Multiple Linear Regression

Basic Concepts

Multiple linear regression is the extension of simple linear regression to the case of two or more independent variables.

In simple linear regression, we had the basic equation:

$$Y = \alpha + \beta \times X + \text{ “error”}$$

whereas now this is extended to:

$$Y = \alpha + \beta_1 \times X_1 + \beta_2 \times X_2 + \beta_3 \times X_3 + \ldots + \beta_p \times X_p + \text{ “error”}$$

As was also the case in simple linear regression, we can alternatively express the model as:

$$E(Y) = \alpha + \beta_1 \times X_1 + \beta_2 \times X_2 + \beta_3 \times X_3 + \ldots + \beta_p \times X_p$$

The “times” symbol is often omitted, giving the notation

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_p X_p + \text{ “error”}$$

The outcome or dependent variable is again denoted by Y, while $X_1, X_2, \ldots, X_p$ represents the set of independent or predictor variables. The parameter α represents the intercept of the linear regression equation (intercept = the value of Y when **all** independent variables X are set equal to zero), and $\beta_1, \beta_2, \ldots \beta_p$ are the regression coefficients for each independent variable, $X_1, X_2, \ldots, X_p$, respectively. Note that p represents the number of independent variables included in the model.

To interpret the β coefficients, we note that if X changes by one unit (for example, changes from $X = 0$ to $X = 1$, or from $X = 2.1$ to $X = 3.1$, and so on), then the mean of Y, $E(Y)$, changes by β, all else remaining equal.

As was the case for simple linear regression, in “simple” multiple regression, we need to make several assumptions, including:

- The “errors” (again known as “residuals”) are independent $N(0, \sigma^2)$

- σ^2 is constant throughout the range
- The relationship between *each* X_i and Y is a straight line.

Unlike linear regression restricted to one variable, in multiple linear regression we are not restricted to linear relationships, because we can, for example, add polynomial terms. We will soon reanalyze our fluoride DMF teeth example adding in a polynomial term to improve our previous linear model.

Brief Outline: Estimation of Regression Coefficients and the Residual Variance

Let's write down the likelihood function for multiple linear regression and see how the coefficients are estimated:

Recall once again that the likelihood function of a model gives the probability that the data would be observed, given the unknown parameters, α, β_i, $i = 1, 2, \dots, p$, and σ^2, in this case, where p is the number of independent variables. The form of the multiple linear relationship, together with the normality assumption for the errors, implies the following, for each data point included in the data set, $i = 1, 2 \dots, n$:

$$Y_i \sim N(\alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots \beta_p X_{ip}, \sigma^2)$$

Recall also the form of the normal distribution:

$$f(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2\right)$$

Thus, the likelihood function contribution from a single data point $(X_{1i}, X_{2i}, \dots, X_{ip}, Y_i)$ pair is:

$$like(Y_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{Y_i - (\alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots \beta_p X_{ip})}{\sigma}\right)^2\right)$$

Since each data point is assumed independent of all others, and since independent probabilities multiply (basic probability rule), we have the following likelihood for our model:

$$like(Y) = \prod_{i=1}^{n} \left\{ \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \left(\frac{Y_i - (\alpha + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots \beta_p X_{ip})}{\sigma} \right)^2 \right) \right\}$$

As is often the case, frequentist inference proceeds by maximizing this likelihood function to find estimates of α, the p β_j's and σ^2. We will omit the details, but the summary of the steps involved is identical to that from simple linear regression:

1. Take the logarithm of the likelihood function. Recall that the logarithm of a function will have the same maxima as the function itself. Because of the exponential term, the log is easier to maximize.
2. To find the values of α, β_j, $j = 1, 2, \ldots, p$ and σ^2 that maximize the logarithm of the likelihood, take the (partial) derivatives with respect to α, the β_j's and σ^2, and set these equations equal to zero. Solving this system of equations ($p+2$ equations in $p+2$ unknowns) gives the maximum likelihood estimators for our unknown parameters.

If the above steps are followed, we can find the the maximum likelihood estimates, but, unlike the case for a single variable, the formulae are extremely tedious to solve by hand, and not that intuitive to look at, and so are omitted here.

[Note to those who know some linear algebra: A compact way to express the estimated beta coefficients is: $\hat{\beta} = (X'X)^{-1}Y$, where X is a "design matrix" comprised of the individual data points X_{ij} arranged into a matrix, preceded by a column of ones representing the "multipliers" of the α intercepts.]

Although we do not present the estimation formulae in complete detail (computer packages such as R will do the calculations for us), it is still useful to note that, as was the case with simple linear regression, all maximum likelihood estimators are asymptotically (as sample size n goes to infinity) unbiased and minimum variance. For finite sample sizes, $\hat{\alpha}$ and $\hat{\beta}$ are unbiased but $\hat{\sigma}^2$ is not unbiased. Therefore, it is more common to use an unbiased (but not maximum likelihood) estimator for σ^2:

$$\hat{\sigma}^2 = \sum_{i=1}^{n} \frac{(Y_i - \hat{Y})^2}{n - (p+1)}$$

Note that this is just the sum of squared distances from the data points to the predicted line, divided essentially by the number of data points (adjusted for number of degrees of freedom "lost" in estimating the parameters).

Inference for Regression Parameters

Inferences when there are more than one X variable are very similar to when there is just a single independent variable. First, when making predictions for future observations, we use:

$$\hat{y}_i = \hat{\alpha} + \hat{\beta}_1 \times x_1 + \hat{\beta}_2 \times x_2 + \ldots + \hat{\beta}_p \times x_p$$

Confidence intervals and tests (if you ever want to use a test!) also follow the same principles (roughly, estimate $\pm$ 1.96 $\times$ SD) as in multiple linear regression, except the formulae are a bit more complicated to write down, and, for purposes of this class, we will trust computer packages such as R to do the computations for us. So, let's see some examples of using multiple linear regression.

Example of Multiple Linear Regression (Polynomial Regression Example)

We continue with an example we have previously analyzed: You may recall that we have already seen the data below, which describe the tooth decay experience of 7257 children 12–14 years old in 21 communities according to the fluoride concentration of their public water supply. Remember that DMF denotes "Decayed, Missing or Filled Teeth".

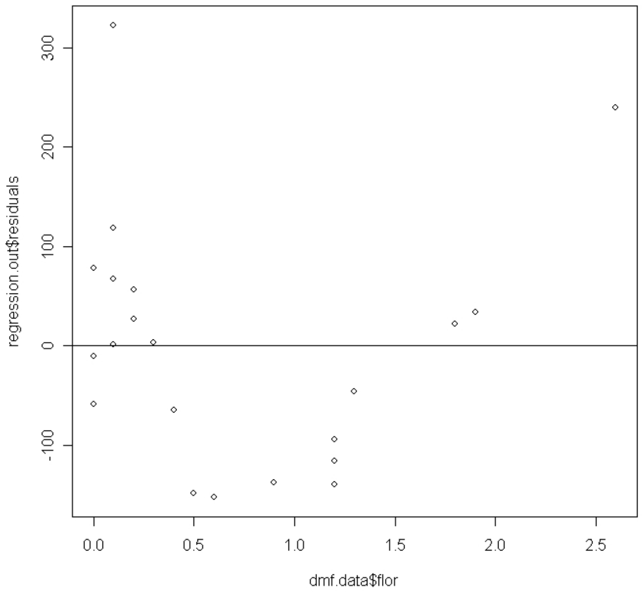
Community Number	DMF per 100 children	Fluoride Concentration in ppm
1	236	1.9
2	246	2.6
3	252	1.8
4	258	1.2
5	281	1.2
6	303	1.2
7	323	1.3
8	343	0.9
9	412	0.6
10	444	0.5
11	556	0.4
12	652	0.3
13	673	0.0
14	703	0.2
15	706	0.1
16	722	0.0
17	733	0.2
18	772	0.1
19	810	0.0
20	823	0.1
21	1027	0.1

Last time we ran simple linear regression on this data set, with the following results:

```
> regression.out<-lm(dmf~flor)
> summary(regression.out)

Call: lm(formula = dmf ~ flor)

Residuals:
     Min       1Q   Median       3Q      Max
-152.825  -94.305    1.575   56.495  322.575

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   732.34      38.55  18.996  8.1e-14
*** flor     -279.20      38.18  -7.312  6.2e-07 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 127.3 on 19 degrees of freedom Multiple
R-Squared: 0.7378,     Adjusted R-squared: 0.724 F-statistic: 53.47
```

```
on 1 and 19 DF,  p-value: 6.199e-07
```

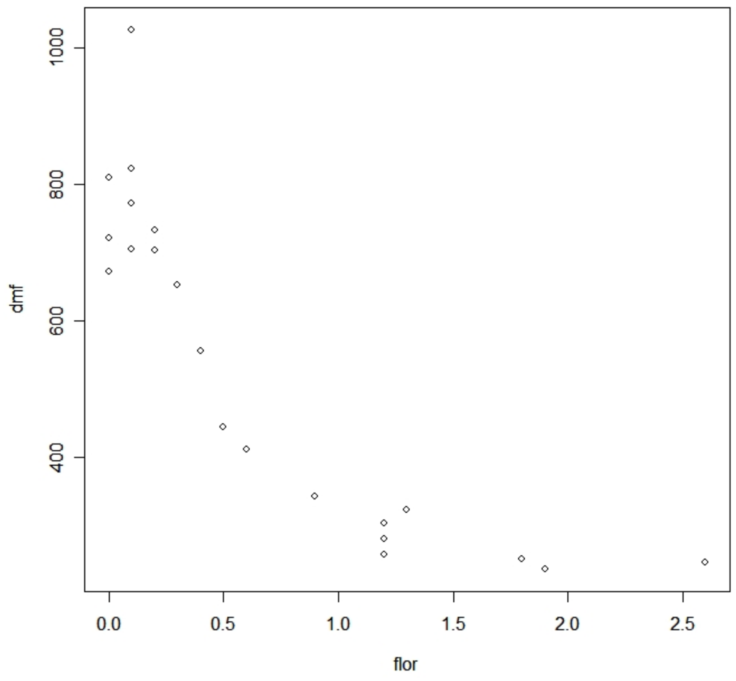
Note the very large standard errors for our parameter estimates, the large residuals indicating a poor fit, and the graph of residuals below:



The residuals form a "U-shape", suggesting that quadratic regression, or polynomial regression of order two, meaning adding an "X^2" or, in this case, "$flor^2$" term may substantially improve the fit.

We will now fit this quadratic multiple regression model using R. The commands we will use are:

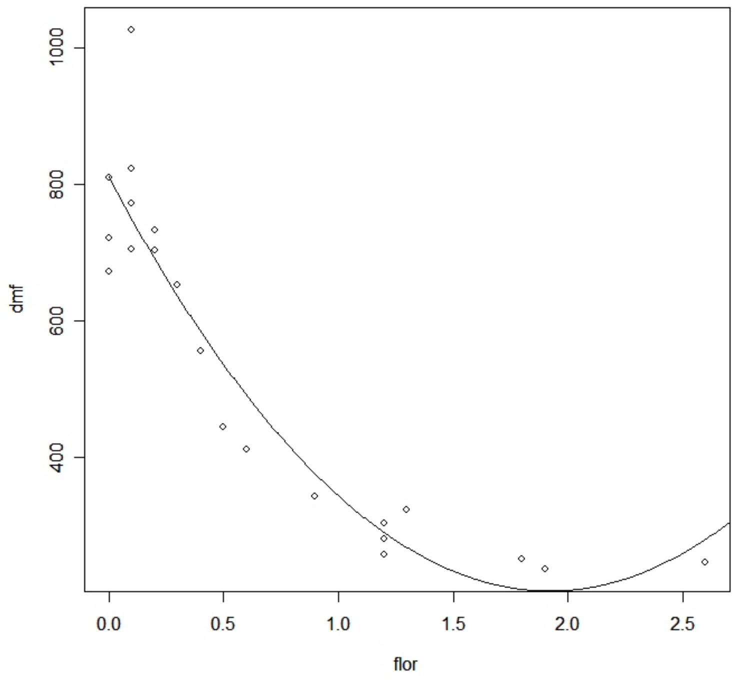
```
#  First enter the data

> dmf.data <- data.frame(
dmf = c( 236, 246, 252, 258, 281, 303, 323, 343, 412, 444, 556, 652,
673, 703, 706, 722, 733, 772, 810, 823, 1027),
flor =  c( 1.9, 2.6, 1.8, 1.2, 1.2, 1.2, 1.3, 0.9, 0.6, 0.5, 0.4, 0.3,
0.0, 0.2, 0.1, 0.0, 0.2, 0.1, 0.0, 0.1, 0.1))

> attach(dmf.data)  #  For convenience, to make the dmf and flor variables
>                   #  available outside the data.frame
>
> plot(flor, dmf)   #  Create a scatter plot
```



The graph and residual plot both strongly suggest a non-linear relationship between DMF teeth and fluoride. We will use R to add a squared fluoride term to the regression, to see if the fit is improved:

```
#  First create the fluoride squared variable:

> flor2 <- flor^2

#  Look at the flor2 variable, the square of flor:

> flor2
 [1] 3.61 6.76 3.24 1.44 1.44 1.44 1.69 0.81 0.36 0.25 0.16 0.09
     0.00 0.04 0.01 0.00 0.04 0.01 0.00 0.01 0.01

#  Next run the multiple linear (polynomial) regression model
#  with both linear and quadratic terms:

> regression.out<-lm(dmf ~ flor + flor2)

# Look at the main results of the regression:

> summary(regression.out)
```

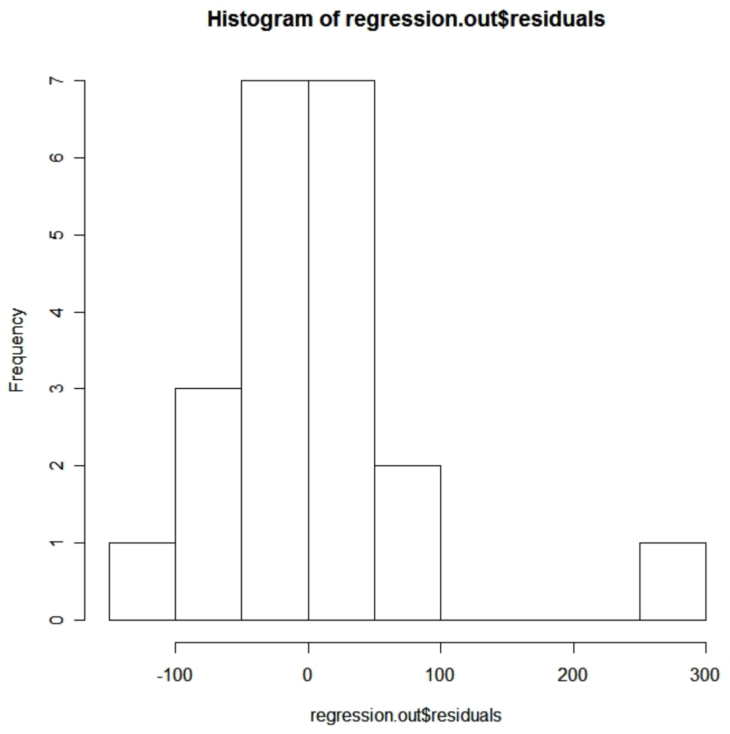
```
Call:
lm(formula = dmf ~ flor + flor2)

Residuals:
     Min       1Q   Median       3Q      Max
-138.510  -34.566   -1.510   31.240  277.030

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   811.51      31.55  25.721 1.20e-15 ***
flor         -631.85      79.91  -7.907 2.90e-07 ***
flor2         164.48      35.19   4.675 0.000189 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 87.91 on 18 degrees of freedom
Multiple R-Squared: 0.8816,     Adjusted R-squared: 0.8684
F-statistic:    67 on 2 and 18 DF,  p-value: 4.578e-09

#  Check the fit of this new regression as a plot against the data:

> x <- seq(0,3,by=0.005)
> plot(flor, dmf)
> points(x, regression.out$coefficients[1] + regression.out$coefficients[2]*x
     + regression.out$coefficients[3] * x^2, type="l")

#  regression.out$coefficients[1] = intercept
#  regression.out$coefficients[2] = slope for flor, the linear term
#  regression.out$coefficients[3] = slope for flor2, the quadratic term
```



Note the very dramatic improvement in fit when adding in the squared (quadratic) fluoride term.

Note also the much improved residuals:

```
> regression.out$residuals
         1          2          3          4          5          6
 31.240279 -34.566092  44.911693 -32.139684  -9.139684  12.860316
         7          8          9         10         11         12
 54.926005 -33.073852 -79.613673 -92.705981 -29.087806  15.240852
        13         14         15         16         17         18
-138.510276  11.279993 -43.970383 -89.510276  41.279993  22.029617
        19         20         21
 -1.510276  73.029617 277.029617
> hist(regression.out$residuals)
```



The residuals are almost perfectly normally distributed, with perhaps one "outlier" out on the right.

Before leaving this example, we note the following additional points:

Interpretation of coefficients in polynomial regression: While the intercept retains the same interpretation as in simple (univariate) linear regression, the coefficient for the linear term, $flor$, changes when the $flor2$ quadratic term is added to the model.

In the previous simple linear model, we have $\alpha = 732.34$, while in the quadratic polynomial regression model it was $\alpha = 811.41$. While these two values are somewhat different (the second estimate is better, since the quadratic model fit better), both estimates are saying roughly the same thing: When there is no fluoride added to the water in a community, there will be about 700 or 800 DMF teeth per 100 children. While these estimates differ numerically, they are both estimating the same quantity.

On the other hand, the beta coefficient for the linear $flor$ term changed from $\beta = -279.2$ in the simple linear model to $\beta = -631.85$ in the second model. This does **NOT** mean that for a one unit change in fluoride the second (better) model almost tripled the estimate of the benefits of fluoride, because these two estimates are **NOT** directly comparable in this way. The first estimate of $\beta = -279.2$ does indeed estimate the effect of a one unit change in $flor$ on DMF teeth, and is interpreted as meaning that for every additional unit of *flor*, mean DMF teeth decreases by about 279 per 100 children (although, as discussed previously, we know that estimate is not accurate because a linear model did not fit that well).

Conversely, the estimate of $\beta = -631.85$ does **NOT** mean that an increase of one unit of *flor* decreases DMF teeth by 631 per 100 children, because there is another *flor* term in the equation, and both must be considered simultaneously in order to infer the correct relationship between DMF and *flor*. In addition, because of the nonlinear nature of this relationship, **there is no unique "slope" that applies throughout the *flor* range.** Let's look at two examples of one unit *flor* changes, first going from 0 to 1, and then going from 1 to 2:

```
#  Prediction for flor=0:

> DMF0 <- regression.out$coefficients[1] + regression.out$coefficients[2]*0
    +  regression.out$coefficients[3] * 0^2

> DMF0
(Intercept)
   811.5103

#  Prediction for flor=1:

> DMF1 <- regression.out$coefficients[1] + regression.out$coefficients[2]*1
     +  regression.out$coefficients[3] * 1^2

> DMF1
   344.1396

#  Prediction for flor=2:

> DMF2 <- regression.out$coefficients[1] + regression.out$coefficients[2]*2
     +  regression.out$coefficients[3] * 2^2

> DMF2
   205.7207

#  Difference when changing flor from 0 to 1:

> DMF0 - DMF1
   467.3707

#  Difference when changing flor from 1 to 2:
```

```
> DMF1 - DMF2
  138.4190
```

So, a one unit change depends on where you start, it is not unique, unlike in simple regression models. As the values of $flor$ increases, the "slope" becomes less and less negative, so that the effect of adding more fluoride is diminished for larger values. This is also clear from the graph of the regression function above.

Beware of range: In any regression model, all conclusions apply only over the range of the collected data. So here, with a reasonably good fitting model, we can expect good predictions over the range, say, from 0 to 2.5 units of fluoride, but not above 2.5 (where the curve starts to rise again, which is not likely).

Predictions in multiple regression models: As shown above, predictions for multiple regression models can be done "manually" once the coefficients are known (or stored in an output object), but you can also do them more "automatically" in R:

```
> regression.out$fitted.values
       1        2        3        4        5        6        7        8
204.7597 280.5661 207.0883 290.1397 290.1397 290.1397 268.0740 376.0739
       9       10       11       12       13       14       15       16
491.6137 536.7060 585.0878 636.7591 811.5103 691.7200 749.9704 811.5103
      17       18       19       20       21
691.7200 749.9704 811.5103 749.9704 749.9704
```

or, for new values:

```
> new <- data.frame(flor=1.5, flor2=1.5^2)

#  Just the predicted mean
> predict.lm(regression.out, newdata=new)
[1] 233.8112

#  With CI for prediction for next city with value flor = 1.5
> predict.lm(regression.out, newdata=new, interval="prediction")
          fit     lwr      upr
[1,] 233.8112 36.3615 431.2608

#  With confidence interval for mean when flor=1.5
> predict.lm(regression.out, newdata=new, interval="confidence")
          fit      lwr      upr
[1,] 233.8112 164.0108 303.6116
```

The first interval is a prediction for the "next town" with a fluoride concentration of 1.5, while the second interval is the confidence interval for the mean prediction (i.e., for an infinite number of towns, each with value 1.5 ... note that second interval is much smaller than the first).

Regression coefficient confidence intervals from R: Confidence intervals for any estimated parameter can also be calculated manually, for example, we can calculate a confidence interval for β_1 (i.e., linear term for $flor$):

```
 Excerpt from above regression results were: Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   811.51      31.55  25.721 1.20e-15 ***
flor         -631.85      79.91  -7.907 2.90e-07 ***
flor2         164.48      35.19   4.675 0.000189 ***

So a CI for flor is:

> -631.85 + c(-1,1)*(qt(0.975, df=18))*79.91
[1] -799.7347 -463.9653
```

Below we will see how to automate these calculations.

Higher order polynomial regression models: There is no reason to stop at a quadratic model, in general one could go to higher orders in search of a better fit, had the residuals indicated this might be fruitful. For example, one could consider the model

$$DMF = \alpha + \beta_1 * flor + +\beta_2 * flor^2 + \beta_3 * flor^3$$

and so on.

Other functions of the X variables: Rather than polynomial terms, one can also try other terms, for example:

$$DMF = \alpha + \beta_1 * flor + \beta_2 * \sqrt{flor}$$

or

$$DMF = \alpha + \beta * \log(flor)$$

and so on.

R program for confidence intervals for multiple regression

Once again, note that, R does not automatically provide confidence intervals for all parameters in the summary, although it does provide all of the information needed to obtain these confidence intervals, or one can use the `confint` function. For convenience, as we did for simple linear regression, we will create our own function to output all of the usual information from a regression, plus confidence intervals.

Unlike simple linear regression, we cannot know in advance how many independent (X) variables we will want to use in our function. Therefore, at the beginning, we will discover the number of beta coefficients we need to estimate, and automatically get confidence intervals for all of them.

Here is the function:

```
multiple.regression.with.ci <- function(regress.out, level=0.95)
{
################################################################
#                                                              #
#  This function takes the output from an lm                   #
#  (linear model) command in R and provides not                #
#  only the usual output from the summary command, but         #
#  adds confidence intervals for intercept and slope.          #
#                                                              #
#  This version accommodates multiple regression parameters    #
#                                                              #
################################################################
usual.output <- summary(regress.out)
t.quantile <- qt(1-(1-level)/2, df=regress.out$df)
number.vars <- length(regress.out$coefficients)
temp.store.result <- matrix(rep(NA, number.vars*2), nrow=number.vars)
for(i in 1:number.vars)
{
     temp.store.result[i,] <- summary(regress.out)$coefficients[i] +
      c(-1, 1) * t.quantile * summary(regress.out)$coefficients[i+number.vars]
}
  intercept.ci <- temp.store.result[1,]
  slopes.ci <- temp.store.result[-1,]
  output <- list(regression.table = usual.output, intercept.ci = intercept.ci,
              slopes.ci = slopes.ci)
return(output)
}
```

```
#  Now test the function:

> multiple.regression.with.ci(regression.out)
$regression.table

Call:
lm(formula = dmf ~ flor + flor2)

Residuals:
     Min       1Q   Median       3Q      Max
-138.510  -34.566   -1.510   31.240  277.030

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   811.51      31.55  25.721 1.20e-15 ***
flor         -631.85      79.91  -7.907 2.90e-07 ***
flor2         164.48      35.19   4.675 0.000189 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 87.91 on 18 degrees of freedom
Multiple R-Squared: 0.8816,     Adjusted R-squared: 0.8684
F-statistic:    67 on 2 and 18 DF,  p-value: 4.578e-09

$intercept.ci
[1] 745.2254 877.7952

$slopes.ci
           [,1]      [,2]
[1,] -799.73982 -463.9532
[2,]   90.55367  238.3980
```

You may wish to modify the function to create "prettier" output, but it works.

Another example of multiple linear regression

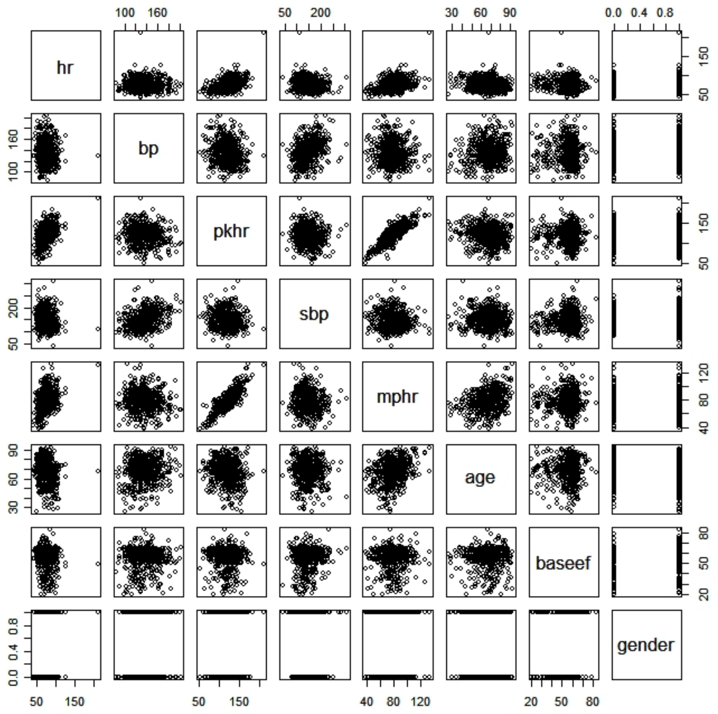
The above example was really just a special case of multiple linear regression, because we used only a single variable, fluoride, which was used multiple times (just twice, in this case, linear and quadratic terms) in the same equation. We will now see a more general example of multiple regression, involving several variables.

Consider predicting the percent heart rate achieved from other basic heart related measurements. We will use the data set from the course web page called "Heart-Data.txt". Definitions of the variables included are:

hr	basal heart rate
bp	basal blood pressure
pkhr	peak heart rate
sbp	systolic blood pressure
mphr	percentage of maximum predicted heart rate achieved
age	age
gender	gender (male = 0, female=1)
baseef	baseline cardiac ejection fraction (measures heart pumping efficiency)

We will see if we can predict mphr from the other variables using multiple linear regression. Save the data set on your hard disk, and run the following R program to obtain the results:

```
#  Input data set as a data frame in R

> heart <- read.table(file="g:\\HeartData.txt", header=T)

#  Check first few lines of data set

> heart[1:10,]
   hr  bp pkhr sbp mphr age baseef gender
1  92 103  114  86   74  85     27      0
2  62 139  120 158   82  73     39      0
3  62 139  120 157   82  73     39      0
4  93 118  118 105   72  57     42      1
5  89 103  129 173   69  34     45      0
6  58 100  123 140   83  71     46      0
7  63 120   98 130   71  81     48      1
8  86 161  144 157  111  90     50      1
9  69 143  115 118   81  81     52      1
10 76 105  126 125   94  86     52      0

#  Quick scatter plots of all variables

> pairs(heart)
```



Note that some variables look strongly related, e.g., mphr and pkhr (probably not a surprise).

```
#  Run a multiple linear regression with these variables, using mphr as the
#  dependent variable, and all others as independent variables:

> regression.out <- lm(mphr ~ hr + bp + pkhr + sbp + age + baseef + gender, data=heart)

#  Now run our recently developed function to get output with CIs:

> multiple.regression.with.ci(regression.out)
$regression.table

Call:
lm(formula = mphr ~ hr + bp + pkhr + sbp + age + baseef + gender,
    data = heart)

Residuals:
     Min       1Q   Median       3Q      Max
-21.3654  -1.6474   0.2319   1.6702  24.3196

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -33.712293   2.374200 -14.199  < 2e-16 ***
hr            0.035350   0.014563   2.427  0.01553 *
bp           -0.030073   0.010025  -3.000  0.00282 **
pkhr          0.603127   0.009739  61.928  < 2e-16 ***
sbp           0.032667   0.005682   5.749 1.49e-08 ***
age           0.525863   0.016094  32.674  < 2e-16 ***
baseef        0.010269   0.018996   0.541  0.58902
gender        0.328743   0.399230   0.823  0.41061
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.405 on 550 degrees of freedom
Multiple R-Squared: 0.9162,     Adjusted R-squared: 0.9152
F-statistic: 859.2 on 7 and 550 DF,  p-value: < 2.2e-16


$intercept.ci
[1] -38.37590 -29.04868

$slopes.ci
             [,1]        [,2]
[1,]  0.006743777  0.06395708
[2,] -0.049765950 -0.01038055
[3,]  0.583996404  0.62225759
[4,]  0.021505285  0.04382926
[5,]  0.494249025  0.55747657
[6,] -0.027045455  0.04758352
[7,] -0.455459465  1.11294500
```

Take a few minutes to look at each variable, especially at the confidence intervals, and interpret the effect from each variable. Keep in mind the correct way to interpret confidence intervals, according to where the upper and lower interval limits fall in relation to the region of clinical equivalence.

In the above analyses, we simply took all possible variables, and ran a single linear regression model with all variables included. This is, in fact, very bad practice. In general, a better procedure would be something like:

1. Look at various descriptive statistics to get a feel for the data.
2. Separately, graph each independent variable against each dependent variable (sometimes more than one dependent variable is of interest. This allows one to see if trends are perhaps non-linear (possibly leading to data transformations for either X or Y variables), as well as whether the variable looks "important" (but remember that looks can sometimes be deceiving).

3. For all variables being considered, calculate a correlation matrix of each variable against each other variable. This allows one to begin to investigate possible confounding and collinearity.

4. Perform a simple linear regression for each independent variable. This again begins to investigate confounding, as well as providing an initial "unadjusted" view of the importance of each variable, by itself.

5. Think about any "interaction terms" that you may want to try in the model.

6. Perform some sort of model selection technique, or, often much better, think about avoiding any strict model selection by finding a set of models that seem to have something to contribute to overall conclusions.

7. Based on all work done, draw some inferences and conclusions. Carefully interpret each estimated parameter, perform "model criticism", possibly repeating some of the above steps (for example, run further models), as needed.

8. Other inferences, such as predictions for future observations, and so on.

The above should not be looked at as "rules" never to be broken, but can be used as a rough guide as to how to proceed through a regression analysis.

In the next few lectures, we will carefully investigate each of these issues using this same data set and others.