Hints for Homework 2
Course : Stat-210

1. Heteroscedasticity means that the variances are different, as opposed to the homoscedastic case. This problem gives one way to handle heteroscedastic case. Here we are assuming that all the error variances are known multiples of some unknown $\sigma^2$. This scenario arises where we know some assignable cause behind the heteroscedasticity, eg, some observations are taken by novice experimenters some by experts. In this case we can assign $w_i = 2$, say, for the novices, while $w_i = 1$ for the experts. Obviously, there is subjectivity in choosing the weights, $w_i$'s.

Another method to handle heteroscedasticity is by variance stabilizing transformations, which we shall discuss in class.

2. Leverage points constitute a trecherous kind of outliers. They do not conform to the general pattern, but still attract the linear regression line so much towards themselves that the “good” points seem to be bad fits!

Try dragging the mouse in a circle around the the data cloud, and see where the regression line changes the direction of motion. Relate that to the data cloud. I do not expect any unique answer to this question, but shall look for one important point.

3. This is a standard diagnostic method in multiple linear regression. The S-plus function lm() will help you here. Please read the online help for useful info. Here is a basic overview. Suppose you want to regress $y$’s on $x_i$’s using the model

$$y_i = \beta_0 + \sum \beta_i x_i + e.$$  

Store the data for the $y$ in a vector, y, say, and store the the $x_i$’s in vectors x1, x5, say. Then to do the regression you have to use the command

```R
> reg <- lm(y ~ 1 + x1 + x2 + x3 + x4 + x5)
```

This means “regress the vector $y$ as a linear combination of 6 vectors: the vector of all ones and the $x_i$ vectors.” The result is stored in the variable reg. You can look up the various parts of reg as follows.

```R
> coef(reg)
```

will list the estimated coefficients. However, the command

```R
> summary(reg)
```

will produce a more comprehensive and self-explanatory output. To do linear regression using only a subset of the data you can use a command like

```R
> reg <- lm(y ~ 1 + x1 + ... + x5, subset = -5)
```

This would drop the 5th case (ie, $(x_5, y_5)$) from consideration. Another S-plus function that comes handy is diag(). It has a two-fold purpose. The command

```R
> a <- diag(A)
```

will store the diagonal entries of A in a vector a. If, however, the input to the function is a vector, like this

```R
> A <- diag(a)
```
then $A$ will store a diagonal matrix with entries of $a$ as its diagonal elements.

4. This is a simpler version of what we have already done in class.

5. Simple matrix algebra.

6. This exercise requires you to make quite a few plots. Rather than make one plot per page you might like to make 4 per page. To do so use the S-plus command

```r
> par(mfrow=c(2,2))
```

before invoking any plotting command. The `lm()` function should help you here too. To do multiple regression of $y$ on $x_1, x_2,$ and $x_3,$ you have to use the command

```r
> reg <- lm(y~x1+x2+x3)
```

7. This is an example of polynomial regression. One often starts with a theoretical model that requires fitting a high degree polynomial, and then tests if the higher order terms are really significant or not. Finally, one chooses the least degree polynomial that fits the data well. Though there is only one predictor yet it is a multiple linear regression problem. And, yes, it is a linear regression problem, even though a cubic polynomial is not linear. It is linear regression because the model is linear in the parameters.

8. Here is a simple function in S-plus that will give you an idea about how to write the ridge function. This sample function computes the soln of the eqn

$$(A + I)x = \lambda b$$

for a given matrix $A$, vector $b$, and constant $\lambda$, which has default value 1.

```r
myFunc <- function(A,b,lambda=1)
{
  nRow <- dim(A)[1]
  nCol <- dim(A)[2]
  if(nRow != nCol) {stop("A is not square!")}
  if(nRow != length(b)) stop("Dimensions do not match!")
  diag(A) <- diag(A) + 1 #adding 1 to each diagonal element
  #which is the same as adding I to A
  solve(A,lambda*b)
}
```

Note that there is no `return` statement in the function. By default an S-plus function returns the value of the last command in the function. You could also store the result in some vector $x$, and then use the command `return(x)`.

Also, the following commands will be useful: `%%` for matrix multiplication, and `t(A)` for the transpose of a matrix $A$. 