating on the correct column or row depending. In the above R code we extract the lower triangular elements of $1 - R_{ij}$ (the matrix is symmetric) in the variable X and the lower triangular elements of the “distance” squared matrix produced from dist (again the matrix is symmetric) in the variable Y and then look at the ratio of these two. If X and Y are related by a multiplicative constant then this ratio would equal this constant. A plot of X vs. Y shows a very streight line with no noise. The summary command also verifies this

```
> summary( X/Y )
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1667  0.1667  0.1667  0.1667  0.1667  0.1667
```

Notice the value at every percentile point is the same i.e. the distribution is constant.

Exercise 8

See the R code chap_10_prob_8.R where this problem is worked.

Performing the two suggested calculations we get

```
> print(pve_1)
   0.62006039 0.24744129 0.08914080 0.04335752
> print(pve_2)
   0.62006039 0.24744129 0.08914080 0.04335752
```

Note that these two expressions are exactly the same as they should be.

Exercise 9

See the R code chap_10_prob_9.R where this problem is worked.

Part (a-b): The requested dendrogram is given in Figure 40 (left).

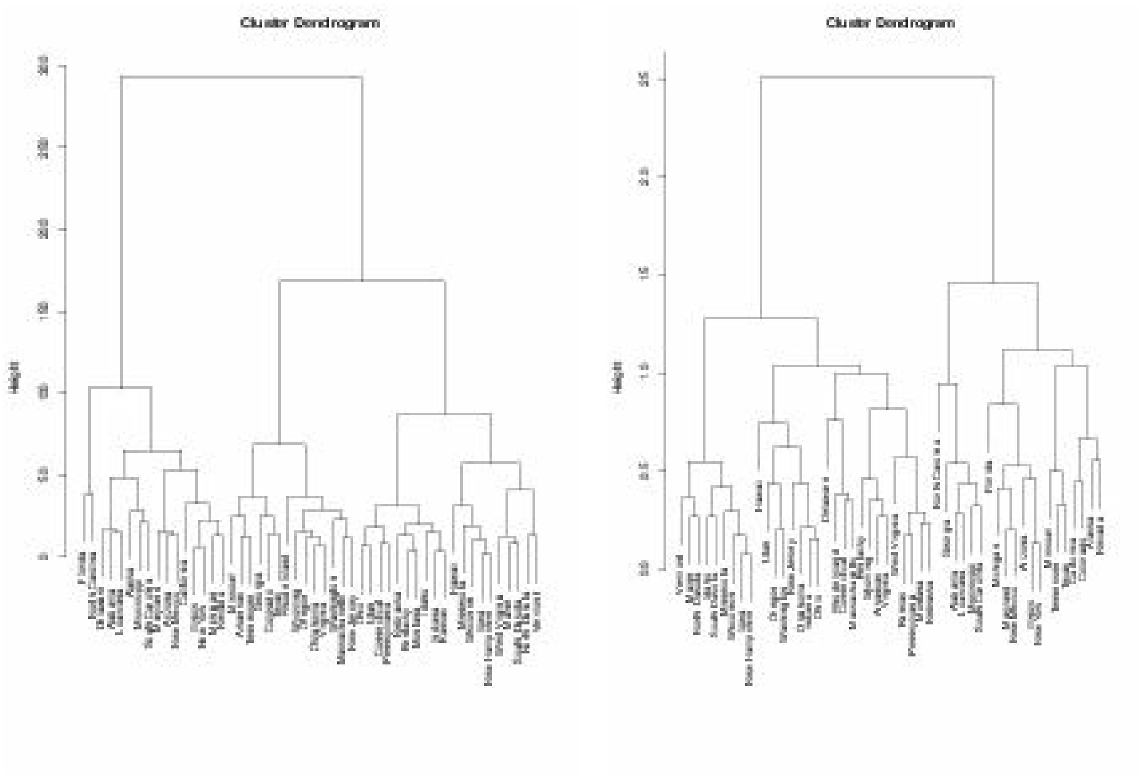

Figure 40: Left: The dendrogram produced from the USArrests data set without any transformation of the features. **Right:** The dendrogram produced from the USArrests data set where we have scaled the features to have mean zero and standard deviation one.

The states that are in the first three clusters are given by

```
[1] 1
 [1] "Alabama"       "Alaska"        "Arizona"       "California"
 [5] "Delaware"      "Florida"       "Illinois"      "Louisiana"
 [9] "Maryland"      "Michigan"      "Mississippi"   "Nevada"
[13] "New Mexico"    "New York"      "North Carolina" "South Carolina"
[1] 2
 [1] "Arkansas"      "Colorado"      "Georgia"       "Massachusetts"
 [5] "Missouri"      "New Jersey"    "Oklahoma"      "Oregon"
 [9] "Rhode Island"  "Tennessee"     "Texas"         "Virginia"
[13] "Washington"    "Wyoming"
[1] 3
 [1] "Connecticut"   "Hawaii"        "Idaho"         "Indiana"
 [5] "Iowa"          "Kansas"        "Kentucky"      "Maine"
 [9] "Minnesota"     "Montana"       "Nebraska"      "New Hampshire"
```

```
[13] "North Dakota"  "Ohio"         "Pennsylvania"  "South Dakota"
[17] "Utah"          "Vermont"      "West Virginia" "Wisconsin"
```

Part (c-d): The requested dendrogram after scaling (no translation) is given in Figure 40 (right).

Exercise 10

See the R code chap_10_prob_10.R where this problem is worked.

Part (a-b): See the plot in Figure 41 where we project the generated data into the space spanned by its first two principle components. Notice how three clusters emerge.

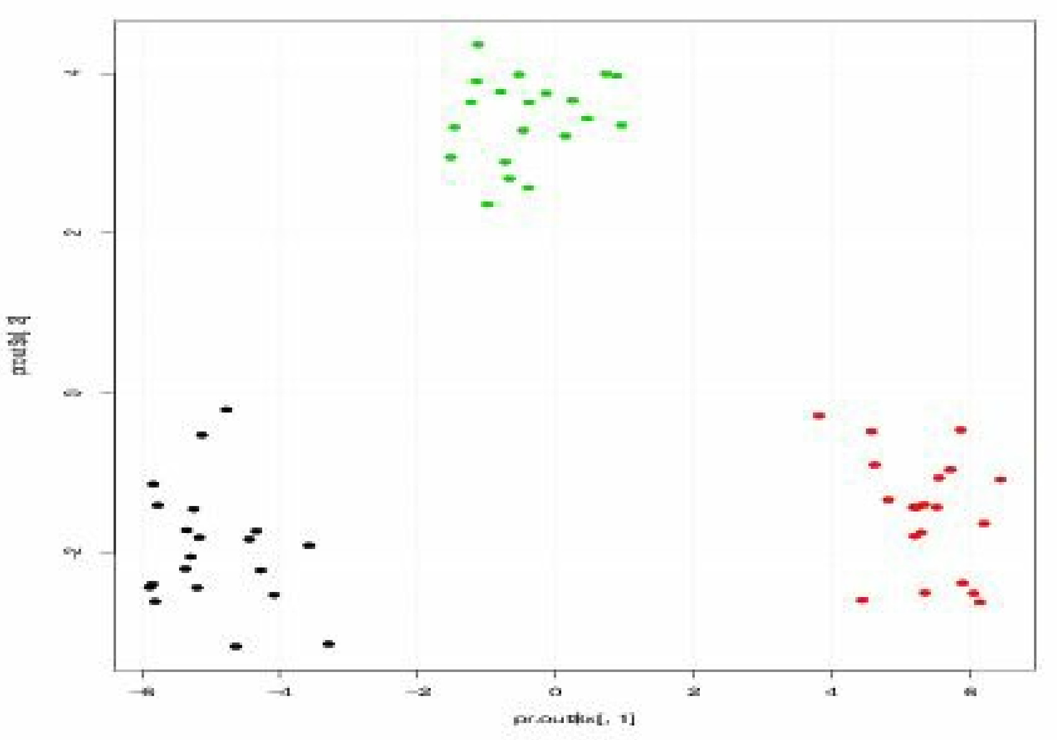

Figure 41: The 60 points projected into the space spanned by their first two principle components.

Part (c): When we run *K*-means on the original data we get back perfect agreement with the three input classes as can be seen from the following table output

```
> table( kmean.out$cluster, labels )
  labels
    1  2  3
```

```
 1 20  0  0
 2  0  0 20
 3  0 20  0
```

Note that *K*-means clustering has switched the labels of the second and third labels from that of the ”truth”.

Part (d): Clustering with *K* = 2 lumps two of the individual clusters (from the three original clusters we started with) into a single cluster and leaves one cluster untouched. One can see this with another table command

```
> table( kmean.out$cluster, labels )
   labels
     1  2  3
 1 20  0 20
 2  0 20  0
```

Notice that the items the *K*-means procedure labels as “cluster one” correspond to samples from the original data set corresponding to clusters one and three.

Part (e): Clustering with *K* = 4 splits one of the original clusters into two parts (so that we will end up with four clusters). The table command gives

```
> table( kmean.out$cluster, labels )
   labels
     1  2  3
 1  0  5  0
 2  0 15  0
 3 20  0  0
 4  0  0 20
```

Part (f): Using the first two principle components with *K* = 3 gives three clusters that correctly match the original cluster labellings. The fact that the data when plotted in terms of their principle components shows clustering (again see Figure 41) helps to understand this fact. The table command on the *K*-means output gives

```
> table( kmean.out$cluster, labels )
   labels
     1  2  3
 1  0  0 20
 2  0 20  0
 3 20  0  0
```

Part (g): Scaling the data has no effect on the clustering since the data will still have smaller variability within the three clusters than between the clusters. This will motivate the K-means algorithm to find these clusters.

Exercise 11

See the R code chap_10_prob_11.R where this problem is worked.

Part (a-b): Following exercise 7 we will take as our dissimilarity metric between the ith and jth samples to be $1 - r_{ij}$, where r_{ij} is the correlation between the two samples. Notice that this function will have its smallest value (of zero) if $r_{ij} = 1$ i.e. the two samples are perfectly correlated. This function will have its largest value (of two) if $r_{ij} = -1$ i.e. the two samples are perfectly anti-correlated. We expect that the gene expression levels of two patients that have the same type of cancer will have very correlated levels. Using this metric as our dissimilarity measure with the hclust function (and complete linkage) we get the dendrogram given in Figure 42 (left).

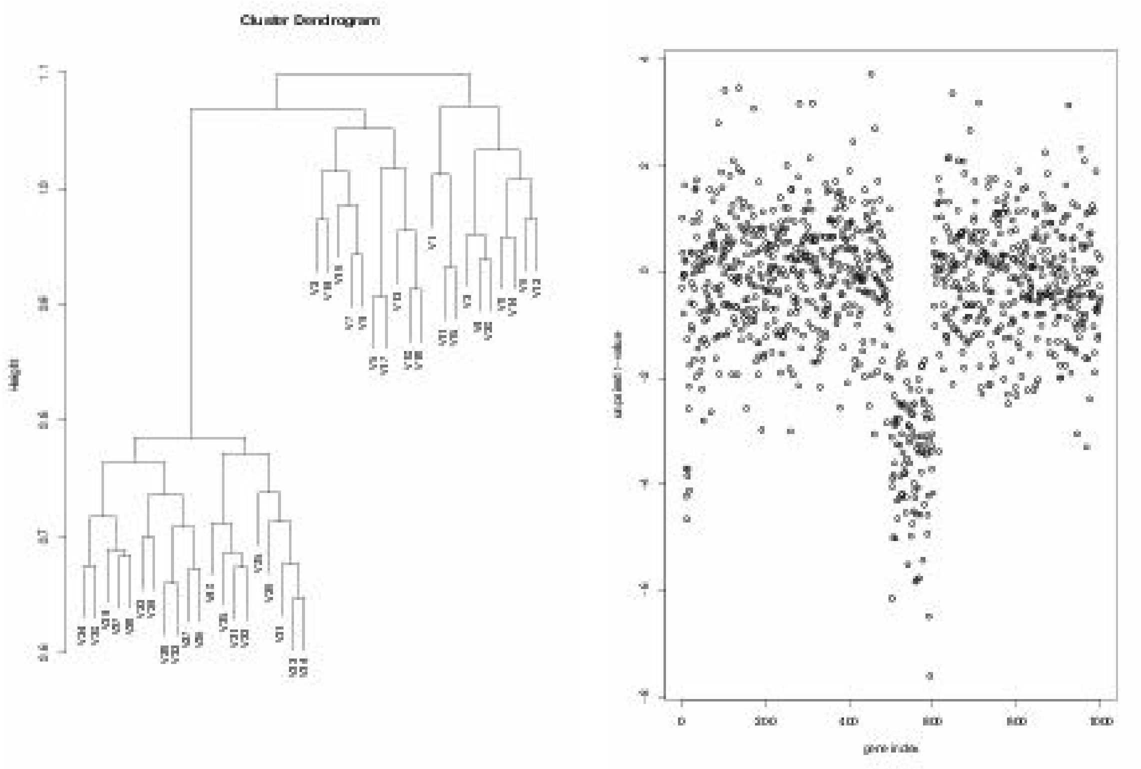

Figure 42: Left: Clustering genes for Exercise 11. **Right:** The unpaired t-statistic for each gene.

If we compare the predicted cluster label to the known truth label where we take 0 to be a healthy patient and 1 to be a diseased patient we get the following

```
         truth
predicted  0  1
        1 10  0
        2 10 20
```

It looks like our predicted class label of “1” corresponds to 10 healthy patients, while the predicted class of “2” corresponds to 10 healthy patients and 10 diseased patients. This might indicate that we want to look for three clusters where this extra cluster might split these 10 patients from the 20 diseased ones. These results do depend on the linkage used as is typically the case.

Part (c): We could apply clustering of the patients as done above and then look at the mean value of each gene in each cluster. For each gene we would have two mean values corresponding to the means in each cluster. The genes with the largest difference in their two means could be reported. A more precise measure would be to compute an unpaired *t*-statistics between the means of each gene in the two clusters. When we do this and then plot this *t*-value as a function of gene index we get the plot given in Figure 42 (right). There we see that there are a large number of genes with indices between 500 and 600 that seem to be most different between the two clusters. While 100 genes might be a large number we could sort these further to give a handful of genes that seem to have the most statistically significant difference in their mean values between the clustered groups.

References

[1] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Pearson Education, 6 edition, 2003.

[1] sometimes abbreviated as ISL.

[2] sometimes abbreviated as ESL.