REGRESSION WITH STATA

BY XIAO CHEN, PHILIP B. ENDER, MICHAEL MITCHELL AND CHRISTINE WELLS (IN ALPHABETICAL ORDER)

The aim of these materials is to help you increase your skills in using regression analysis with Stata. This web book does not teach regression, per se, but focuses on how to perform regression analyses using Stata. It is assumed that you have had at least a one quarter/semester course in regression (linear models) or a general statistical methods course that covers simple and multiple regression and have access to a regression textbook that explains the theoretical background of the materials covered in these chapters. These materials also assume you are familiar with using Stata, for example that you have taken the Introduction to Stata class or have equivalent knowledge of Stata.

Source: http://www.ats.ucla.edu/stat/stata/webbooks/reg/

Book Chapters

Book Chapters and Outline

Chapter 1 - Simple and Multiple Regression

Chapter Outline

1.0 Introduction

This book is composed of four chapters covering a variety of topics about using Stata for regression. We should emphasize that this book is about "data analysis" and that it demonstrates how Stata can be used for regression analysis, as opposed to a book that covers the statistical basis of multiple regression. We assume that you have had at least one statistics course covering regression analysis and that you have a regression book that you can use as a reference (see the Regression With Stata page and our Statistics Books for Loan page for recommended regression analysis books). This book is designed to apply your knowledge of regression, combine it with instruction on Stata, to perform, understand and interpret regression analyses.

This first chapter will cover topics in simple and multiple regression, as well as the supporting tasks that are important in preparing to analyze your data, e.g., data checking, getting familiar with your data file, and examining the distribution of your variables. We will illustrate the basics of simple and multiple regression and demonstrate the importance of inspecting, checking and verifying your data before accepting the results of your analysis. In general, we hope to show that the results of your regression analysis can be misleading without further probing of your data, which could reveal relationships that a casual analysis could overlook.

In this chapter, and in subsequent chapters, we will be using a data file that was created by randomly sampling 400 elementary schools from the California Department of Education's API 2000 dataset. This data file contains a measure of school academic performance as well as other attributes of the elementary schools, such as, class size, enrollment, poverty, etc.

You can access this data file over the web from within Stata with the Stata **use** command as shown below. **Note:** Do not type the leading dot in the command -- the dot is a convention to indicate that the statement is a Stata command.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi
```

Once you have read the file, you probably want to store a copy of it on your computer (so you don't need to read it over the web every time). Let's say you are using Windows and want to store the file in a folder called **c:\regstata** (you can choose a different name if you like). First, you can make this folder within Stata using the **mkdir** command.

```
mkdir c:\regstata
```

We can then change to that directory using the **cd** command.

```
cd c:\regstata
```

And then if you save the file it will be saved in the **c:\regstata** folder. Let's save the file as **elemapi** .

```
save elemapi
```

Now the data file is saved as **c:\regstata\elemapi.dta** and you could quit Stata and the data file would still be there. When you wish to use the file in the future, you would just use the **cd** command to change to the **c:\regstata** directory (or whatever you called it) and then **use** the **elemapi** file.

```
cd c:\regstata
use elemapi
```

1.1 A First Regression Analysis

Let's dive right in and perform a regression analysis using the variables **api00**, **acs_k3**, **meals** and **full**. These measure the academic performance of the school (**api00**), the average class size in kindergarten through 3rd grade (**acs_k3**), the percentage of students receiving free meals (**meals**) - which is an indicator of poverty, and the percentage of teachers who have full teaching credentials (**full**). We expect that better academic performance would be associated with lower class size, fewer students receiving free meals, and a higher percentage of teachers having full teaching credentials. Below, we show the Stata command for testing this regression model followed by the Stata output.

```
regress api00 acs_k3 meals full

      Source |       SS       df       MS              Number of obs =
313
-------------+------------------------------           F(  3,   309) =
213.41
       Model |  2634884.26     3  878294.754           Prob > F      =
0.0000
    Residual |  1271713.21   309  4115.57673           R-squared     =
0.6745
-------------+------------------------------           Adj R-squared =
0.6713
       Total |  3906597.47   312  12521.1457           Root MSE      =
64.153

------------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-------
      acs_k3 |  -2.681508   1.393991    -1.92   0.055    -5.424424
.0614073
       meals |  -3.702419   .1540256   -24.04   0.000    -4.005491   -
3.399348
        full |   .1086104    .090719     1.20   0.232    -.0698947
.2871154
       _cons |   906.7392   28.26505    32.08   0.000     851.1228
962.3555
------------------------------------------------------------------------------
-------
```

Let's focus on the three predictors, whether they are statistically significant and, if so, the direction of the relationship. The average class size (**acs_k3**, b=-2.68), is not statistically significant at the 0.05 level (p=0.055), but only just so. The coefficient is negative which would indicate that larger class size is related to lower academic performance -- which is what we would expect. Next, the effect of **meals** (b=-3.70, p=.000) is significant and its coefficient is negative indicating that the greater the proportion students receiving free meals, the lower the academic performance. Please note, that we are not saying that free meals are causing lower academic performance. The **meals** variable is highly related to income level and functions more as a proxy for poverty. Thus, higher levels of poverty are associated with lower academic performance. This result also makes sense. Finally, the percentage of teachers with full credentials (**full**, b=0.11, p=.232) seems to be unrelated to academic performance. This would seem to indicate that the percentage of teachers with full credentials is not an important factor in predicting academic performance -- this result was somewhat unexpected.

Should we take these results and write them up for publication? From these results, we would conclude that lower class sizes are related to higher performance, that fewer students receiving free meals is associated with higher performance, and that the percentage of teachers with full credentials was not related to academic performance in the schools. Before we write this up for publication, we should do a number of checks to make sure we can firmly stand behind these results. We start by getting more familiar with the data file, doing preliminary data checking, looking for errors in the data.

1.2 Examining data

First, let's use the **describe** command to learn more about this data file. We can verify how many observations it has and see the names of the variables it contains. To do this, we simply type

```
describe
```

```
Contains data from
http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi.dta
  obs:           400
 vars:            21                          25 Feb 2001 16:58
 size:        14,800 (92.3% of memory free)
-----------------------------------------------------------------------
--------
              storage  display     value
variable name   type   format      label      variable label
-----------------------------------------------------------------------
--------
snum            int    %9.0g                  school number
dnum            int    %7.0g       dname      district number
api00           int    %6.0g                  api 2000
api99           int    %6.0g                  api 1999
growth          int    %6.0g                  growth 1999 to 2000
meals           byte   %4.0f                  pct free meals
ell             byte   %4.0f                  english language learners
yr_rnd          byte   %4.0f       yr_rnd     year round school
mobility        byte   %4.0f                  pct 1st year in school
```

```
acs_k3          byte   %4.0f                      avg class size k-3
acs_46          byte   %4.0f                      avg class size 4-6
not_hsg         byte   %4.0f                      parent not hsg
hsg             byte   %4.0f                      parent hsg
some_col        byte   %4.0f                      parent some college
col_grad        byte   %4.0f                      parent college grad
grad_sch        byte   %4.0f                      parent grad school
avg_ed          float  %9.0g                      avg parent ed
full            float  %4.0f                      pct full credential
emer            byte   %4.0f                      pct emer credential
enroll          int    %9.0g                      number of students
mealcat         byte   %18.0g      mealcat    Percentage free meals in
3
                                                categories
-----------------------------------------------------------------------
--------
Sorted by:  dnum
```

We will not go into all of the details of this output. Note that there are 400 observations and 21 variables. We have variables about academic performance in 2000 and 1999 and the change in performance, **api00**, **api99** and **growth** respectively. We also have various characteristics of the schools, e.g., class size, parents education, percent of teachers with full and emergency credentials, and number of students. Note that when we did our original regression analysis it said that there were 313 observations, but the **describe** command indicates that we have 400 observations in the data file.

If you want to learn more about the data file, you could **list** all or some of the observations. For example, below we **list** the first five observations.

```
list in 1/5

Observation 1

        snum          906         dnum            41       api00
693
       api99          600       growth            93       meals
67
         ell            9       yr_rnd            No    mobility
11
      acs_k3           16       acs_46            22     not_hsg
0
         hsg            0     some_col             0    col_grad
0
    grad_sch            0       avg_ed             .        full
76.00
        emer           24       enroll           247     mealcat 47-80%
free


Observation 2

        snum          889         dnum            41       api00
570
```

```
      api99         501      growth         69        meals
92
        ell          21      yr_rnd         No     mobility
33
     acs_k3          15      acs_46         32      not_hsg
0
        hsg           0    some_col          0     col_grad
0
   grad_sch           0      avg_ed          .         full
79.00
       emer          19      enroll        463      mealcat 81-100%
free
```

Observation 3

```
       snum         887        dnum         41        api00
546
      api99         472      growth         74        meals
97
        ell          29      yr_rnd         No     mobility
36
     acs_k3          17      acs_46         25      not_hsg
0
        hsg           0    some_col          0     col_grad
0
   grad_sch           0      avg_ed          .         full
68.00
       emer          29      enroll        395      mealcat 81-100%
free
```

Observation 4

```
       snum         876        dnum         41        api00
571
      api99         487      growth         84        meals
90
        ell          27      yr_rnd         No     mobility
27
     acs_k3          20      acs_46         30      not_hsg
36
        hsg          45    some_col          9     col_grad
9
   grad_sch           0      avg_ed       1.91         full
87.00
       emer          11      enroll        418      mealcat 81-100%
free
```

Observation 5

```
       snum         888        dnum         41        api00
478
      api99         425      growth         53        meals
89
```

```
          ell           30       yr_rnd           No    mobility
44
       acs_k3           18       acs_46           31     not_hsg
50
          hsg           50     some_col            0    col_grad
0
     grad_sch            0       avg_ed          1.5        full
87.00
         emer           13       enroll          520     mealcat 81-100%
free
```

This takes up lots of space on the page, but does not give us a lot of information. Listing our data can be very helpful, but it is more helpful if you **list** just the variables you are interested in. Let's **list** the first 10 observations for the variables that we looked at in our first regression analysis.

```
list api00 acs_k3 meals full  in 1/10

      api00  acs~3  meals      full
  1.    693     16     67     76.00
  2.    570     15     92     79.00
  3.    546     17     97     68.00
  4.    571     20     90     87.00
  5.    478     18     89     87.00
  6.    858     20      .    100.00
  7.    918     19      .    100.00
  8.    831     20      .     96.00
  9.    860     20      .    100.00
 10.    737     21     29     96.00
```

We see that among the first 10 observations, we have four missing values for **meals**. It is likely that the missing data for **meals** had something to do with the fact that the number of observations in our first regression analysis was 313 and not 400.

Another useful tool for learning about your variables is the **codebook** command. Let's do **codebook** for the variables we included in the regression analysis, as well as the variable **yr_rnd**. We have interspersed some comments on this output in **[square brackets and in bold]**.

```
codebook api00 acs_k3 meals full yr_rnd

api00 ----------------------------------------------------------------
api 2000
                  type:  numeric (int)

                 range:  [369,940]                    units:  1
         unique values:  271                  coded missing:  0 / 400

                  mean:   647.622
              std. dev:   142.249

           percentiles:        10%       25%       50%       75%
90%
```

```
                            465.5     523.5       643     762.5
850
```
[the api scores don't have any missing values, and range from 369-940]
[this makes sense since the api scores can range from 200 to 1000]

```
acs_k3 ----------------------------------------------------- avg class
size k-3
                  type:  numeric (byte)

                 range:  [-21,25]                     units:  1
         unique values:  14                   coded missing:  2 / 400

                  mean:   18.5477
              std. dev:   5.00493

           percentiles:        10%       25%       50%       75%
90%
                               17        18        19        20
21
```
[the average class size ranges from -21 to 25 and 2 are missing.]
[it seems odd for a class size to be -21]

```
meals --------------------------------------------------------- pct
free meals
                  type:  numeric (byte)

                 range:  [6,100]                      units:  1
         unique values:  80                   coded missing:  85 / 400

                  mean:   71.9937
              std. dev:   24.3856

           percentiles:        10%       25%       50%       75%
90%
                               33        57        77        93
99
```
[the percent receiving free meals ranges from 6 to 100, but 85 are missing]
[this seems like a large number of missing values!]

```
full ------------------------------------------------------ pct full
credential
                  type:  numeric (float)

                 range:  [.42,100]                    units:  .01
         unique values:  92                   coded missing:  0 / 400

                  mean:   66.0568
              std. dev:   40.2979

           percentiles:        10%       25%       50%       75%
90%
67       .95        87        97       100
```
[The percent credentialed ranges from .42 to 100 with no missing]

```
yr_rnd ------------------------------------------------------ year
round school
                  type:  numeric (byte)
                 label:  yr_rnd

                 range:  [0,1]                        units:  1
         unique values:  2                    coded missing:  0 / 400

            tabulation:  Freq.   Numeric  Label
                           308         0  No
                            92         1  Yes
```

[the variable yr_rnd is coded 0=No (not year round) and 1=Yes (year round)]
[308 are non-year round and 92 are year round, and none are missing]

The codebook command has uncovered a number of peculiarities worthy of further examination. Let's use the **summarize** command to learn more about these variables. As shown below, the **summarize** command also reveals the large number of missing values for **meals** (400 - 315 = 85) and we see the unusual minimum for **acs_k3** of -21.

summarize api00 acs_k3 meals full

```
    Variable |     Obs        Mean   Std. Dev.       Min        Max
-------------+-----------------------------------------------------
       api00 |     400    647.6225    142.249        369        940
      acs_k3 |     398    18.54774   5.004933        -21         25
       meals |     315    71.99365   24.38557          6        100
        full |     400     66.0568   40.29793        .42        100
```

Let's get a more detailed summary for **acs_k3**. In Stata, the comma after the variable list indicates that options follow, in this case, the option is **detail**. As you can see below, the **detail** option gives you the percentiles, the four largest and smallest values, measures of central tendency and variance, etc. Note that **summarize**, and other commands, can be abbreviated: we could have typed **sum acs_k3, d**.

summarize acs_k3, detail

```
                     avg class size k-3
-------------------------------------------------------------
      Percentiles      Smallest
 1%          -20            -21
 5%           16            -21
10%           17            -21       Obs                 398
25%           18            -20       Sum of Wgt.         398

50%           19                      Mean           18.54774
                        Largest       Std. Dev.      5.004933
75%           20             23
90%           21             23       Variance       25.04935
95%           21             23       Skewness      -7.078785
99%           23             25       Kurtosis       55.33497
```

It seems as though some of the class sizes somehow became negative, as though a negative sign was incorrectly typed in front of them. Let's do a **tabulate** of class size to see if this seems plausible.

```
tabulate acs_k3

  avg class |
   size k-3 |      Freq.     Percent        Cum.
------------+-----------------------------------
        -21 |          3        0.75        0.75
        -20 |          2        0.50        1.26
        -19 |          1        0.25        1.51
         14 |          2        0.50        2.01
         15 |          1        0.25        2.26
         16 |         14        3.52        5.78
         17 |         20        5.03       10.80
         18 |         64       16.08       26.88
         19 |        143       35.93       62.81
         20 |         97       24.37       87.19
         21 |         40       10.05       97.24
         22 |          7        1.76       98.99
         23 |          3        0.75       99.75
         25 |          1        0.25      100.00
------------+-----------------------------------
      Total |        398      100.00
```

Indeed, it seems that some of the class sizes somehow got negative signs put in front of them. Let's look at the school and district number for these observations to see if they come from the same district. Indeed, they all come from district 140.

```
list snum dnum acs_k3 if acs_k3 < 0

          snum     dnum  acs~3
 37.       602      140   -21
 96.       600      140   -20
173.       595      140   -21
223.       596      140   -19
229.       611      140   -20
282.       592      140   -21
```

Let's look at all of the observations for district 140.

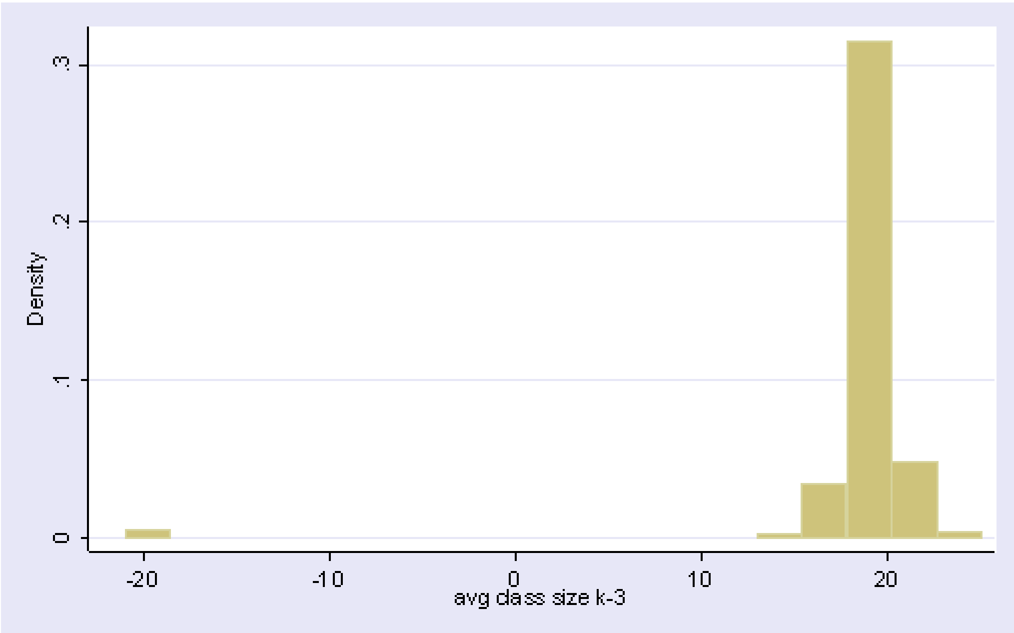
```
list dnum snum api00 acs_k3 meals full if dnum == 140

        dnum       snum   api00  acs~3  meals      full
 37.     140        602     864    -21      .    100.00
 96.     140        600     843    -20      .     91.00
173.     140        595     713    -21     63     92.00
223.     140        596     800    -19      .     94.00
229.     140        611     857    -20      .    100.00
282.     140        592     804    -21      .     97.00
```

All of the observations from district 140 seem to have this problem. When you find such a problem, you want to go back to the original source of the data to verify the values. We have to reveal that we fabricated this error for illustration purposes, and that the actual data had no such problem. Let's pretend that we checked with district 140 and there was a problem with the data there, a hyphen was accidentally put in front of the class sizes making them negative. We will make a note to fix this! Let's continue checking our data.

Let's take a look at some graphical methods for inspecting data. For each variable, it is useful to inspect them using a histogram, boxplot, and stem-and-leaf plot. These graphs can show you information about the shape of your variables better than simple numeric statistics can. We already know about the problem with **acs_k3**, but let's see how these graphical methods would have revealed the problem with this variable.

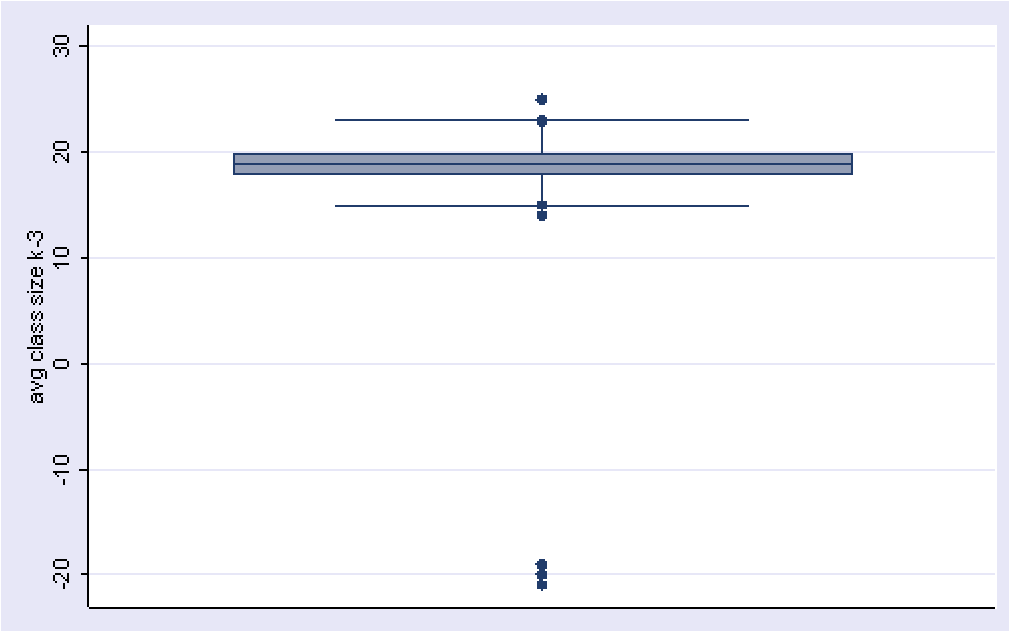
First, we show a histogram for **acs_k3**. This shows us the observations where the average class size is negative.

```
histogram acs_k3
```



Likewise, a boxplot would have called these observations to our attention as well. You can see the outlying negative observations way at the bottom of the boxplot.

```
graph box acs_k3
```



Finally, a stem-and-leaf plot would also have helped to identify these observations. This plot shows the exact values of the observations, indicating that there were three -21s, two -20s, and one -19.

```
stem acs_k3

Stem-and-leaf plot for acs_k3 (avg class size k-3)

 -2* | 11100
 -1. | 9
 -1s |
 -1f |
 -1t |
 -1* |
 -0. |
 -0s |
 -0f |
 -0t |
 -0* |
  0* |
  0t |
  0f |
  0s |
  0. |
  1* |
  1t |
  1f | 445
  1s | 6666666666666677777777777777777777
  1. | 888888888888888888888888888888888888888888888888888888888888888
... (207)
  2* | 000000000000000000000000000000000000000000000000000000000000000
... (137)
  2t | 2222222333
```

```
  2f | 5
```

We recommend plotting all of these graphs for the variables you will be analyzing. We will omit, due to space considerations, showing these graphs for all of the variables. However, in examining the variables, the stem-and-leaf plot for **full** seemed rather unusual. Up to now, we have not seen anything problematic with this variable, but look at the stem and leaf plot for **full** below. It shows 104 observations where the percent with a full credential is less than one. This is over 25% of the schools, and seems very unusual.

stem full

```
Stem-and-leaf plot for full (pct full credential)

full rounded to nearest multiple of .1
plot in units of .1

   0** | 04,04,05,05,05,05,05,05,05,05,05,05,06,06,06,06,06,06,06,06,
... (104)
   0** |
   0** |
   0** |
   0** |
   1** |
   1** |
   1** |
   1** |
   1** |
   2** |
   2** |
   2** |
   2** |
   2** |
   3** |
   3** |
   3** |
   3** | 70
   3** |
   4** | 10
   4** |
   4** | 40,40,50,50
   4** | 60
   4** | 80
   5** |
   5** | 30
   5** |
   5** | 70
   5** | 80,80,80,90
   6** | 10
   6** | 30,30
   6** | 40,50
   6** |
   6** | 80,80,90,90,90
   7** | 00,10,10,10
   7** | 20,30,30
   7** | 40,50,50,50,50
```

```
   7** | 60,60,60,60,70,70
   7** | 80,80,80,80,90,90,90
   8** | 00,00,00,00,00,00,00,00,00,00,10,10,10,10
   8** | 20,20,20,30,30,30,30,30,30,30,30,30
   8** | 40,40,40,40,50,50,50,50,50,50,50,50
   8** | 60,60,60,60,60,70,70,70,70,70,70,70,70,70,70,70,70
   8** | 80,80,80,80,80,80,90,90,90,90,90
   9** | 00,00,00,00,00,00,00,00,00,10,10,10,10,10,10,10,10
   9** | 20,20,20,20,20,20,20,30,30,30,30,30,30,30,30,30,30,30,30
   9** | 40,40,40,40,40,40,40,40,40,40,50,50,50,50,50,50,50,50,50,50,
... (27)
   9** | 60,60,60,60,60,60,60,60,60,60,60,60,60,60,60,60,60,70,70,70,
... (28)
   9** | 80,80,80,80,80,80,80,80,80
  10** | 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
... (81)
```

Let's look at the frequency distribution of **full** to see if we can understand this better. The values go from 0.42 to 1.0, then jump to 37 and go up from there. It appears as though some of the percentages are actually entered as proportions, e.g., 0.42 was entered instead of 42 or 0.96 which really should have been 96.

tabulate full

```
   pct full |
 credential |      Freq.     Percent        Cum.
------------+-----------------------------------
       0.42 |          1        0.25        0.25
       0.45 |          1        0.25        0.50
       0.46 |          1        0.25        0.75
       0.47 |          1        0.25        1.00
       0.48 |          1        0.25        1.25
       0.50 |          3        0.75        2.00
       0.51 |          1        0.25        2.25
       0.52 |          1        0.25        2.50
       0.53 |          1        0.25        2.75
       0.54 |          1        0.25        3.00
       0.56 |          2        0.50        3.50
       0.57 |          2        0.50        4.00
       0.58 |          1        0.25        4.25
       0.59 |          3        0.75        5.00
       0.60 |          1        0.25        5.25
       0.61 |          4        1.00        6.25
       0.62 |          2        0.50        6.75
       0.63 |          1        0.25        7.00
       0.64 |          3        0.75        7.75
       0.65 |          3        0.75        8.50
       0.66 |          2        0.50        9.00
       0.67 |          6        1.50       10.50
       0.68 |          2        0.50       11.00
       0.69 |          3        0.75       11.75
       0.70 |          1        0.25       12.00
       0.71 |          1        0.25       12.25
       0.72 |          2        0.50       12.75
       0.73 |          6        1.50       14.25
```

0.75	4	1.00	15.25
0.76	2	0.50	15.75
0.77	2	0.50	16.25
0.79	3	0.75	17.00
0.80	5	1.25	18.25
0.81	8	2.00	20.25
0.82	2	0.50	20.75
0.83	2	0.50	21.25
0.84	2	0.50	21.75
0.85	3	0.75	22.50
0.86	2	0.50	23.00
0.90	3	0.75	23.75
0.92	1	0.25	24.00
0.93	1	0.25	24.25
0.94	2	0.50	24.75
0.95	2	0.50	25.25
0.96	1	0.25	25.50
1.00	2	0.50	26.00
37.00	1	0.25	26.25
41.00	1	0.25	26.50
44.00	2	0.50	27.00
45.00	2	0.50	27.50
46.00	1	0.25	27.75
48.00	1	0.25	28.00
53.00	1	0.25	28.25
57.00	1	0.25	28.50
58.00	3	0.75	29.25
59.00	1	0.25	29.50
61.00	1	0.25	29.75
63.00	2	0.50	30.25
64.00	1	0.25	30.50
65.00	1	0.25	30.75
68.00	2	0.50	31.25
69.00	3	0.75	32.00
70.00	1	0.25	32.25
71.00	3	0.75	33.00
72.00	1	0.25	33.25
73.00	2	0.50	33.75
74.00	1	0.25	34.00
75.00	4	1.00	35.00
76.00	4	1.00	36.00
77.00	2	0.50	36.50
78.00	4	1.00	37.50
79.00	3	0.75	38.25
80.00	10	2.50	40.75
81.00	4	1.00	41.75
82.00	3	0.75	42.50
83.00	9	2.25	44.75
84.00	4	1.00	45.75
85.00	8	2.00	47.75
86.00	5	1.25	49.00
87.00	12	3.00	52.00
88.00	6	1.50	53.50
89.00	5	1.25	54.75
90.00	9	2.25	57.00
91.00	8	2.00	59.00
92.00	7	1.75	60.75

```
      93.00 |         12        3.00       63.75
      94.00 |         10        2.50       66.25
      95.00 |         17        4.25       70.50
      96.00 |         17        4.25       74.75
      97.00 |         11        2.75       77.50
      98.00 |          9        2.25       79.75
     100.00 |         81       20.25      100.00
------------+-----------------------------------
      Total |        400      100.00
```

Let's see which district(s) these data came from.

```
tabulate dnum if full <= 1

   district |
     number |      Freq.     Percent        Cum.
------------+-----------------------------------
        401 |        104      100.00      100.00
------------+-----------------------------------
      Total |        104      100.00
```

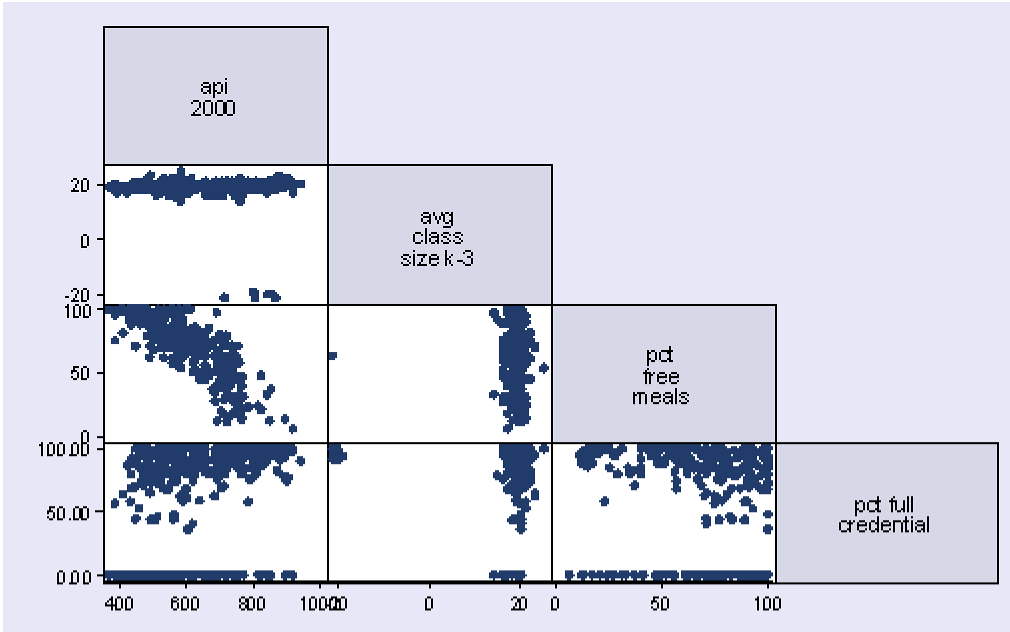
We note that all 104 observations in which **full** was less than or equal to one came from district 401. Let's count how many observations there are in district 401 using the **count** command and we see district 401 has 104 observations.

```
count if dnum==401
  104
```

All of the observations from this district seem to be recorded as proportions instead of percentages. Again, let us state that this is a pretend problem that we inserted into the data for illustration purposes. If this were a real life problem, we would check with the source of the data and verify the problem. We will make a note to fix this problem in the data as well.

Another useful graphical technique for screening your data is a scatterplot matrix. While this is probably more relevant as a diagnostic tool searching for non-linearities and outliers in your data, it can also be a useful data screening tool, possibly revealing information in the joint distributions of your variables that would not be apparent from examining univariate distributions. Let's look at the scatterplot matrix for the variables in our regression model. This reveals the problems we have already identified, i.e., the negative class sizes and the percent full credential being entered as proportions.

```
graph matrix api00 acs_k3 meals full, half
```



We have identified three problems in our data. There are numerous missing values for **meals**, there were negatives accidentally inserted before some of the class sizes (**acs_k3**) and over a quarter of the values for **full** were proportions instead of percentages. The corrected version of the data is called **elemapi2**. Let's use that data file and repeat our analysis and see if the results are the same as our original analysis. First, let's repeat our original regression analysis below.

```
regress api00 acs_k3 meals full

      Source |       SS       df       MS              Number of obs =
313
-------------+------------------------------           F(  3,   309) =
213.41
       Model |  2634884.26     3  878294.754           Prob > F      =
0.0000
    Residual |  1271713.21   309  4115.57673           R-squared     =
0.6745
-------------+------------------------------           Adj R-squared =
0.6713
       Total |  3906597.47   312  12521.1457           Root MSE      =
64.153

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      acs_k3 |  -2.681508   1.393991    -1.92   0.055    -5.424424
.0614073
       meals |  -3.702419   .1540256   -24.04   0.000    -4.005491   -
3.399348
        full |   .1086104    .090719     1.20   0.232    -.0698947
.2871154
```

```
       _cons |   906.7392   28.26505    32.08   0.000     851.1228
962.3555
------------------------------------------------------------------------
-------
```

Now, let's use the corrected data file and repeat the regression analysis. We see quite a difference in the results! In the original analysis (above), **acs_k3** was nearly significant, but in the corrected analysis (below) the results show this variable to be not significant, perhaps due to the cases where class size was given a negative value. Likewise, the percentage of teachers with full credentials was not significant in the original analysis, but is significant in the corrected analysis, perhaps due to the cases where the value was given as the proportion with full credentials instead of the percent. Also, note that the corrected analysis is based on 398 observations instead of 313 observations, due to getting the complete data for the **meals** variable which had lots of missing values.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2

regress api00 acs_k3 meals full

      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  3,   394) =
615.55
       Model |  6604966.18     3  2201655.39           Prob > F      =
0.0000
    Residual |  1409240.96   394   3576.7537           R-squared     =
0.8242
-------------+------------------------------           Adj R-squared =
0.8228
       Total |  8014207.14   397  20186.9197           Root MSE      =
59.806

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      acs_k3 |  -.7170622   2.238821    -0.32   0.749    -5.118592
3.684468
       meals |  -3.686265   .1117799   -32.98   0.000    -3.906024   -
3.466505
        full |   1.327138  .2388739     5.56   0.000      .857511
1.796765
       _cons |   771.6581   48.86071    15.79   0.000     675.5978
867.7184
------------------------------------------------------------------------
-------
```

From this point forward, we will use the corrected, **elemapi2**, data file. You might want to save this on your computer so you can use it in future analyses.

```
save elemapi2
```

So far we have covered some topics in data checking/verification, but we have not really discussed regression analysis itself. Let's now talk more about performing regression analysis in Stata.

1.3 Simple Linear Regression

Let's begin by showing some examples of simple linear regression using Stata. In this type of regression, we have only one predictor variable. This variable may be continuous, meaning that it may assume all values within a range, for example, age or height, or it may be dichotomous, meaning that the variable may assume only one of two values, for example, 0 or 1. The use of categorical variables with more than two levels will be covered in Chapter 3. There is only one response or dependent variable, and it is continuous.

In Stata, the dependent variable is listed immediately after the **regress** command followed by one or more predictor variables. Let's examine the relationship between the size of school and academic performance to see if the size of the school is related to academic performance. For this example, **api00** is the dependent variable and **enroll** is the predictor.

```
regress api00 enroll
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
44.83
       Model |  817326.293     1  817326.293           Prob > F      =
0.0000
    Residual |  7256345.70   398  18232.0244           R-squared     =
0.1012
-------------+------------------------------           Adj R-squared =
0.0990
       Total |  8073672.00   399  20234.7669           Root MSE      =
135.03

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      enroll |  -.1998674   .0298512    -6.70   0.000    -.2585532   -
.1411817
       _cons |   744.2514   15.93308    46.71   0.000     712.9279
775.5749
------------------------------------------------------------------------
-------
```

Let's review this output a bit more carefully. First, we see that the F-test is statistically significant, which means that the model is statistically significant. The R-squared of .1012 means that approximately 10% of the variance of **api00** is accounted for by the model, in this case, **enroll**. The t-test for **enroll** equals -6.70, and is statistically significant, meaning that the regression coefficient for **enroll** is significantly different from zero. Note that $(-6.70)^2 = 44.89$, which is the same as the F-statistic (with some rounding error). The coefficient for **enroll** is -

.1998674, or approximately -.2, meaning that for a one unit increase in **enroll**, we would expect a .2-unit decrease in **api00**. In other words, a school with 1100 students would be expected to have an api score 20 units lower than a school with 1000 students. The constant is 744.2514, and this is the predicted value when **enroll** equals zero. In most cases, the constant is not very interesting. We have prepared an annotated output which shows the output from this regression along with an explanation of each of the items in it.

In addition to getting the regression table, it can be useful to see a scatterplot of the predicted and outcome variables with the regression line plotted. After you run a regression, you can create a variable that contains the predicted values using the **predict** command. You can get these values at any point after you run a **regress** command, but remember that once you run a new regression, the predicted values will be based on the most recent regression. To create predicted values you just type predict and the name of a new variable Stata will give you the fitted values. For this example, our new variable name will be **fv**, so we will type

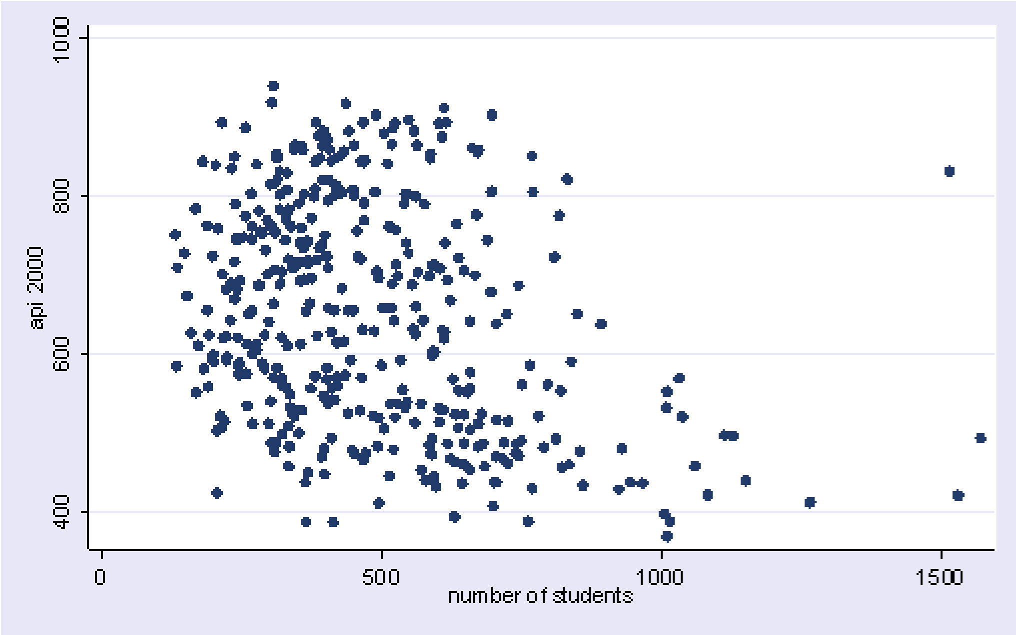
```
predict fv

(option xb assumed; fitted values)
```

If we use the **list** command, we see that a fitted value has been generated for each observation.

```
list api00 fv in 1/10
      api00         fv
  1.    369   542.5851
  2.    386   671.4996
  3.    386   661.7062
  4.    387   541.7857
  5.    387   592.1523
  6.    394   618.5348
  7.    397   543.5845
  8.    406   604.5441
  9.    411   645.5169
 10.    412    491.619
```

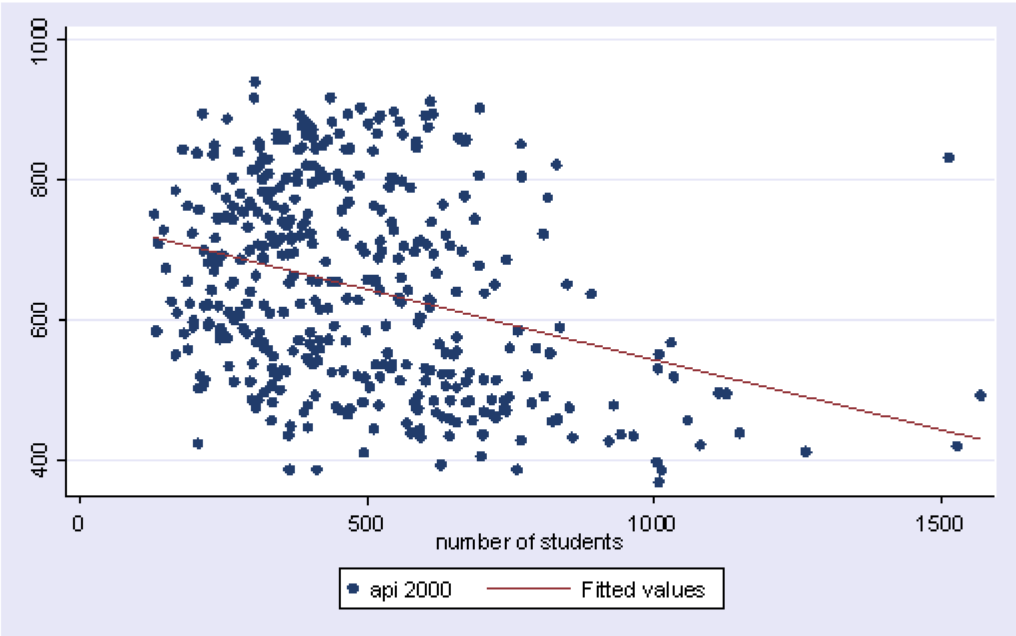
Below we can show a scatterplot of the outcome variable, **api00** and the predictor, **enroll**.

```
scatter api00 enroll
```



We can combine **scatter** with **lfit** to show a scatterplot with fitted values.

```
twoway (scatter api00 enroll) (lfit api00 enroll)
```



As you see, some of the points appear to be outliers. If you use the **mlabel(snum)** option on the **scatter** command, you can see the school number for each point. This allows us to see, for example, that one of the outliers is school 2910.

```
twoway (scatter api00 enroll, mlabel(snum)) (lfit api00 enroll)
```

As we saw earlier, the **predict** command can be used to generate predicted (fitted) values after running **regress**. You can also obtain residuals by using the **predict** command followed by a variable name, in this case **e**, with the **residual** option.

```
predict e, residual
```

This command can be shortened to **predict e, resid** or even **predict e, r**. The table below shows some of the other values can that be created with the **predict** option.

Value to be created	Option after Predict
predicted values of y (y is the dependent variable)	no option needed
residuals	resid
standardized residuals	rstandard
studentized or jackknifed residuals	rstudent
leverage	lev or hat
standard error of the residual	stdr
Cook's D	cooksd
standard error of predicted individual y	stdf
standard error of predicted mean y	stdp

1.4 Multiple Regression

Now, let's look at an example of multiple regression, in which we have one outcome (dependent) variable and multiple predictors. Before we begin with our next example, we need to make a decision regarding the variables that we have created, because we will be creating similar variables with our multiple regression, and we don't want to get the variables confused. For example, in the simple regression we created a variable **fv** for our predicted (fitted) values and **e** for the residuals. If we want to create predicted values for our next example we could call the predicted value something else, e.g., **fv_mr**, but this could start getting confusing. We could drop the variables we have created, using **drop fv e**. Instead, let's clear out the data in memory and **use** the **elemapi2** data file again. When we start new examples in future chapters, we will clear out the existing data file and use the file again to start fresh.

```
clear
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2
```

For this multiple regression example, we will regress the dependent variable, **api00**, on all of the predictor variables in the data set.

```
regress api00 ell meals yr_rnd mobility acs_k3 acs_46 full emer enroll
      Source |       SS       df       MS              Number of obs =
395
-------------+------------------------------           F(  9,   385) =
232.41
       Model |  6740702.01     9   748966.89           Prob > F      =
0.0000
    Residual |  1240707.78   385  3222.61761           R-squared     =
0.8446
-------------+------------------------------           Adj R-squared =
0.8409
       Total |  7981409.79   394  20257.3852           Root MSE      =
56.768

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
         ell |  -.8600707   .2106317    -4.08   0.000    -1.274203   -
.4459382
       meals |  -2.948216   .1703452   -17.31   0.000     -3.28314   -
2.613293
      yr_rnd |  -19.88875   9.258442    -2.15   0.032    -38.09218    -
1.68531
    mobility |  -1.301352   .4362053    -2.98   0.003    -2.158995   -
.4437089
      acs_k3 |     1.3187   2.252683     0.59   0.559      -3.1104
5.747801
      acs_46 |   2.032456   .7983213     2.55   0.011      .462841
3.602071
        full |    .609715   .4758205     1.28   0.201    -.3258169
1.545247
        emer |  -.7066192   .6054086    -1.17   0.244     -1.89694
.4837018
      enroll |   -.012164   .0167921    -0.72   0.469    -.0451798
.0208517
       _cons |   778.8305   61.68663    12.63   0.000     657.5457
900.1154
------------------------------------------------------------------------
-------
```

Let's examine the output from this regression analysis. As with the simple regression, we look to the p-value of the F-test to see if the overall model is significant. With a p-value of zero to four decimal places, the model is statistically significant. The R-squared is 0.8446, meaning that approximately 84% of the variability of **api00** is accounted for by the variables in the model. In this case, the adjusted R-squared indicates that about 84% of the variability of **api00** is accounted for by the model, even after taking into account the number of predictor variables in the model. The coefficients for each of the variables indicates the amount of change one could expect in **api00** given a one-unit change in the value of that variable, given that all other variables in the model are held constant. For example, consider the variable **ell**. We would expect a decrease of 0.86 in the **api00** score for every one unit increase in **ell**, assuming that all other variables in the model are held constant. The interpretation of much of the output from the multiple regression is the same as it was for the simple regression. We have prepared an annotated output that more thoroughly explains the output of this multiple regression analysis.

You may be wondering what a 0.86 change in **ell** really means, and how you might compare the strength of that coefficient to the coefficient for another variable, say **meals**. To address this problem, we can add an option to the **regress** command called **beta**, which will give us the standardized regression coefficients. The beta coefficients are used by some researchers to compare the relative strength of the various predictors within the model. Because the beta coefficients are all measured in standard deviations, instead of the units of the variables, they can be compared to one another. In other words, the beta coefficients are the coefficients that you would obtain if the outcome and predictor variables were all transformed standard scores, also called z-scores, before running the regression.

```
regress api00 ell meals yr_rnd mobility acs_k3 acs_46 full emer enroll,
beta

      Source |       SS       df       MS              Number of obs =
395
-------------+------------------------------           F(  9,   385) =
232.41
       Model |  6740702.01     9   748966.89           Prob > F      =
0.0000
    Residual |  1240707.78   385  3222.61761           R-squared     =
0.8446
-------------+------------------------------           Adj R-squared =
0.8409
       Total |  7981409.79   394  20257.3852           Root MSE      =
56.768

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|
Beta
-------------+----------------------------------------------------------
-------
         ell |  -.8600707   .2106317    -4.08   0.000                  -
.1495771
       meals |  -2.948216   .1703452   -17.31   0.000                  -
.6607003
```

```
      yr_rnd |  -19.88875   9.258442    -2.15   0.032                  -
.0591404
    mobility |  -1.301352   .4362053    -2.98   0.003                  -
.0686382
      acs_k3 |     1.3187   2.252683     0.59   0.559                   
.0127287
      acs_46 |   2.032456   .7983213     2.55   0.011                   
.0549752
        full |    .609715   .4758205     1.28   0.201                   
.0637969
        emer |  -.7066192   .6054086    -1.17   0.244                  -
.0580132
      enroll |   -.012164   .0167921    -0.72   0.469                  -
.0193554
       _cons |   778.8305   61.68663    12.63   0.000                   
.
----------------------------------------------------------------------
-------
```

Because the coefficients in the Beta column are all in the same standardized units you can compare these coefficients to assess the relative strength of each of the predictors. In this example, **meals** has the largest Beta coefficient, -0.66 (in absolute value), and **acs_k3** has the smallest Beta, 0.013. Thus, a one standard deviation increase in **meals** leads to a 0.66 standard deviation decrease in predicted **api00**, with the other variables held constant. And, a one standard deviation increase in **acs_k3**, in turn, leads to a 0.013 standard deviation increase in predicted **api00** with the other variables in the model held constant.

In interpreting this output, remember that the difference between the numbers listed in the Coef. column and the Beta column is in the units of measurement. For example, to describe the raw coefficient for **ell** you would say "A one-unit decrease in **ell** would yield a .86-unit increase in the predicted **api00**." However, for the standardized coefficient (Beta) you would say, "A one standard deviation decrease in **ell** would yield a .15 standard deviation increase in the predicted **api00**."

The **listcoef** command gives more extensive output regarding standardized coefficients. It is not part of Stata, but you can download it over the internet like this.

```
findit listcoef
```

and then follow the instructions (see also [How can I use the findit command to search for programs and get additional help?]() for more information about using **findit**). Now that we have downloaded **listcoef**, we can run it like this.

```
listcoef

regress (N=395): Unstandardized and Standardized Estimates

 Observed SD: 142.32844
 SD of Error: 56.768104
```

```
--------------------------------------------------------------------------
----
   api00 |      b         t      P>|t|    bStdX    bStdY   bStdXY
SDofX
---------+----------------------------------------------------------------
----
     ell |  -0.86007   -4.083   0.000 -21.2891  -0.0060  -0.1496
24.7527
   meals |  -2.94822  -17.307   0.000 -94.0364  -0.0207  -0.6607
31.8960
  yr_rnd | -19.88875   -2.148   0.032  -8.4174  -0.1397  -0.0591
0.4232
mobility |  -1.30135   -2.983   0.003  -9.7692  -0.0091  -0.0686
7.5069
  acs_k3 |   1.31870    0.585   0.559   1.8117   0.0093   0.0127
1.3738
  acs_46 |   2.03246    2.546   0.011   7.8245   0.0143   0.0550
3.8498
    full |   0.60972    1.281   0.201   9.0801   0.0043   0.0638
14.8924
    emer |  -0.70662   -1.167   0.244  -8.2569  -0.0050  -0.0580
11.6851
  enroll |  -0.01216   -0.724   0.469  -2.7548  -0.0001  -0.0194
226.4732
--------------------------------------------------------------------------
----
```

Let us compare the **regress** output with the **listcoef** output. You will notice that the values listed in the Coef., t, and P>|t| values are the same in the two outputs. The values listed in the Beta column of the **regress** output are the same as the values in the bStadXY column of **listcoef**. The bStdX column gives the unit change in Y expected with a one standard deviation change in X. The bStdY column gives the standard deviation change in Y expected with a one unit change in X. The SDofX column gives that standard deviation of each predictor variable in the model.

For example, the bStdX for **ell** is -21.3, meaning that a one standard deviation increase in **ell** would lead to an expected 21.3 unit decrease in **api00**. The bStdY value for **ell** of -0.0060 means that for a one unit, one percent, increase in english language learners, we would expect a 0.006 standard deviation decrease in **api00**. Because the bStdX values are in standard units for the predictor variables, you can use these coefficients to compare the relative strength of the predictors like you would compare Beta coefficients. The difference is BStdX coefficients are interpreted as changes in the units of the outcome variable instead of in standardized units of the outcome variable. For example, the BStdX for **meals** versus **ell** is -94 versus -21, or about 4 times as large, the same ratio as the ratio of the Beta coefficients. We have created an annotated output that more thoroughly explains the output from **listcoef**.

So far, we have concerned ourselves with testing a single variable at a time, for example looking at the coefficient for **ell** and determining if that is significant. We can also test sets of variables, using the **test** command, to see if the set of variables are significant. First, let's start by testing a single variable, **ell**, using the **test** command.

```
test ell==0
```

```
 ( 1)  ell = 0.0

       F(  1,   385) =   16.67
            Prob > F =    0.0001
```

If you compare this output with the output from the last regression you can see that the result of the F-test, 16.67, is the same as the square of the result of the t-test in the regression (-4.083^2 = 16.67). Note that you could get the same results if you typed the following since Stata defaults to comparing the term(s) listed to 0.

```
test ell
 ( 1)  ell = 0.0

       F(  1,   385) =   16.67
            Prob > F =    0.0001
```

Perhaps a more interesting test would be to see if the contribution of class size is significant. Since the information regarding class size is contained in two variables, **acs_k3** and **acs_46**, we include both of these with the **test** command.

```
test acs_k3 acs_46
 ( 1)  acs_k3 = 0.0
 ( 2)  acs_46 = 0.0

       F(  2,   385) =    3.95
            Prob > F =    0.0200
```

The significant F-test, 3.95, means that the collective contribution of these two variables is significant. One way to think of this, is that there is a significant difference between a model with **acs_k3** and **acs_46** as compared to a model without them, i.e., there is a significant difference between the "full" model and the "reduced" models.

Finally, as part of doing a multiple regression analysis you might be interested in seeing the correlations among the variables in the regression model. You can do this with the **correlate** command as shown below.

```
correlate api00 ell meals yr_rnd mobility acs_k3 acs_46 full emer
enroll
(obs=395)

             |    api00      ell    meals   yr_rnd mobility   acs_k3
acs_46
-------------+------------------------------------------------------
------
       api00 |   1.0000
         ell |  -0.7655   1.0000
       meals |  -0.9002   0.7711   1.0000
      yr_rnd |  -0.4831   0.5104   0.4247   1.0000
    mobility |  -0.2106  -0.0149   0.2207   0.0321   1.0000
      acs_k3 |   0.1712  -0.0553  -0.1888   0.0222   0.0397   1.0000
      acs_46 |   0.2340  -0.1743  -0.2137  -0.0419   0.1280   0.2708
1.0000
```

```
        full |   0.5759  -0.4867  -0.5285  -0.4045   0.0235   0.1611
0.1212
        emer |  -0.5902   0.4824   0.5402   0.4401   0.0612  -0.1111  -
0.1283
      enroll |  -0.3221   0.4149   0.2426   0.5920   0.1007   0.1084
0.0281

             |     full     emer   enroll
-------------+---------------------------
        full |   1.0000
        emer |  -0.9059   1.0000
      enroll |  -0.3384   0.3417   1.0000
```

If we look at the correlations with **api00**, we see **meals** and **ell** have the two strongest correlations with **api00**. These correlations are negative, meaning that as the value of one variable goes down, the value of the other variable tends to go up. Knowing that these variables are strongly associated with **api00**, we might predict that they would be statistically significant predictor variables in the regression model.

We can also use the **pwcorr** command to do pairwise correlations. The most important difference between **correlate** and **pwcorr** is the way in which missing data is handled. With **correlate**, an observation or case is dropped if any variable has a missing value, in other words, **correlate** uses listwise , also called casewise, deletion. **pwcorr** uses pairwise deletion, meaning that the observation is dropped only if there is a missing value for the pair of variables being correlated. Two options that you can use with **pwcorr**, but not with **correlate**, are the **sig** option, which will give the significance levels for the correlations and the **obs** option, which will give the number of observations used in the correlation. Such an option is not necessary with **corr** as Stata lists the number of observations at the top of the output.

```
pwcorr api00 ell meals yr_rnd mobility acs_k3 acs_46 full emer enroll,
obs sig

             |    api00      ell    meals   yr_rnd mobility   acs_k3
acs_46
-------------+----------------------------------------------------------
------
       api00 |   1.0000
             |
             |      400
             |
         ell |  -0.7676   1.0000
             |   0.0000
             |      400      400
             |
       meals |  -0.9007   0.7724   1.0000
             |   0.0000   0.0000
             |      400      400      400
             |
      yr_rnd |  -0.4754   0.4979   0.4185   1.0000
             |   0.0000   0.0000   0.0000
             |      400      400      400      400
             |
    mobility |  -0.2064  -0.0205   0.2166   0.0348   1.0000
```

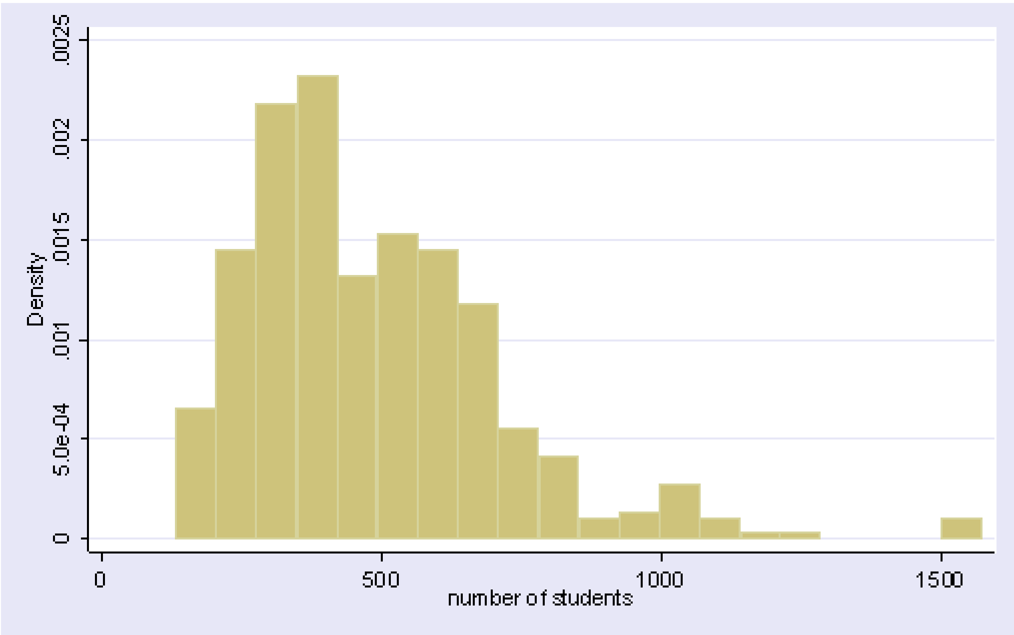
```
             |   0.0000   0.6837   0.0000   0.4883
             |      399      399      399      399      399
             |
      acs_k3 |   0.1710  -0.0557  -0.1880   0.0227   0.0401   1.0000
             |   0.0006   0.2680   0.0002   0.6517   0.4245
             |      398      398      398      398      398      398
             |
      acs_46 |   0.2329  -0.1733  -0.2131  -0.0421   0.1277   0.2708
1.0000
             |   0.0000   0.0005   0.0000   0.4032   0.0110   0.0000
             |      397      397      397      397      396      395
397
             |
        full |   0.5744  -0.4848  -0.5276  -0.3977   0.0252   0.1606
0.1177
             |   0.0000   0.0000   0.0000   0.0000   0.6156   0.0013
0.0190
             |      400      400      400      400      399      398
397
             |
        emer |  -0.5827   0.4722   0.5330   0.4347   0.0596  -0.1103  -
0.1245
             |   0.0000   0.0000   0.0000   0.0000   0.2348   0.0277
0.0131
             |      400      400      400      400      399      398
397
             |
      enroll |  -0.3182   0.4030   0.2410   0.5918   0.1050   0.1089
0.0283
             |   0.0000   0.0000   0.0000   0.0000   0.0360   0.0298
0.5741
             |      400      400      400      400      399      398
397
             |

             |     full     emer   enroll
-------------+---------------------------
        full |   1.0000
             |
             |      400
             |
        emer |  -0.9057   1.0000
             |   0.0000
             |      400      400
             |
      enroll |  -0.3377   0.3431   1.0000
             |   0.0000   0.0000
             |      400      400      400
             |
```

1.5 Transforming Variables

Earlier we focused on screening your data for potential errors. In the next chapter, we will focus on regression diagnostics to verify whether your data meet the assumptions of linear regression. Here, we will focus on the issue of normality. Some researchers believe that linear regression requires that the outcome (dependent) and predictor variables be normally distributed. We need to clarify this issue. In actuality, it is the residuals that need to be normally distributed. In fact, the residuals need to be normal only for the t-tests to be valid. The estimation of the regression coefficients do not require normally distributed residuals. As we are interested in having valid t-tests, we will investigate issues concerning normality.

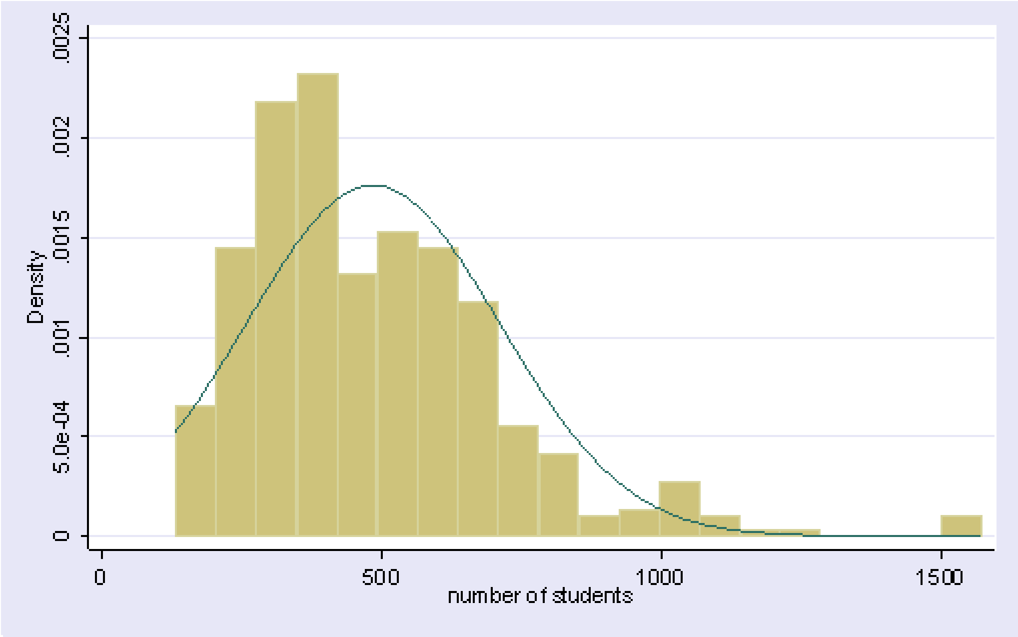
A common cause of non-normally distributed residuals is non-normally distributed outcome and/or predictor variables. So, let us explore the distribution of our variables and how we might transform them to a more normal shape. Let's start by making a histogram of the variable **enroll**, which we looked at earlier in the simple regression.

```
histogram enroll
```



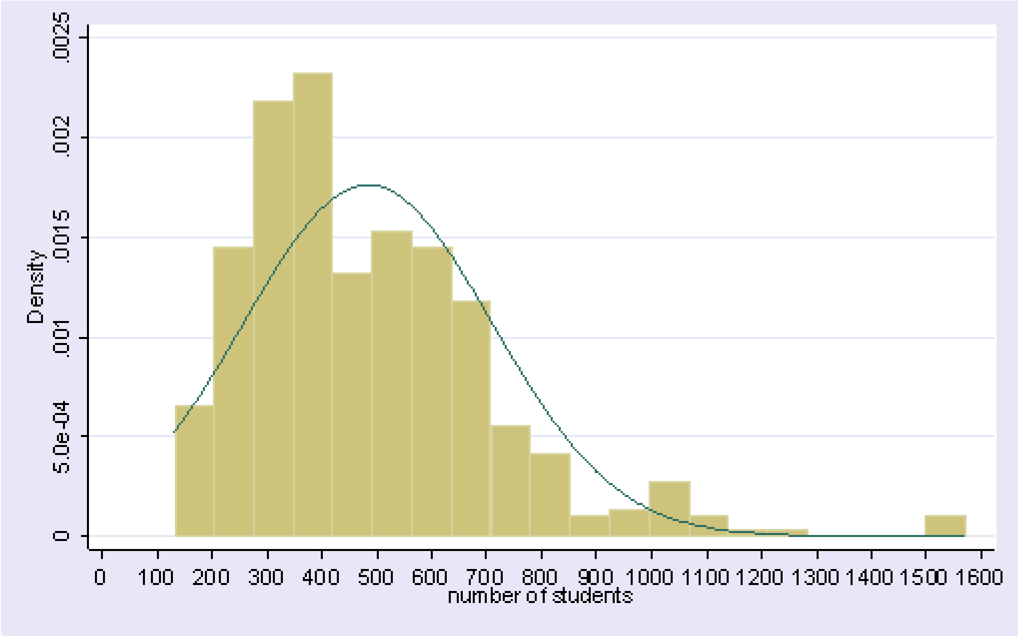
We can use the **normal** option to superimpose a normal curve on this graph and the **bin(20**) option to use 20 bins. The distribution looks skewed to the right.

```
histogram enroll, normal bin(20)
```



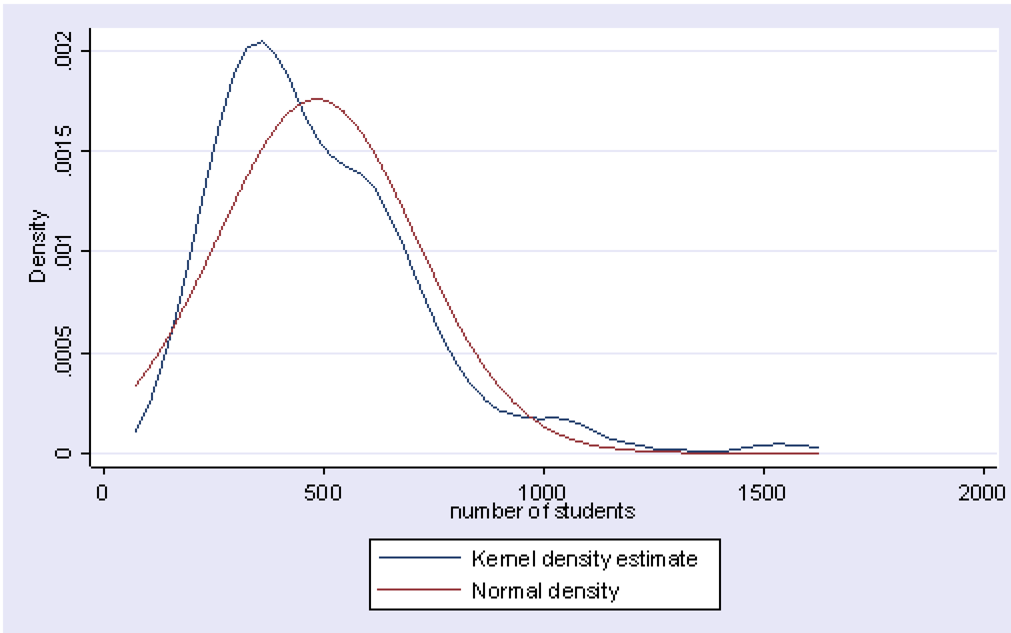
You may also want to modify labels of the axes. For example, we use the **xlabel()** option for labeling the x-axis below, labeling it from 0 to 1600 incrementing by 100.

```
histogram enroll, normal bin(20) xlabel(0(100)1600)
```



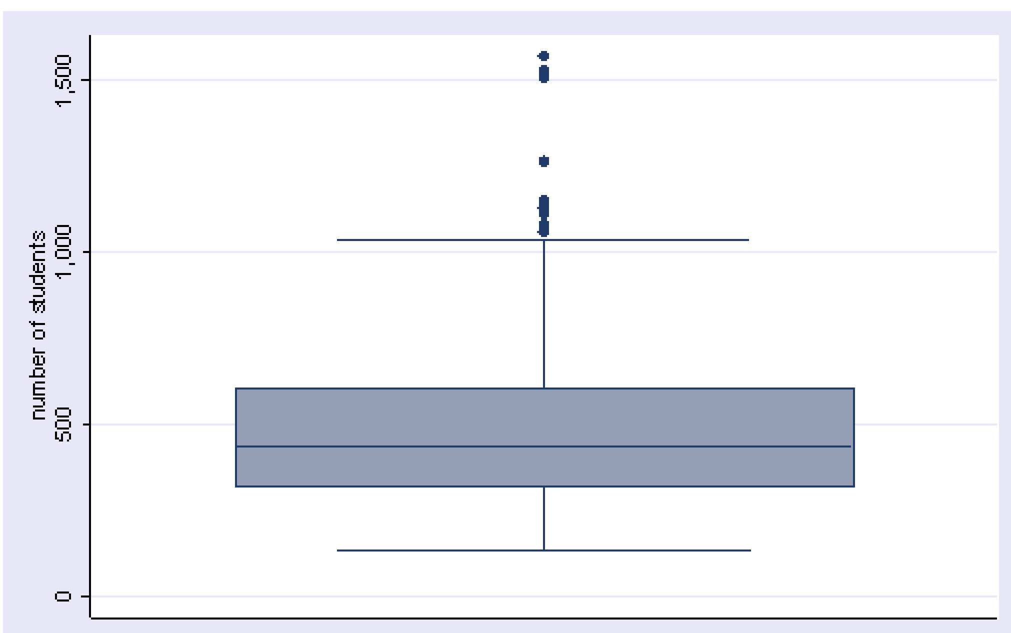
Histograms are sensitive to the number of bins or columns that are used in the display. An alternative to histograms is the kernel density plot, which approximates the probability density of the variable. Kernel density plots have the advantage of being smooth and of being independent of the choice of origin, unlike histograms. Stata implements kernel density plots with the **kdensity** command.

```
kdensity enroll, normal
```



Not surprisingly, the **kdensity** plot also indicates that the variable **enroll** does not look normal. Now let's make a boxplot for **enroll**, using **graph box** command.

```
graph box enroll
```

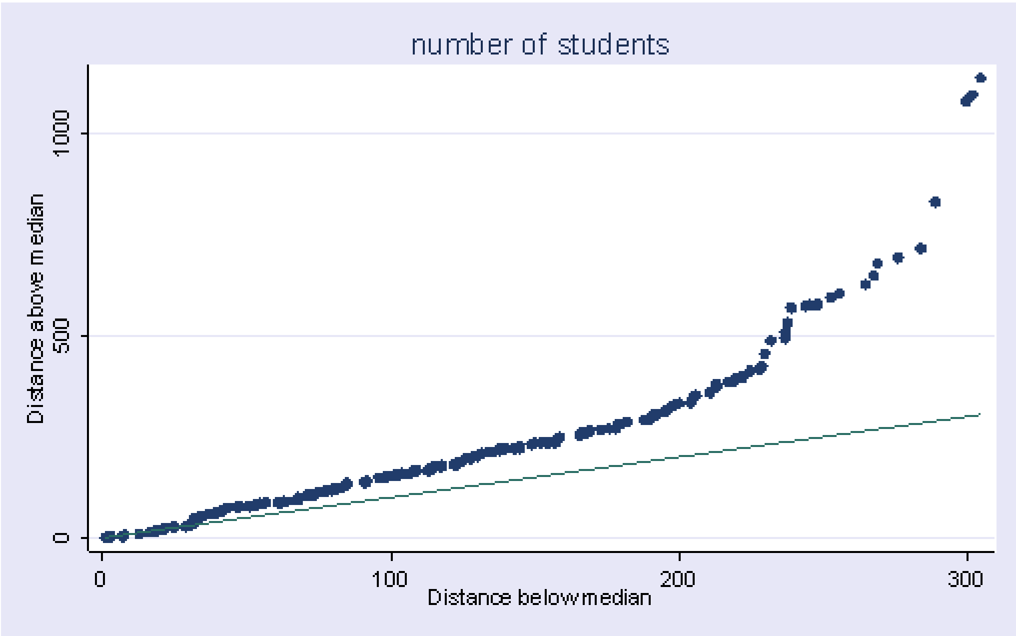


Note the dots at the top of the boxplot which indicate possible outliers, that is, these data points are more than 1.5*(interquartile range) above the 75th percentile. This boxplot also confirms that **enroll** is skewed to the right.

There are three other types of graphs that are often used to examine the distribution of variables; symmetry plots, normal quantile plots and normal probability plots.

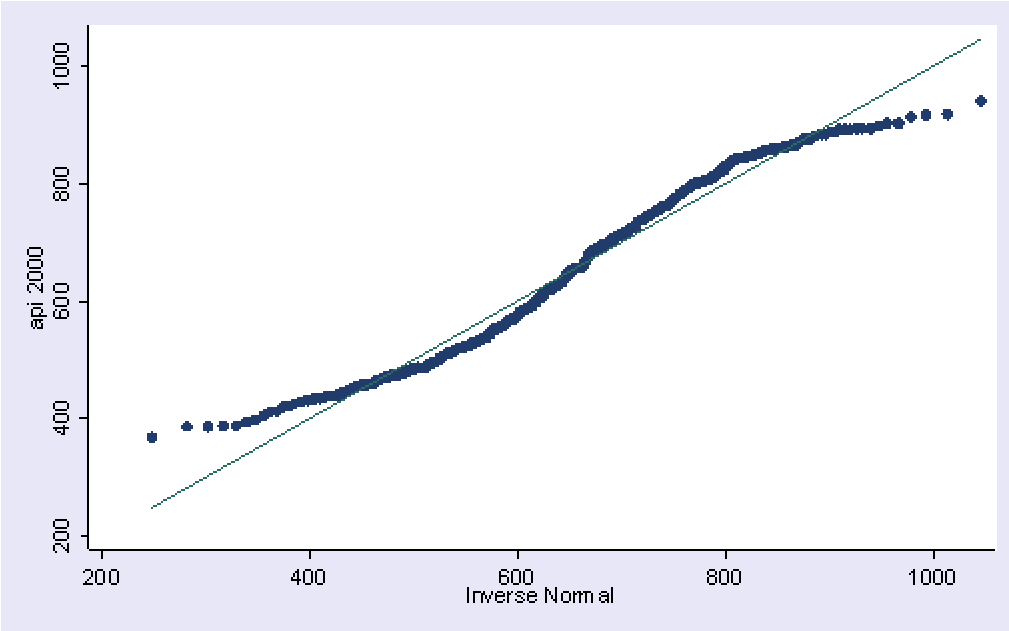
A symmetry plot graphs the distance above the median for the i-th value against the distance below the median for the i-th value. A variable that is symmetric would have points that lie on the diagonal line. As we would expect, this distribution is not symmetric.

```
symplot enroll
```



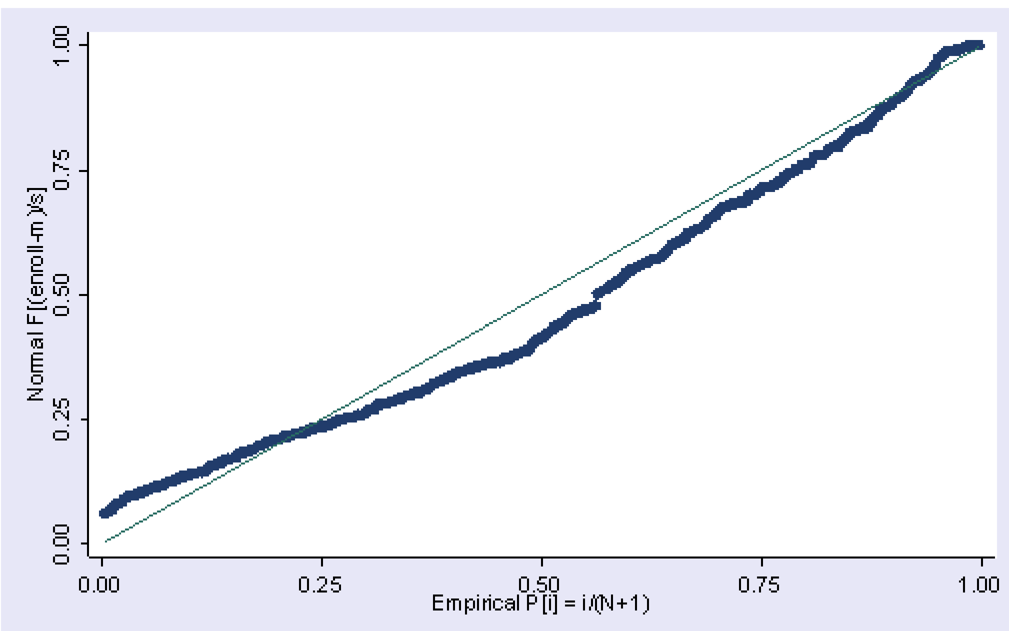
A normal quantile plot graphs the quantiles of a variable against the quantiles of a normal (Gaussian) distribution. **qnorm** is sensitive to non-normality near the tails, and indeed we see considerable deviations from normal, the diagonal line, in the tails. This plot is typical of variables that are strongly skewed to the right.

```
qnorm api00
```



Finally, the normal probability plot is also useful for examining the distribution of variables. **pnorm** is sensitive to deviations from normality nearer to the center of the distribution. Again, we see indications of non-normality in **enroll**.

```
pnorm enroll
```



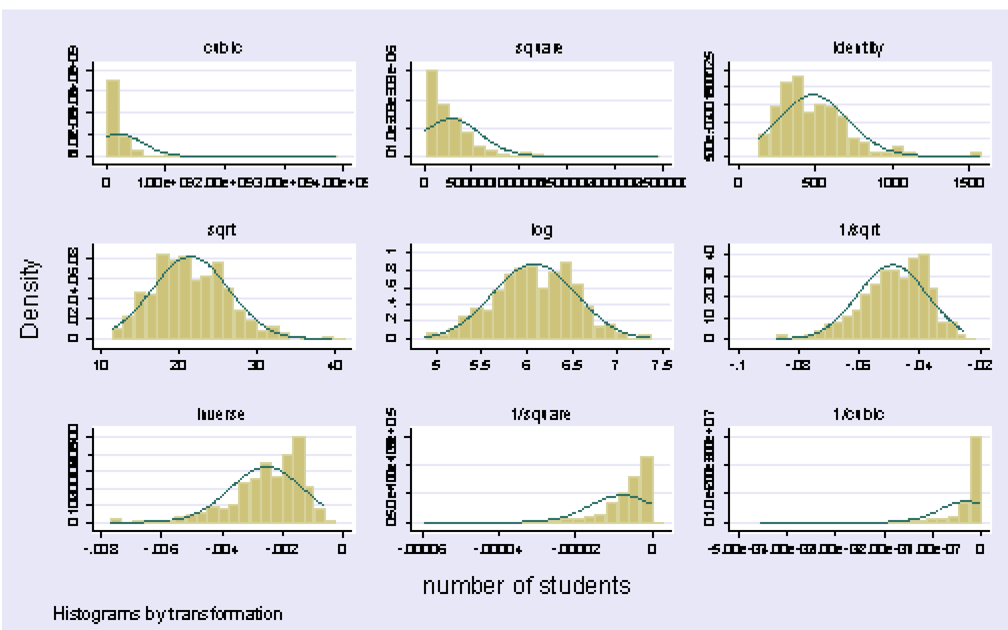
Having concluded that **enroll** is not normally distributed, how should we address this problem? First, we may try entering the variable as-is into the regression, but if we see problems, which we likely would, then we may try to transform **enroll** to make it more normally distributed. Potential transformations include taking the log, the square root or raising the variable to a power. Selecting the appropriate transformation is somewhat of an art. Stata includes the **ladder** and **gladder** commands to help in the process. **Ladder** reports numeric results and **gladder** produces a graphic display. Let's start with **ladder** and look for the transformation with the smallest chi-square.

```
ladder enroll
```

```
ladder enroll

Transformation         formula                chi2(2)       P(chi2)
-------------------------------------------------------------------
cube                   enroll^3                     .         0.000
square                 enroll^2                     .         0.000
raw                    enroll                       .         0.000
square-root            sqrt(enroll)             20.56         0.000
log                    log(enroll)               0.71         0.701
reciprocal root        1/sqrt(enroll)           23.33         0.000
reciprocal             1/enroll                 73.47         0.000
reciprocal square      1/(enroll^2)                 .         0.000
reciprocal cube        1/(enroll^3)                 .         0.000
```

The log transform has the smallest chi-square. Let's verify these results graphically using **gladder**.

```
gladder enroll
```

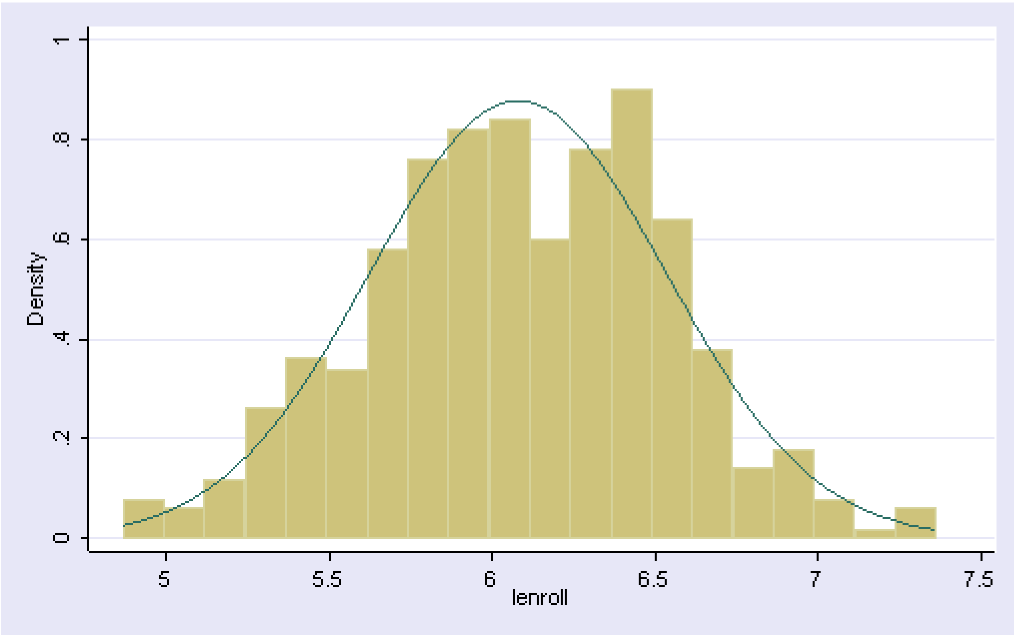


This also indicates that the log transformation would help to make **enroll** more normally distributed. Let's use the **generate** command with the **log** function to create the variable **lenroll** which will be the log of enroll. Note that **log** in Stata will give you the natural log, not log base 10. To get log base 10, type **log10(var)**.

```
generate lenroll = log(enroll)
```

Now let's graph our new variable and see if we have normalized it.

```
hist lenroll, normal
```



We can see that **lenroll** looks quite normal. We would then use the **symplot**, **qnorm** and **pnorm** commands to help us assess whether **lenroll** seems normal, as well as seeing how **lenroll** impacts the residuals, which is really the important consideration.

1.6 Summary

In this lecture we have discussed the basics of how to perform simple and multiple regressions, the basics of interpreting output, as well as some related commands. We examined some tools and techniques for screening for bad data and the consequences such data can have on your results. Finally, we touched on the assumptions of linear regression and illustrated how you can check the normality of your variables and how you can transform your variables to achieve normality. The next chapter will pick up where this chapter has left off, going into a more thorough discussion of the assumptions of linear regression and how you can use Stata to assess these assumptions for your data. In particular, the next lecture will address the following issues.

- Checking for points that exert undue influence on the coefficients
- Checking for constant error variance (homoscedasticity)
- Checking for linear relationships
- Checking model specification

- Checking for multicollinearity
- Checking normality of residuals

See the Stata Topics: Regression page for more information and resources on simple and multiple regression in Stata.

1.7 Self Assessment

1. Make five graphs of **api99**: histogram, kdensity plot, boxplot, symmetry plot and normal quantile plot.
2. What is the correlation between **api99** and **meals**?
3. Regress **api99** on **meals**. What does the output tell you?
4. Create and list the fitted (predicted) values.
5. Graph **meals** and **api99** with and without the regression line.
6. Look at the correlations among the variables **api99 meals ell avg_ed** using the **corr** and **pwcorr** commands. Explain how these commands are different. Make a scatterplot matrix for these variables and relate the correlation results to the scatterplot matrix.
7. Perform a regression predicting **api99** from **meals** and **ell**. Interpret the output.

Click here for our answers to these self assessment questions.

1.8 For More Information

- Stata Manuals
  - **[R] regress**
  - **[R] predict**
  - **[R] test**
- Related Web Pages
  - Stata FAQ- How can I do a scatterplot with regression line in Stata?
  - Annotated Stata Output- Regression
- Stata Add On Programs
  - http://www.ats.ucla.edu/stat/stata/ado

Chapter 1 - Self Assessment

1. Make five graphs of **api99**: histogram, kdensity plot, boxplot, symmetry plot and normal quantile plot.
2. What is the correlation between **api99** and **meals**?
3. Regress **api99** on **meals**. What does the output tell you?
4. Create and list the fitted (predicted) values.
5. Graph **meals** and **api99** with and without the regression line.
6. Look at the correlations among the variables **api99 meals ell avg_ed** using the **corr** and **pwcorr** commands. Explain how these commands are different. Make a scatterplot matrix for these variables and relate the correlation results to the scatterplot matrix.
7. Perform a regression predicting **api99** from **meals** and **ell**. Interpret the output.

Chapter 1
Self Assessment Answers

Question 1.
Make five graphs of api99: histogram, kdensity plot, boxplot, symmetry plot and normal quantile plot.

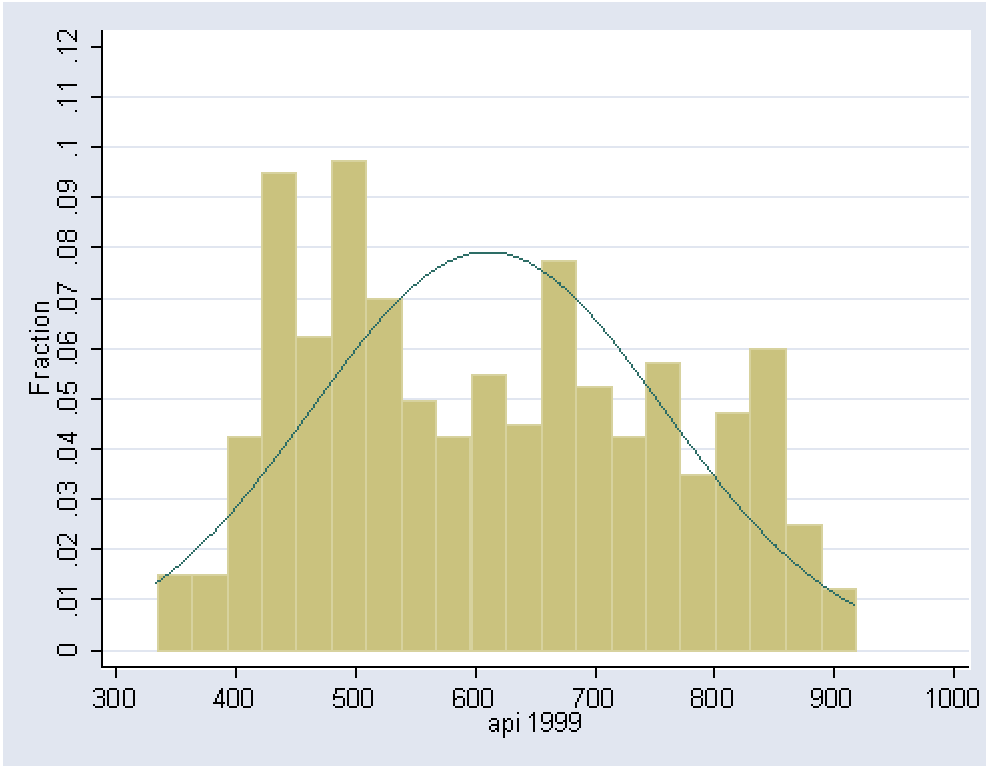
Answer 1.
First we use the **elemapi2** data file.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
```

Below we make the plots mentioned in question 1.

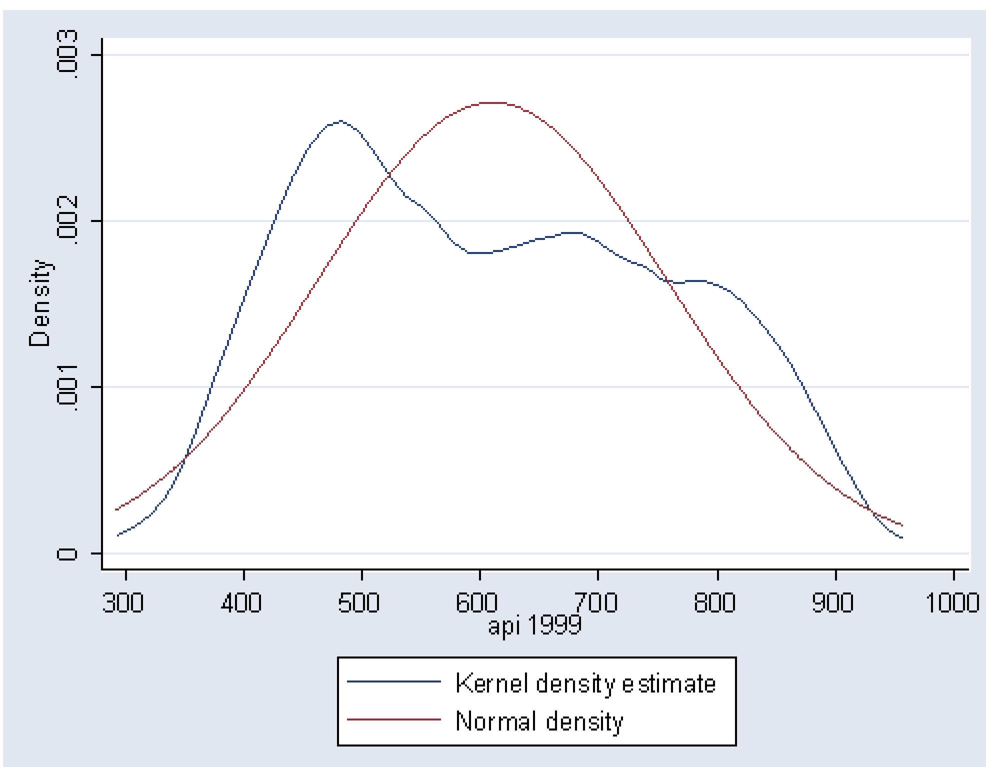
Histogram

```
histogram api99, bin(20) fraction normal xlabel(300(100)1000)
ylabel(0(.01).12)
```



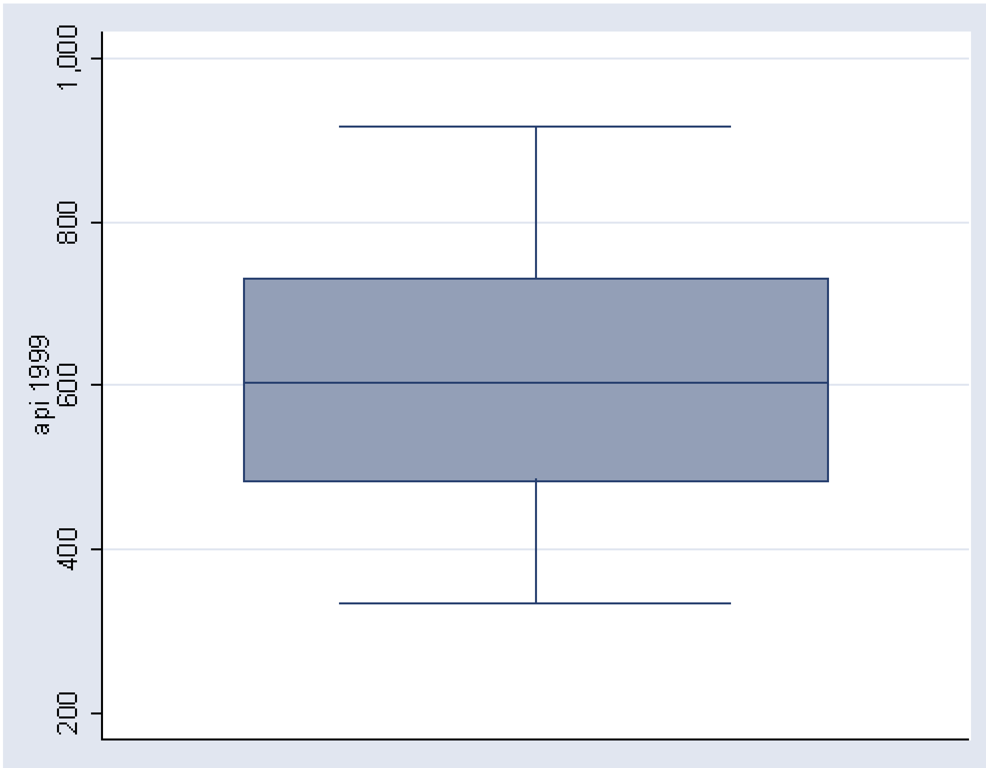
kdensity plot

```
kdensity api99, normal xlabel(300(100)1000)
```



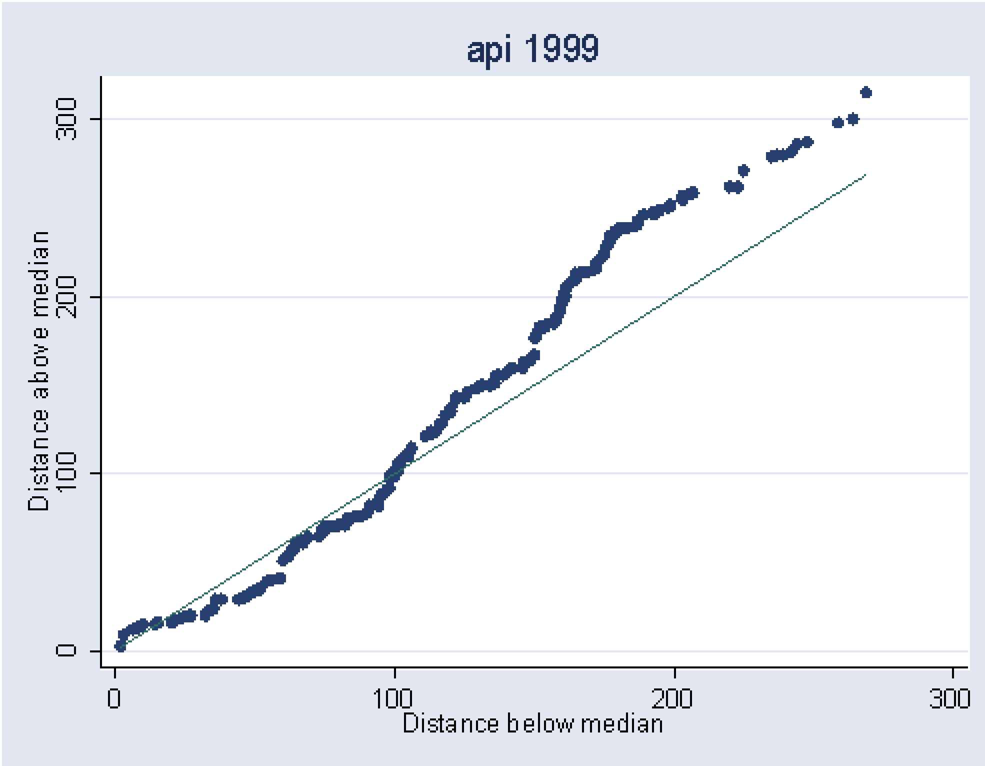
boxplot

```
graph box api99
```



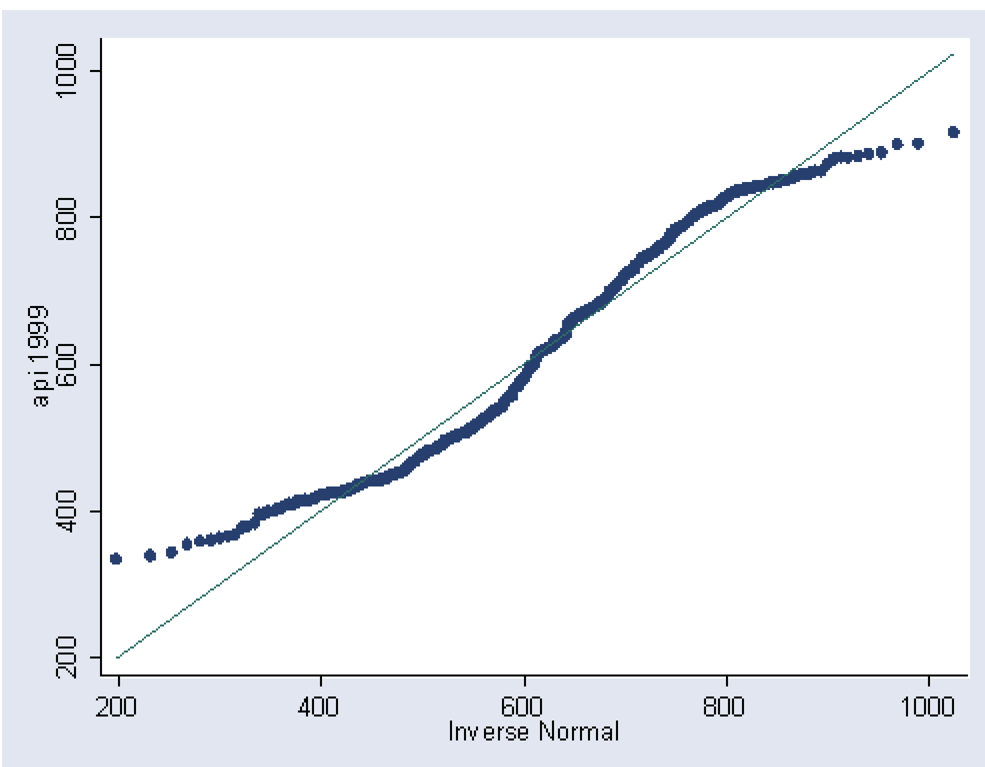
symmetry plot

```
symplot api99
```



normal quantile plot

qnorm api99



Question 2.
What is the correlation between api99 and meals?

Answer 2.
Below we use the **corr** command to get this correlation.

```
corr api99 meals
(obs=400)

             |    api99    meals
-------------+------------------
       api99 |   1.0000
       meals |  -0.9081   1.0000
```

Question 3.
Regress **api99** on **meals**. What does the output tell you?

Answer 3.
Below we perform the regression predicting **api99** from **meals**.

```
regress api99 meals
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
1872.39
       Model |  7123743.65     1  7123743.65           Prob > F      =
0.0000
    Residual |  1514239.28   398  3804.62132           R-squared     =
0.8247
-------------+------------------------------           Adj R-squared =
0.8243
       Total |  8637982.94   399    21649.08           Root MSE      =
61.682

------------------------------------------------------------------------
-------
       api99 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
       meals |  -4.187142   .0967652   -43.27   0.000    -4.377377   -
3.996908
       _cons |     862.76    6.60114   130.70   0.000     849.7825
875.7375
------------------------------------------------------------------------
-------
```

We see that the coefficient for **meals** has a t value of -43 and that it is significant. The coefficient is -4.18 (let's round it to -4.2) so very every unit increase in meals, api99 goes down by 4.22 points. In other words, for every percent increase in children who receive free meals in a school, the api score for that school would be predicted to decrease by 4.2 points.

Question 4.
Create and list the fitted (predicted) values.

Answer 4.
We can create the predicted values using the predict command, as shown below.

```
predict yhat
(option xb assumed; fitted values)
```

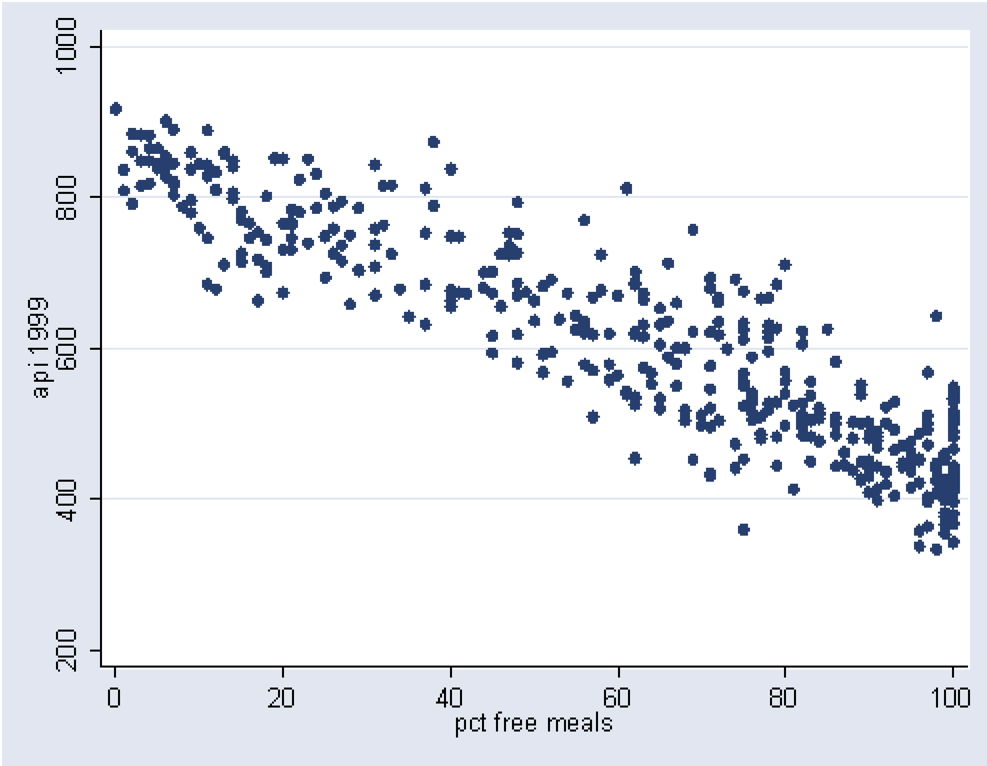
We can view the first 20 predicted and actual values for api99 like this.

```
list api99 yhat in 1/20
      api99       yhat
  1.    600   582.2214
  2.    501   477.5429
  3.    472   456.6072
  4.    487   485.9172
  5.    425   490.1043
  6.    844   820.8885
  7.    864   841.8243
  8.    791   854.3857
  9.    838   841.8243
 10.    703   741.3329
 11.    808   858.5729
 12.    496   565.4729
 13.    815   850.1985
 14.    711   808.3271
 15.    802     833.45
 16.    780   770.6429
 17.    816     833.45
 18.    677   695.2743
 19.    759   820.8885
 20.    632   707.8358
```

Question 5.
Graph meals and api99 with and without the regression line.

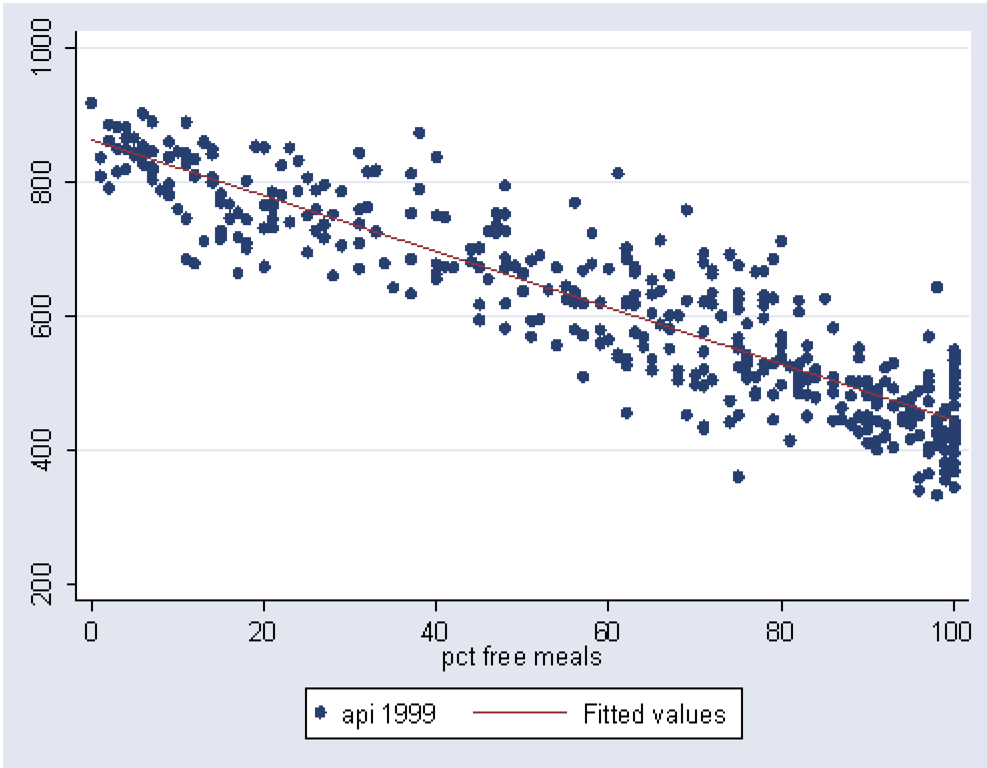
Answer 5.
We can graph **api99** by **meals** like this.

```
graph twoway scatter api99 meals
```



We can show a graph of **api99** by **meals** with a regression line using the **scatter** program (assuming you installed it as shown in chapter 1) like this.

```
graph twoway (scatter api99 meals) (lfit api99 meals)
```



Question 6.
Look at the correlations among the variables api99 meals ell avg_ed using the **corr** and **pwcorr** commands. Explain how these commands are different. Make a scatterplot matrix for these variables and relate the correlation results to the scatterplot matrix.

We first show the output using the **corr** command.

```
corr api99 meals ell avg_ed
(obs=381)

             |    api99    meals      ell   avg_ed
-------------+------------------------------------
       api99 |   1.0000
       meals |  -0.9088   1.0000
         ell |  -0.7638   0.7772   1.0000
      avg_ed |   0.7953  -0.8136  -0.6930   1.0000
```

Now we use the **pwcorr** command.

```
pwcorr api99 meals ell avg_ed
             |    api99    meals      ell   avg_ed
-------------+------------------------------------
       api99 |   1.0000
       meals |  -0.9081   1.0000
         ell |  -0.7628   0.7724   1.0000
      avg_ed |   0.7953  -0.8136  -0.6930   1.0000
```

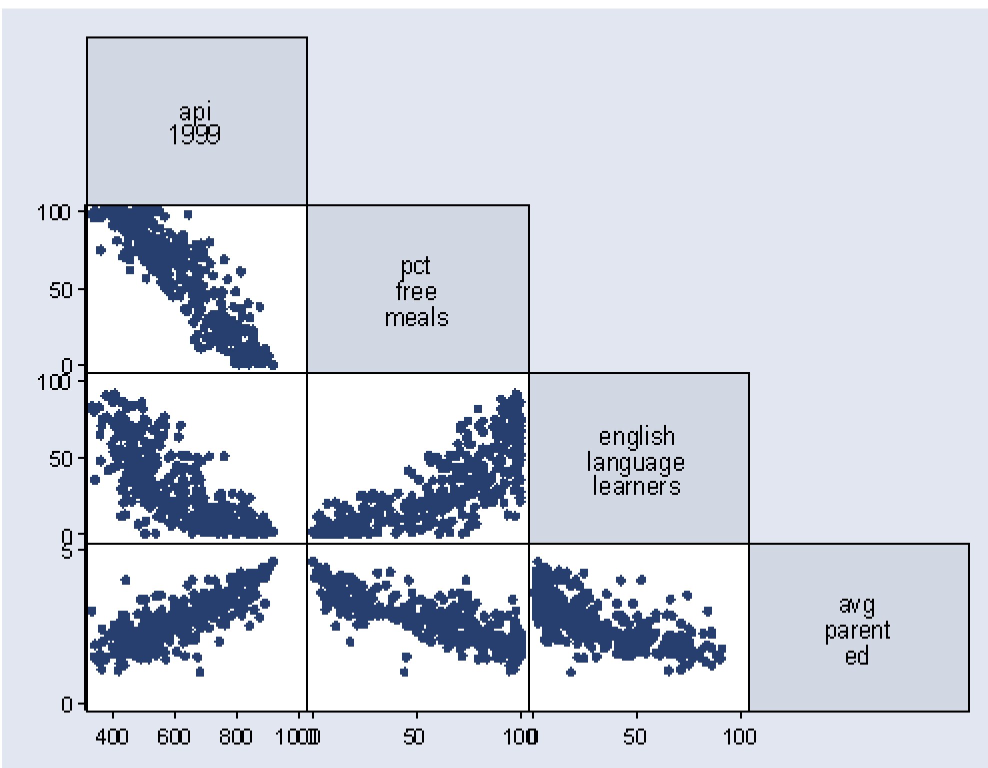
It is hard to see the differences unless we use the **obs** option.

```
pwcorr api99 meals ell avg_ed, obs
             |    api99    meals      ell   avg_ed
-------------+------------------------------------
       api99 |   1.0000 
             |      400
             |
       meals |  -0.9081   1.0000 
             |      400      400
             |
         ell |  -0.7628   0.7724   1.0000 
             |      400      400      400
             |
      avg_ed |   0.7953  -0.8136  -0.6930   1.0000 
             |      381      381      381      381
             |
```

The **corr** command performs listwise deletion, so all of the correlations are based on the listwise n of 381. The **pwcorr** performs pairwise deletion and shows the correlation based on the number valid observations for each pair, for example **api99** and **meals** have 400 valid pairs, but **api99** and **avg_ed** have 381 valid pairs.

Below we show the scatterplot for **api99 meals ell avg_ed**.

```
graph matrix api99 meals ell avg_ed, half
```



The scatterplot matrix is a visual representation of the correlation between the variables. For each scatterplot in the scatterplot matrix, you can see the corresponding correlation in the correlation matrix.

Question 7.
Perform a regression predicting **api99** from **meals ell avg_ed**. Interpret the output.

Answer 7.
We can run this regression as shown below.

```
regress api99 meals ell
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
997.57
       Model |  7204423.31     2  3602211.66           Prob > F      =
0.0000
    Residual |  1433559.63   397  3610.98143           R-squared     =
0.8340
-------------+------------------------------           Adj R-squared =
0.8332
       Total |  8637982.94   399    21649.08           Root MSE      =
60.091

------------------------------------------------------------------------
-------
       api99 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
       meals |   -3.64528   .1484193   -24.56   0.000    -3.937066   -
3.353494
         ell |  -.9013059    .190679    -4.73   0.000    -1.276173   -
.5264392
       _cons |   858.4259   6.495999   132.15   0.000      845.655
871.1967
------------------------------------------------------------------------
-------
```

The t- value for all of these predictors are significant, so each is useful in predicting **api99**. The coefficient for **meals** is -3.6 and indicates that for every additional percent of children who receive free meals, the api score is predicted to be 3.6 points lower. The coefficient for **ell** is -.9, indicating that for every percentage increase in non-English speaking students, the api score for the school is predicted to be .9 units less.

Chapter 2 - Regression Diagnostics

Chapter Outline

2.0 Regression Diagnostics

In the previous chapter, we learned how to do ordinary linear regression with Stata, concluding with methods for examining the distribution of our variables. Without verifying that your data have met the assumptions underlying OLS regression, your results may be misleading. This chapter will explore how you can use Stata to check on how well your data meet the assumptions of OLS regression. In particular, we will consider the following assumptions.

- Linearity - the relationships between the predictors and the outcome variable should be linear
- Normality - the errors should be normally distributed - technically normality is necessary only for hypothesis tests to be valid,
  estimation of the coefficients only requires that the errors be identically and independently distributed
- Homogeneity of variance (homoscedasticity) - the error variance should be constant
- Independence - the errors associated with one observation are not correlated with the errors of any other observation
- Errors in variables - predictor variables are measured without error (we will cover this in Chapter 4)
- Model specification - the model should be properly specified (including all relevant variables, and excluding irrelevant variables)

Additionally, there are issues that can arise during the analysis that, while strictly speaking are not assumptions of regression, are none the less, of great concern to data analysts.

- Influence - individual observations that exert undue influence on the coefficients
- Collinearity - predictors that are highly collinear, i.e., linearly related, can cause problems in estimating the regression coefficients.

Many graphical methods and numerical tests have been developed over the years for regression diagnostics. Stata has many of these methods built-in, and others are available that can be downloaded over the internet. In particular, Nicholas J. Cox (University of Durham) has produced a collection of convenience commands which can be downloaded from SSC (**ssc**

install *commandname*). These commands include **indexplot**, **rvfplot2**, **rdplot**, **qfrplot** and **ovfplot**. In this chapter, we will explore these methods and show how to verify regression assumptions and detect potential problems using Stata.

2.1 Unusual and influential data

A single observation that is substantially different from all other observations can make a large difference in the results of your regression analysis. If a single observation (or small group of observations) substantially changes your results, you would want to know about this and investigate further. There are three ways that an observation can be unusual.

Outliers: In linear regression, an outlier is an observation with large residual. In other words, it is an observation whose dependent-variable value is unusual given its values on the predictor variables. An outlier may indicate a sample peculiarity or may indicate a data entry error or other problem.

Leverage: An observation with an extreme value on a predictor variable is called a point with high leverage. Leverage is a measure of how far an observation deviates from the mean of that variable. These leverage points can have an effect on the estimate of regression coefficients.

Influence: An observation is said to be influential if removing the observation substantially changes the estimate of coefficients. Influence can be thought of as the product of leverage and outlierness.

How can we identify these three types of observations? Let's look at an example dataset called **crime**. This dataset appears in *Statistical Methods for Social Sciences, Third Edition* by Alan Agresti and Barbara Finlay (Prentice Hall, 1997). The variables are state id (**sid**), state name (**state**), violent crimes per 100,000 people (**crime**), murders per 1,000,000 (**murder**), the percent of the population living in metropolitan areas (**pctmetro**), the percent of the population that is white (**pctwhite**), percent of population with a high school education or above (**pcths**), percent of population living under poverty line (**poverty**), and percent of population that are single parents (**single**).

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/crime
  (crime data from agresti & finlay - 1997)

describe

Contains data from crime.dta
  obs:            51                          crime data from agresti &
                                                finlay - 1997
 vars:            11                          6 Feb 2001 13:52
 size:         2,295 (98.9% of memory free)
-----------------------------------------------------------------------
--------
   1. sid       float  %9.0g
   2. state     str3   %9s
   3. crime     int    %8.0g                  violent crime rate
   4. murder    float  %9.0g                  murder rate
```

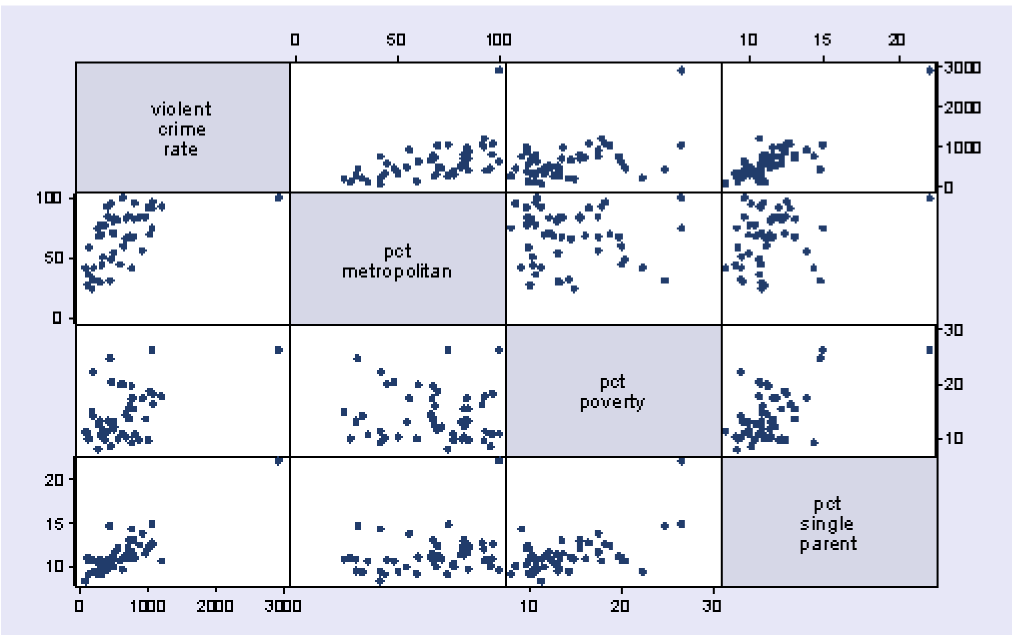
```
   5. pctmetro  float  %9.0g                  pct metropolitan
   6. pctwhite  float  %9.0g                  pct white
   7. pcths     float  %9.0g                  pct hs graduates
   8. poverty   float  %9.0g                  pct poverty
   9. single    float  %9.0g                  pct single parent
----------------------------------------------------------------------
--------
Sorted by:
```

summarize crime murder pctmetro pctwhite pcths poverty single

```
Variable |     Obs        Mean    Std. Dev.       Min        Max
---------+------------------------------------------------------------
   crime |      51    612.8431    441.1003         82       2922
  murder |      51    8.727451    10.71758        1.6       78.5
pctmetro |      51     67.3902    21.95713         24        100
pctwhite |      51    84.11569    13.25839       31.8       98.5
   pcths |      51    76.22353    5.592087       64.3       86.6
 poverty |      51    14.25882    4.584242          8       26.4
  single |      51    11.32549    2.121494        8.4       22.1
```

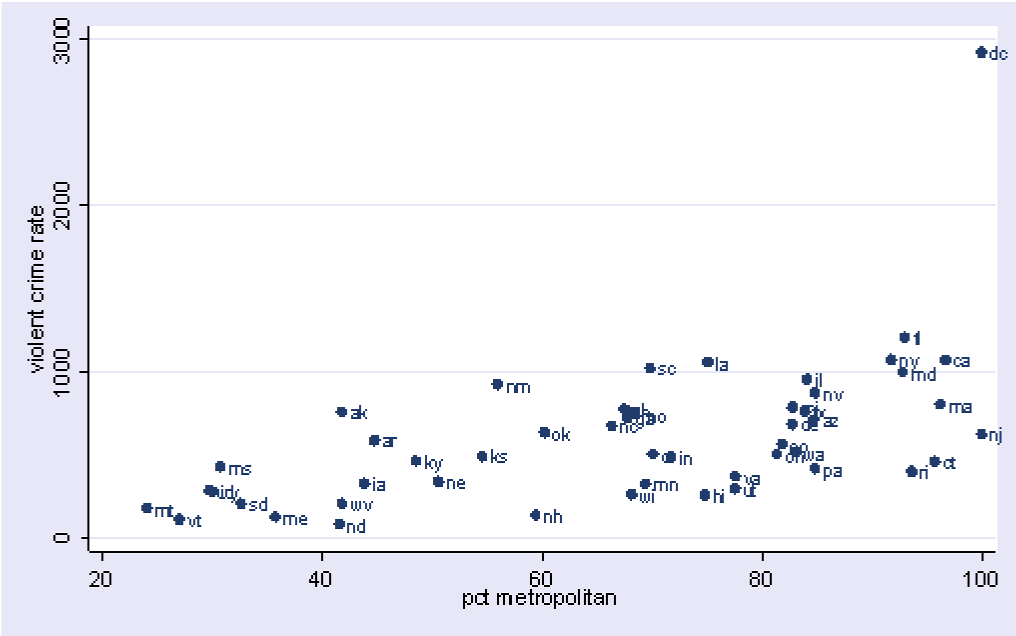
Let's say that we want to predict **crime** by **pctmetro**, **poverty**, and **single**. That is to say, we want to build a linear regression model between the response variable **crime** and the independent variables **pctmetro**, **poverty** and **single**. We will first look at the scatter plots of crime against each of the predictor variables before the regression analysis so we will have some ideas about potential problems. We can create a scatterplot matrix of these variables as shown below.

graph matrix crime pctmetro poverty single

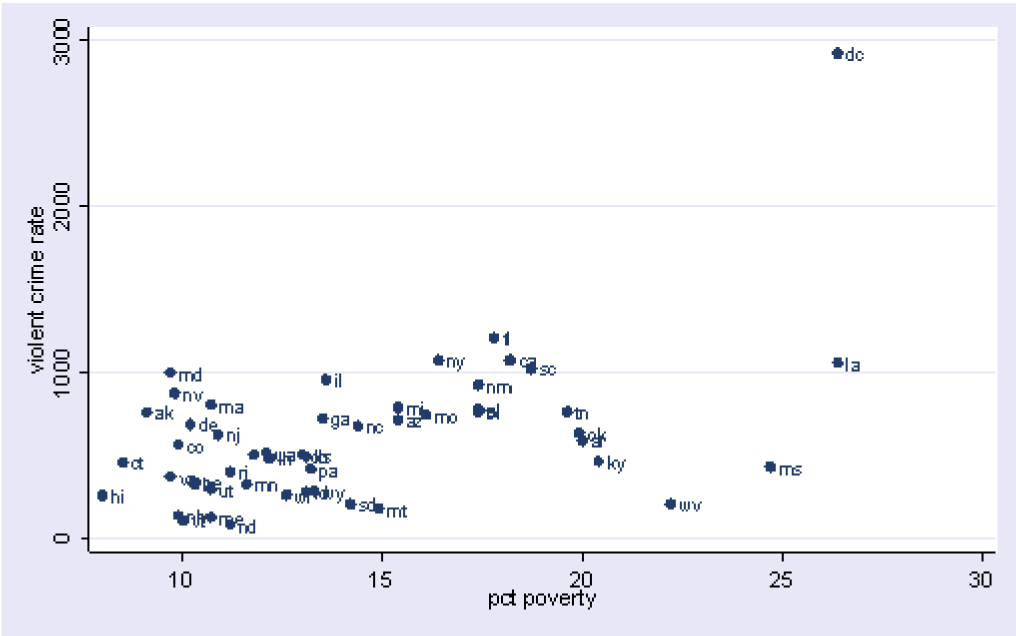


The graphs of **crime** with other variables show some potential problems. In every plot, we see a data point that is far away from the rest of the data points. Let's make individual graphs of **crime** with **pctmetro** and **poverty** and **single** so we can get a better view of these scatterplots. We will add the **mlabel(state)** option to label each marker with the state name to identify outlying states.

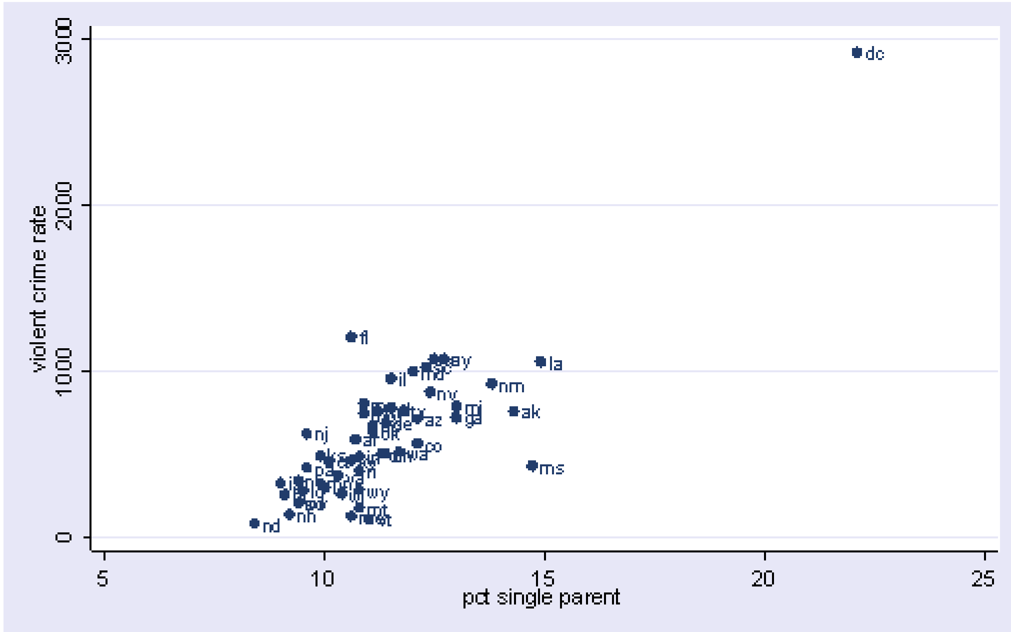
```
scatter crime pctmetro, mlabel(state)
```




```
scatter crime poverty, mlabel(state)
```




```
scatter crime single, mlabel(state)
```



All the scatter plots suggest that the observation for **state** = dc is a point that requires extra attention since it stands out away from all of the other points. We will keep it in mind when we do our regression analysis.

Now let's try the regression command predicting **crime** from **pctmetro poverty** and **single**. We will go step-by-step to identify all the potentially unusual or influential points afterwards.

```
regress crime pctmetro poverty single

  Source |       SS       df       MS                  Number of obs =
51
---------+------------------------------               F(  3,    47) =
82.16
   Model |  8170480.21     3  2723493.40               Prob > F      =
0.0000
Residual |  1557994.53    47  33148.8199               R-squared     =
0.8399
---------+------------------------------               Adj R-squared =
0.8296
   Total |  9728474.75    50  194569.495               Root MSE      =
182.07

-----------------------------------------------------------------------
-------
   crime |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
pctmetro |   7.828935   1.254699      6.240   0.000       5.304806
10.35306
 poverty |   17.68024    6.94093      2.547   0.014       3.716893
31.64359
```

```
  single |   132.4081   15.50322      8.541   0.000       101.2196
163.5965
   _cons |  -1666.436    147.852    -11.271   0.000      -1963.876   -
1368.996
----------------------------------------------------------------------
-------
```

Let's examine the studentized residuals as a first means for identifying outliers. Below we use the **predict** command with the **rstudent** option to generate studentized residuals and we name the residuals **r**. We can choose any name we like as long as it is a legal Stata variable name. Studentized residuals are a type of standardized residual that can be used to identify outliers.

```
predict r, rstudent
```

Let's examine the residuals with a stem and leaf plot. We see three residuals that stick out, -3.57, 2.62 and 3.77.

```
stem r

Stem-and-leaf plot for r (Studentized residuals)

r rounded to nearest multiple of .01
plot in units of .01

-3** | 57
-3** |
-2** |
-2** |
-1** | 84,69
-1** | 30,15,13,04,02
-0** | 87,85,65,58,56,55,54
-0** | 47,46,45,38,36,30,28,21,08,02
 0** | 05,06,08,13,27,28,29,31,35,41,48,49
 0** | 56,64,70,80,82
 1** | 01,03,03,08,15,29
 1** | 59
 2** |
 2** | 62
 3** |
 3** | 77
```

The stem and leaf display helps us see some potential outliers, but we cannot see which **state** (which observations) are potential outliers. Let's sort the data on the residuals and show the 10 largest and 10 smallest residuals along with the state id and state name. Note that in the second **list** command the **-10/l** the last value is the letter "l", NOT the number one.

```
sort r
list sid state r in 1/10

           sid      state          r
  1.        25         ms  -3.570789
  2.        18         la  -1.838577
  3.        39         ri  -1.685598
```

```
     4.         47          wa  -1.303919
     5.         35          oh   -1.14833
     6.         48          wi   -1.12934
     7.          6          co  -1.044952
     8.         22          mi  -1.022727
     9.          4          az  -.8699151
    10.         44          ut  -.8520518
```

```
list sid state r in -10/l

               sid      state           r
    42.         24          mo   .8211724
    43.         20          md    1.01299
    44.         29          ne   1.028869
    45.         40          sc   1.030343
    46.         16          ks   1.076718
    47.         14          il   1.151702
    48.         13          id   1.293477
    49.         12          ia   1.589644
    50.          9          fl   2.619523
    51.         51          dc   3.765847
```

We should pay attention to studentized residuals that exceed +2 or -2, and get even more concerned about residuals that exceed +2.5 or -2.5 and even yet more concerned about residuals that exceed +3 or -3. These results show that DC and MS are the most worrisome observations followed by FL.

Another way to get this kind of output is with a command called **hilo**. You can download **hilo** from within Stata by typing **findit hilo** (see How can I used the findit command to search for programs and get additional help? for more information about using **findit**).

Once installed, you can type the following and get output similar to that above by typing just one command.

```
hilo r state
10 smallest and largest observations on r

        r      state
-3.570789         ms
-1.838577         la
-1.685598         ri
-1.303919         wa
 -1.14833         oh
 -1.12934         wi
-1.044952         co
-1.022727         mi
-.8699151         az
-.8520518         ut

        r      state
8211724          mo
  1.01299         md
 1.028869         ne
 1.030343         sc
```

```
 1.076718         ks
 1.151702         il
 1.293477         id
 1.589644         ia
 2.619523         fl
 3.765847         dc
```

Let's show all of the variables in our regression where the studentized residual exceeds +2 or -2, i.e., where the absolute value of the residual exceeds 2. We see the data for the three potential outliers we identified, namely Florida, Mississippi and Washington D.C. Looking carefully at these three observations, we couldn't find any data entry error, though we may want to do another regression analysis with the extreme point such as DC deleted. We will return to this issue later.

```
      list r crime pctmetro poverty single if abs(r) > 2

                   r      crime   pctmetro    poverty     single
        1. -3.570789        434       30.7       24.7       14.7
       50.  2.619523       1206         93       17.8       10.6
       51.  3.765847       2922        100       26.4       22.1
```

Now let's look at the leverage's to identify observations that will have potential great influence on regression coefficient estimates.

```
      predict lev, leverage
      stem lev

      Stem-and-leaf plot for l (Leverage)

      l rounded to nearest multiple of .001
      plot in units of .001

        0** | 20,24,24,28,29,29,31,31,32,32,34,35,37,38,39,43,45,45,46,47,49
        0** |
      50,57,60,61,62,63,63,64,64,67,72,72,73,76,76,82,83,85,85,85,91,95
        1** | 00,02,36
        1** | 65,80,91
        2** |
        2** | 61
        3** |
        3** |
        4** |
        4** |
        5** | 36
```

We use the **show(5) high** options on the **hilo** command to show just the 5 largest observations (the **high** option can be abbreviated as **h**). We see that DC has the largest leverage.

```
hilo lev state, show(5) high

5 largest observations on lev

      lev      state
```

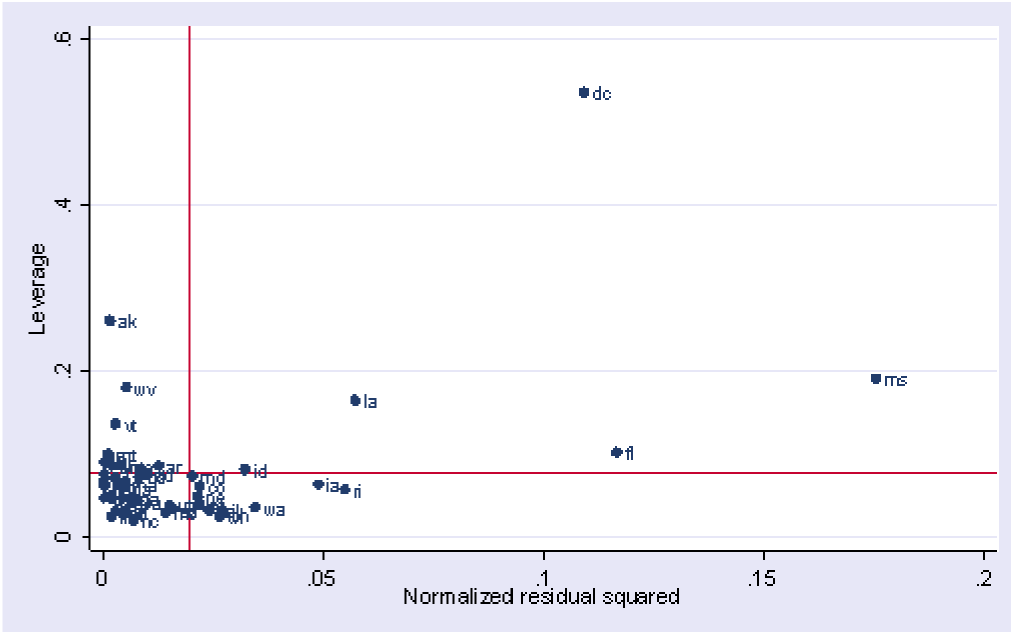
```
.1652769          la
.1802005          wv
 .191012          ms
.2606759          ak
 .536383          dc
```

Generally, a point with leverage greater than **(2k+2)/n** should be carefully examined. Here **k** is the number of predictors and **n** is the number of observations. In our example, we can do the following.

```
display (2*3+2)/51
.15686275

list crime pctmetro poverty single state lev if lev >.156

        crime   pctmetro    poverty     single      state   lev
  5.      208       41.8       22.2        9.4         wv   .1802005
 48.      761       41.8        9.1       14.3         ak   .2606759
 49.      434       30.7       24.7       14.7         ms    .191012
 50.     1062         75       26.4       14.9         la   .1652769
 51.     2922        100       26.4       22.1         dc    .536383
```

As we have seen, DC is an observation that both has a large residual and large leverage. Such points are potentially the most influential. We can make a plot that shows the leverage by the residual squared and look for observations that are jointly high on both of these measures. We can do this using the **lvr2plot** command. **lvr2plot** stands for leverage versus residual squared plot. Using residual squared instead of residual itself, the graph is restricted to the first quadrant and the relative positions of data points are preserved. This is a quick way of checking potential influential observations and outliers at the same time. Both types of points are of great concern for us.

```
lvr2plot, mlabel(state)
```



The two reference lines are the means for leverage, horizontal, and for the normalized residual squared, vertical. The points that immediately catch our attention is DC (with the largest leverage) and MS (with the largest residual squared). We'll look at those observations more carefully by listing them.

```
list state crime pctmetro poverty single if state=="dc" | state=="ms"
```

	state	crime	pctmetro	poverty	single
49.	ms	434	30.7	24.7	14.7
51.	dc	2922	100	26.4	22.1

Now let's move on to overall measures of influence, specifically let's look at Cook's D and DFITS. These measures both combine information on the residual and leverage. Cook's D and DFITS are very similar except that they scale differently but they give us similar answers.

The lowest value that Cook's D can assume is zero, and the higher the Cook's D is, the more influential the point. The convention cut-off point is **4/n**. We can list any observation above the cut-off point by doing the following. We do see that the Cook's D for DC is by far the largest.

```
predict d, cooksd
list crime pctmetro poverty single state d if d>4/51
```

	crime	pctmetro	poverty	single	state	d
1.	434	30.7	24.7	14.7	ms	.602106
2.	1062	75	26.4	14.9	la	.1592638
50.	1206	93	17.8	10.6	fl	.173629
51.	2922	100	26.4	22.1	dc	3.203429

Now let's take a look at DFITS. The cut-off point for DFITS is **2*sqrt(k/n)**. DFITS can be either positive or negative, with numbers close to zero corresponding to the points with small or zero influence. As we see, **dfit** also indicates that DC is, by far, the most influential observation.

```
predict dfit, dfits
list crime pctmetro poverty single state dfit if abs(dfit)>2*sqrt(3/51)

        crime   pctmetro    poverty     single       state        dfit
 18.     1206         93       17.8       10.6          fl    .8838196
 49.      434       30.7       24.7       14.7          ms   -1.735096
 50.     1062         75       26.4       14.9          la   -.8181195
 51.     2922        100       26.4       22.1          dc    4.050611
```

The above measures are general measures of influence. You can also consider more specific measures of influence that assess how each coefficient is changed by deleting the observation. This measure is called **DFBETA** and is created for each of the predictors. Apparently this is more computational intensive than summary statistics such as Cook's D since the more predictors a model has, the more computation it may involve. We can restrict our attention to only those predictors that we are most concerned with to see how well behaved those predictors are. In Stata, the **dfbeta** command will produce the DFBETAs for each of the predictors. The names for the new variables created are chosen by Stata automatically and begin with the letters DF.

```
dfbeta
                          DFpctmetro:  DFbeta(pctmetro)
                           DFpoverty:  DFbeta(poverty)
                            DFsingle:  DFbeta(single)
```

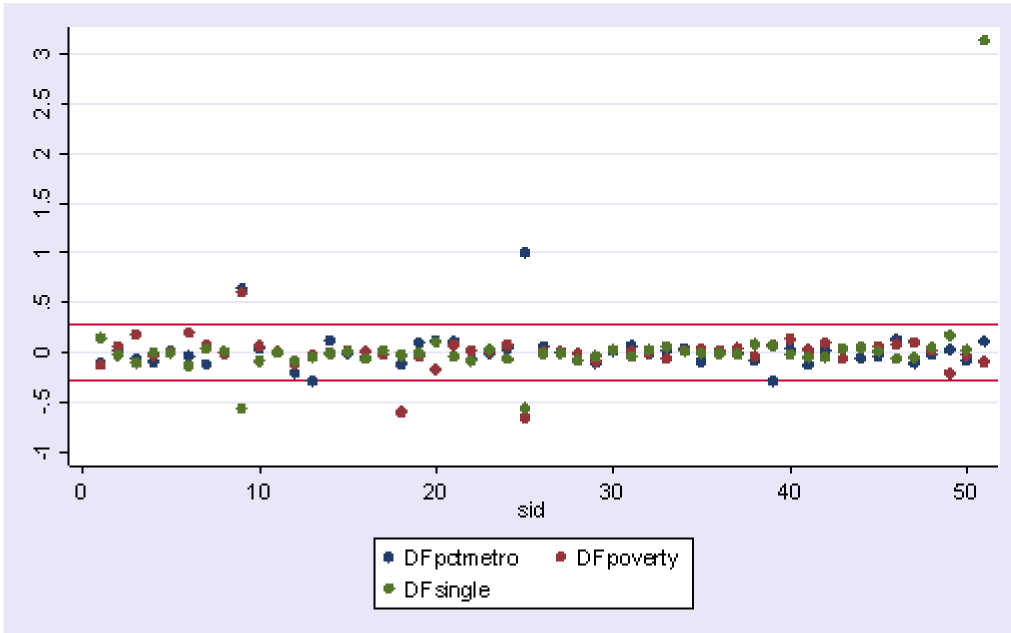
This created three variables, **DFpctmetro**, **DFpoverty** and **DFsingle**. Let's look at the first 5 values.

```
list state DFpctmetro DFpoverty DFsingle in 1/5

        state  DFpctme~o  DFpoverty   DFsingle
  1.       ak  -.1061846  -.1313398   .1451826
  2.       al   .0124287   .0552852  -.0275128
  3.       ar  -.0687483   .1753482  -.1052626
  4.       az  -.0947614  -.0308833    .001242
  5.       ca   .0126401   .0088009  -.0036361
```

The value for **DFsingle** for Alaska is .14, which means that by being included in the analysis (as compared to being excluded), Alaska increases the coefficient for **single** by 0.14 standard errors, i.e., .14 times the standard error for **BSingle** or by (0.14 * 15.5). Since the inclusion of an observation could either contribute to an increase or decrease in a regression coefficient, DFBETAs can be either positive or negative. A DFBETA value in excess of **2/sqrt(n)** merits further investigation. In this example, we would be concerned about absolute values in excess of 2/sqrt(51) or .28.

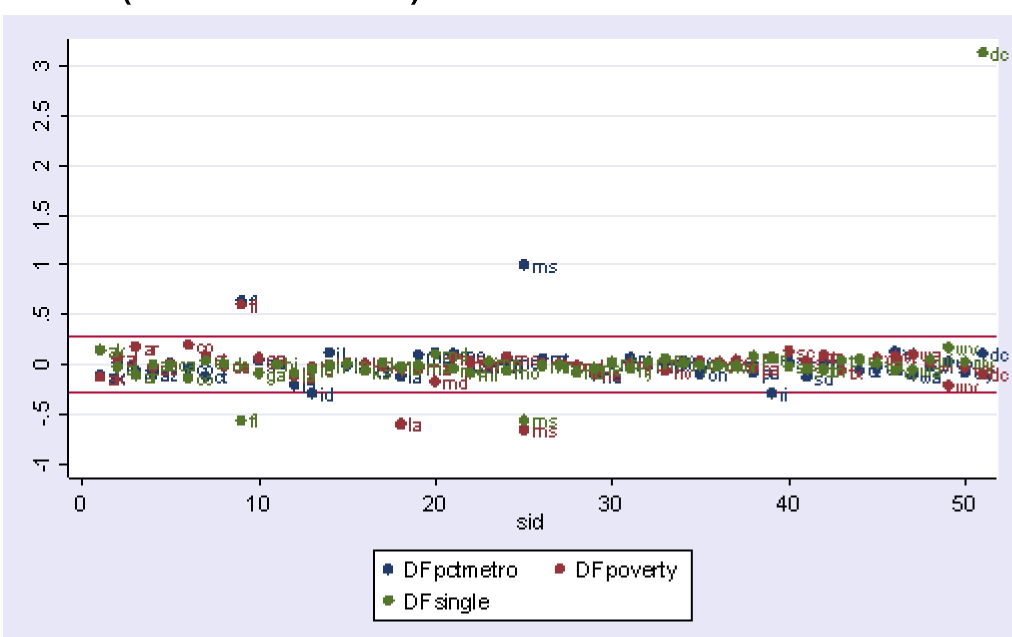
We can plot all three DFBETA values against the state id in one graph shown below. We add a line at .28 and -.28 to help us see potentially troublesome observations. We see the largest value is about 3.0 for **DFsingle**.

```
scatter DFpctmetro DFpoverty DFsingle sid, ylabel(-1(.5)3) yline(.28 -
.28)
```



We can repeat this graph with the **mlabel()** option in the graph command to label the points. With the graph above we can identify which DFBeta is a problem, and with the graph below we can associate that observation with the state that it originates from.

```
scatter DFpctmetro DFpoverty DFsingle sid, ylabel(-1(.5)3) yline(.28 -
.28) ///
  mlabel(state state state)
```



Now let's list those observations with **DFsingle** larger than the cut-off value.

```
list DFsingle state crime pctmetro poverty single if abs(DFsingle) >
2/sqrt(51)

      DFsingle      state     crime   pctmetro    poverty     single
  9. -.5606022         fl      1206         93       17.8       10.6
 25. -.5680245         ms       434       30.7       24.7       14.7
 51.  3.139084         dc      2922        100       26.4       22.1
```

The following table summarizes the general rules of thumb we use for these measures to identify observations worthy of further investigation (where k is the number of predictors and n is the number of observations).

Measure	**Value**
leverage	>(2k+2)/n
abs(rstu)	> 2
Cook's D	> 4/n
abs(DFITS)	> 2*sqrt(k/n)
abs(DFBETA)	> 2/sqrt(n)

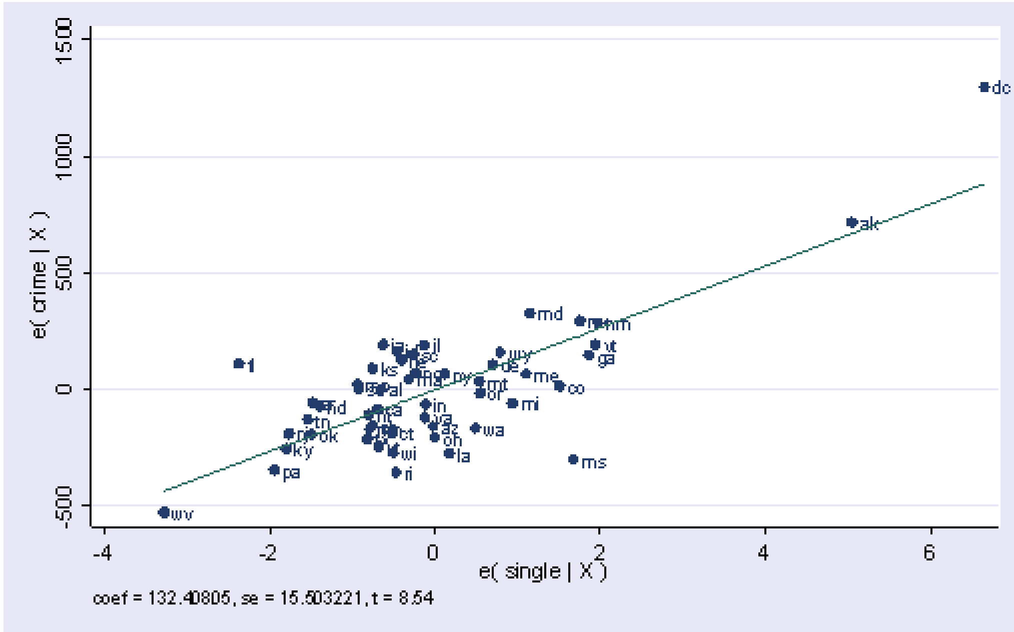
We have used the **predict** command to create a number of variables associated with regression analysis and regression diagnostics. The **help regress** command not only gives help on the regress command, but also lists all of the statistics that can be generated via the **predict** command. Below we show a snippet of the Stata help file illustrating the various statistics that can be computed via the **predict** command.

```
help regress

----------------------------------------------------------------------
--------
help for regress                                          (manual:  [R]
regress)
----------------------------------------------------------------------
--------

<--output omitted-->

The syntax of predict following regress is

        predict [type] newvarname [if exp] [in range] [, statistic]

where statistic is
```

```
        xb                             fitted values; the default
        pr(a,b)                        Pr(y |a>y>b)  (a and b may be 
numbers
        e(a,b)                         E(y |a>y>b)    or variables; a==. 
means
        ystar(a,b)                     E(y*)          -inf; b==. means 
inf)
        cooksd                         Cook's distance
        leverage | hat                 leverage (diagonal elements of hat 
matrix)
        residuals                      residuals
        rstandard                      standardized residuals
        rstudent                       Studentized (jackknifed) residuals
        stdp                           standard error of the prediction
        stdf                           standard error of the forecast
        stdr                           standard error of the residual
    (*) covratio                       COVRATIO
    (*) dfbeta(varname)                DFBETA for varname
    (*) dfits                          DFITS
    (*) welsch                         Welsch distance

Unstarred statistics are available both in and out of sample; type 
"predict ...
if e(sample) ..." if wanted only for the estimation sample.  Starred 
statistics
are calculated for the estimation sample even when "if e(sample)" is 
not speci-
fied.

<--more output omitted here.-->
```

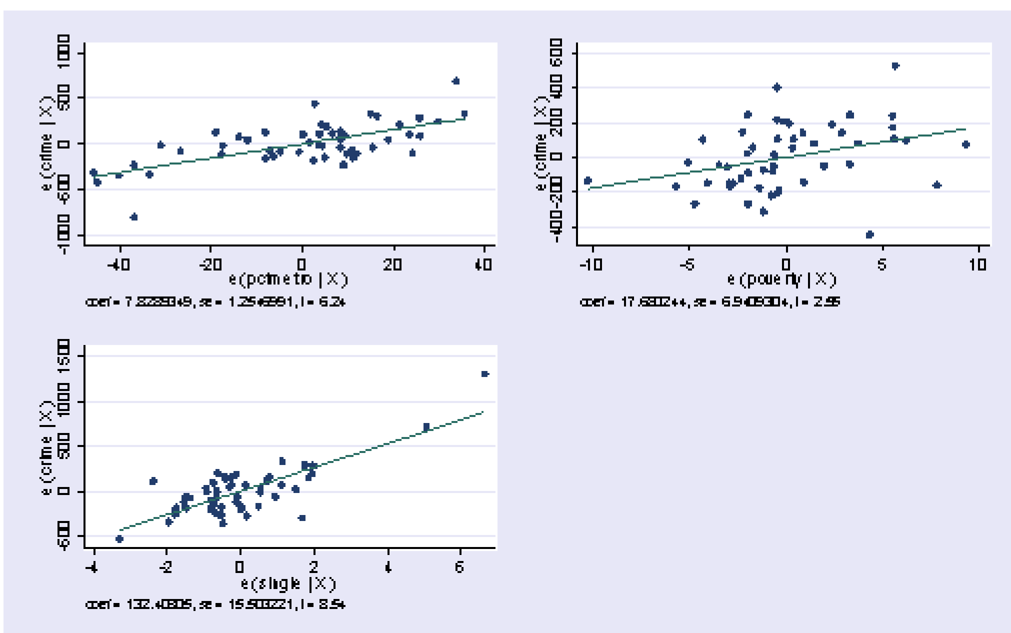
We have explored a number of the statistics that we can get after the **regress** command. There are also several graphs that can be used to search for unusual and influential observations. The **avplot** command graphs an *added-variable plot*. It is also called a *partial-regression* plot and is very useful in identifying influential points. For example, in the avplot for **single** shown below, the graph shows **crime** by **single** after both **crime** and **single** have been adjusted for all other predictors in the model. The line plotted has the same slope as the coefficient for **single**. This plot shows how the observation for DC influences the coefficient. You can see how the regression line is tugged upwards trying to fit through the extreme value of DC. Alaska and West Virginia may also exert substantial leverage on the coefficient of **single**.

```
avplot single, mlabel(state)
```



Stata also has the **avplots** command that creates an added variable plot for all of the variables, which can be very useful when you have many variables. It does produce small graphs, but these graphs can quickly reveal whether you have problematic observations based on the added variable plots.

```
avplots
```



DC has appeared as an outlier as well as an influential point in every analysis. Since DC is really not a state, we can use this to justify omitting it from the analysis saying that we really wish to just analyze states. First, let's repeat our analysis including DC by just typing **regress**.

```
regress

  Source |       SS       df       MS                  Number of obs =
51
---------+------------------------------               F(  3,    47) =
82.16
   Model |  8170480.21     3  2723493.40               Prob > F      =
0.0000
Residual |  1557994.53    47  33148.8199               R-squared     =
0.8399
---------+------------------------------               Adj R-squared =
0.8296
   Total |  9728474.75    50  194569.495               Root MSE      =
182.07

------------------------------------------------------------------------
-------
   crime |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
pctmetro |   7.828935   1.254699      6.240   0.000       5.304806
10.35306
 poverty |   17.68024    6.94093      2.547   0.014       3.716893
31.64359
  single |   132.4081   15.50322      8.541   0.000       101.2196
163.5965
   _cons |  -1666.436    147.852    -11.271   0.000      -1963.876   -
1368.996
------------------------------------------------------------------------
-------
```

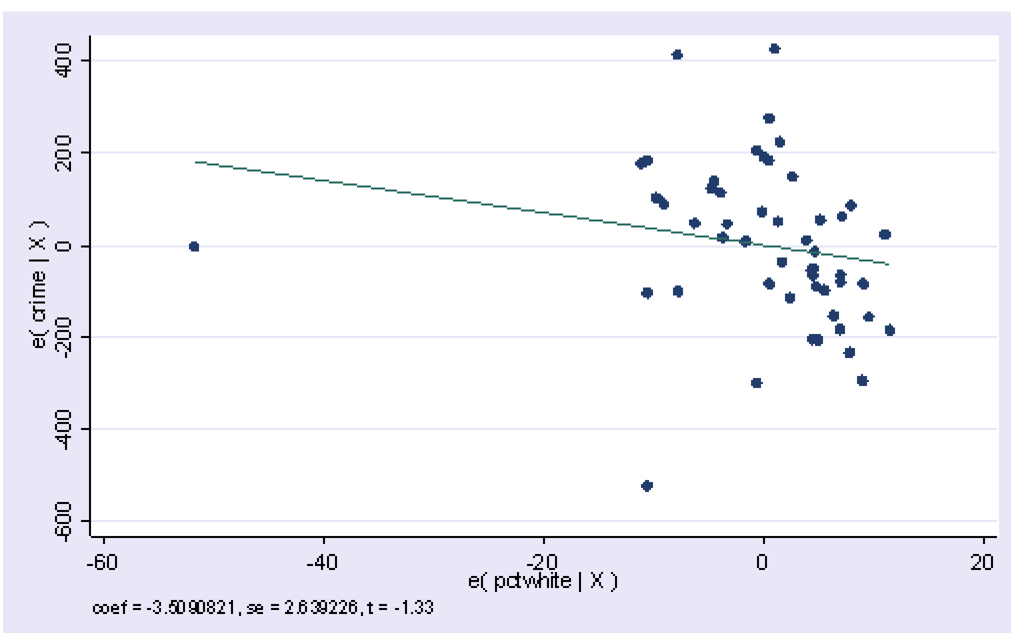
Now, let's run the analysis omitting DC by including **if state != "dc"** on the **regress** command (here **!=** stands for "not equal to" but you could also use **~=** to mean the same thing). As we expect, deleting DC made a large change in the coefficient for **single**. The coefficient for **single** dropped from 132.4 to 89.4. After having deleted DC, we would repeat the process we have illustrated in this section to search for any other outlying and influential observations.

```
regress crime pctmetro poverty single if state!="dc"

  Source |       SS       df       MS                  Number of obs =
50
---------+------------------------------               F(  3,    46) =
39.90
   Model |  3098767.11     3  1032922.37               Prob > F      =
0.0000
Residual |  1190858.11    46  25888.2199               R-squared     =
0.7224
---------+------------------------------               Adj R-squared =
0.7043
```

```
   Total |  4289625.22    49  87543.3718               Root MSE      =
160.90

------------------------------------------------------------------------------
-
   crime |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
pctmetro |   7.712334   1.109241      6.953   0.000       5.479547
9.94512
 poverty |   18.28265   6.135958      2.980   0.005       5.931611
30.6337
  single |   89.40078   17.83621      5.012   0.000       53.49836
125.3032
   _cons |  -1197.538   180.4874     -6.635   0.000       -1560.84   -
834.2358
------------------------------------------------------------------------------
-
```

Finally, we showed that the **avplot** command can be used to searching for outliers among existing variables in your model, but we should note that the **avplot** command not only works for the variables in the model, it also works for variables that are not in the model, which is why it is called *added-variable plot*. Let's use the regression that includes DC as we want to continue to see ill-behavior caused by DC as a demonstration for doing regression diagnostics. We can do an **avplot** on variable **pctwhite**.

```
regress crime pctmetro poverty single
avplot pctwhite
```



At the top of the plot, we have "coef=-3.509". It is the coefficient for **pctwhite** if it were put in the model. We can check that by doing a regression as below.

regress crime pctmetro pctwhite poverty single

```
  Source |       SS       df       MS               Number of obs =
51
---------+------------------------------               F(  4,    46) =
63.07
   Model |  8228138.87     4  2057034.72               Prob > F      =
0.0000
Residual |  1500335.87    46  32615.9972               R-squared     =
0.8458
---------+------------------------------               Adj R-squared =
0.8324
   Total |  9728474.75    50  194569.495               Root MSE      =
180.60

---------------------------------------------------------------------
-------
   crime |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-----------------------------------------------------------
-------
pctmetro |   7.404075   1.284941      5.762   0.000       4.817623
9.990526
pctwhite |  -3.509082   2.639226     -1.330   0.190      -8.821568
1.803404
 poverty |   16.66548   6.927095      2.406   0.020       2.721964
30.609
  single |   120.3576    17.8502      6.743   0.000       84.42702
156.2882
   _cons |  -1191.689   386.0089     -3.087   0.003      -1968.685   -
414.6936
---------------------------------------------------------------------
-------
```

Summary

In this section, we explored a number of methods of identifying outliers and influential points. In a typical analysis, you would probably use only some of these methods. Generally speaking, there are two types of methods for assessing outliers: statistics such as residuals, leverage, Cook's D and DFITS, that assess the overall impact of an observation on the regression results, and statistics such as DFBETA that assess the specific impact of an observation on the regression coefficients.

In our example, we found that DC was a point of major concern. We performed a regression with it and without it and the regression equations were very different. We can justify removing it from our analysis by reasoning that our model is to predict crime rate for states, not for metropolitan areas.

2.2 Checking Normality of Residuals

Many researchers believe that multiple regression requires normality. This is not the case. Normality of residuals is only required for valid hypothesis testing, that is, the normality assumption assures that the p-values for the t-tests and F-test will be valid. Normality is not required in order to obtain unbiased estimates of the regression coefficients. OLS regression merely requires that the residuals (errors) be identically and independently distributed. Furthermore, there is no assumption or requirement that the predictor variables be normally distributed. If this were the case than we would not be able to use dummy coded variables in our models.

After we run a regression analysis, we can use the **predict** command to create residuals and then use commands such as **kdensity**, **qnorm** and **pnorm** to check the normality of the residuals.

Let's use the **elemapi2** data file we saw in Chapter 1 for these analyses. Let's predict academic performance (**api00**) from percent receiving free meals (**meals**), percent of English language learners (**ell**), and percent of teachers with emergency credentials (**emer**).

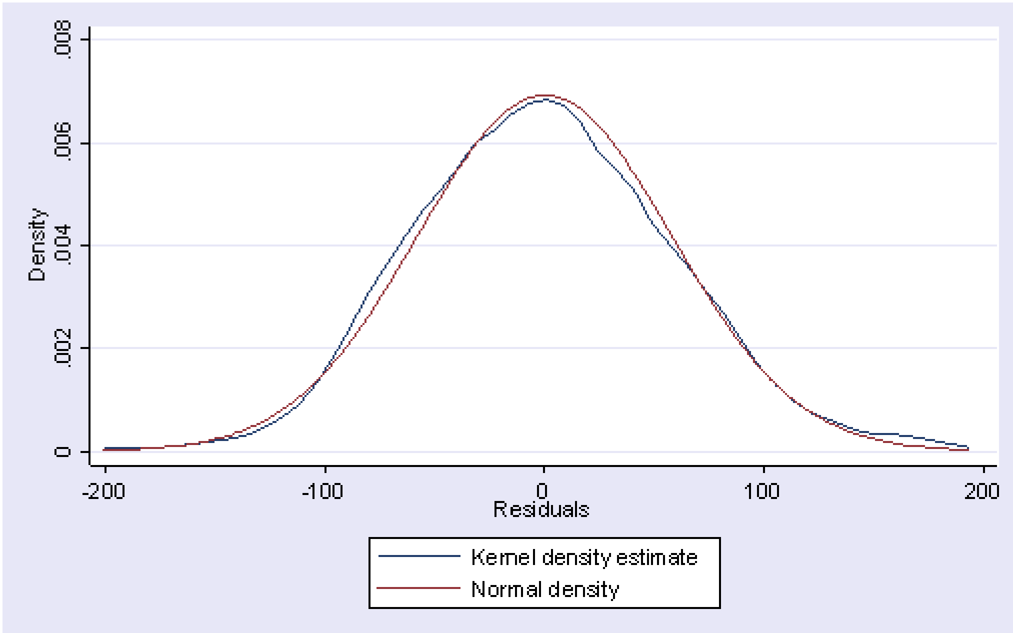
```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2
regress api00 meals ell emer

  Source |       SS       df       MS               Number of obs =
400
---------+------------------------------            F(  3,   396) =
673.00
   Model |  6749782.75     3  2249927.58            Prob > F      =
0.0000
Residual |  1323889.25   396  3343.15467            R-squared     =
0.8360
---------+------------------------------            Adj R-squared =
0.8348
   Total |  8073672.00   399  20234.7669            Root MSE      =
57.82

------------------------------------------------------------------------------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
   meals |  -3.159189   .1497371    -21.098   0.000      -3.453568   -
2.864809
     ell |  -.9098732   .1846442     -4.928   0.000      -1.272878   -
.5468678
    emer |  -1.573496    .293112     -5.368   0.000      -2.149746   -
.9972456
   _cons |   886.7033    6.25976    141.651   0.000       874.3967
899.0098
------------------------------------------------------------------------------
```

We then use the **predict** command to generate residuals.

```
predict r, resid
```

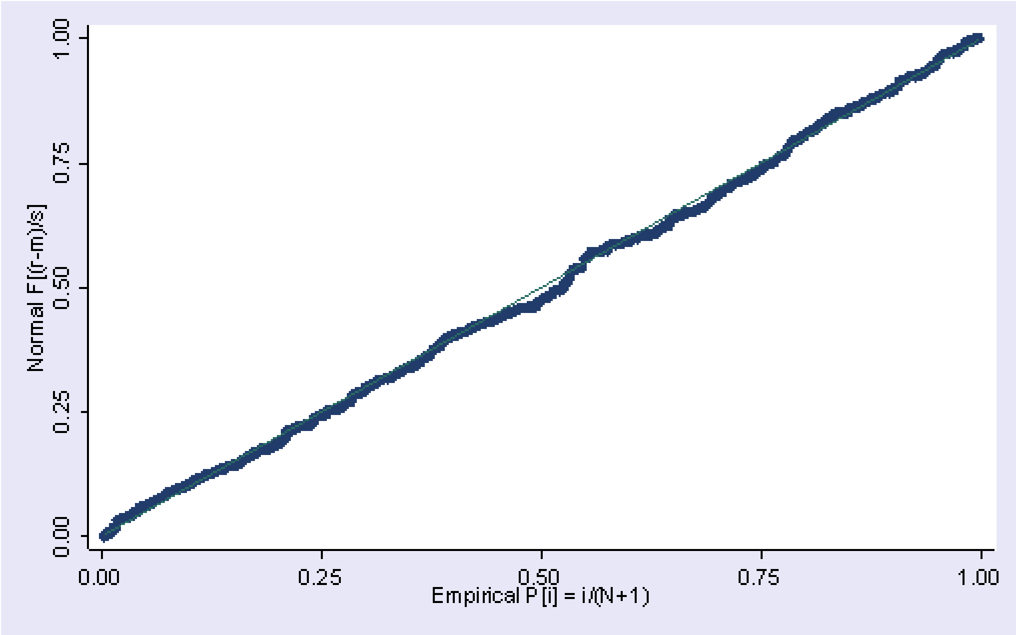
Below we use the **kdensity** command to produce a kernel density plot with the **normal** option requesting that a normal density be overlaid on the plot. **kdensity** stands for kernel density estimate. It can be thought of as a histogram with narrow bins and moving average.

```
kdensity r, normal
```

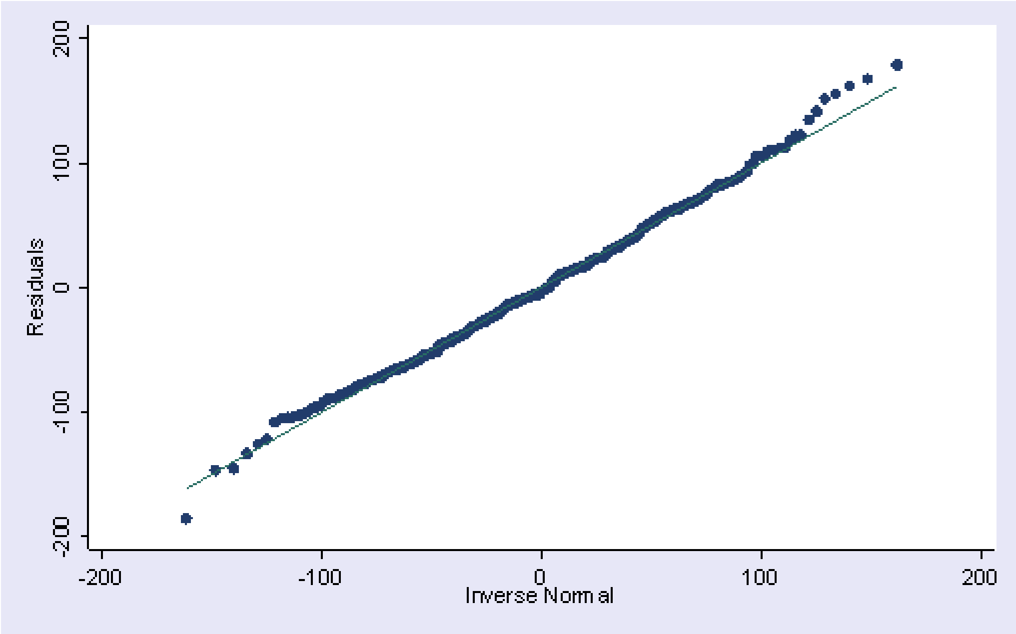


The **pnorm** command graphs a standardized normal probability (P-P) plot while **qnorm** plots the quantiles of a variable against the quantiles of a normal distribution. **pnorm** is sensitive to non-normality in the middle range of data and **qnorm** is sensitive to non-normality near the tails. As you see below, the results from **pnorm** show no indications of non-normality, while the **qnorm** command shows a slight deviation from normal at the upper tail, as can be seen in the **kdensity** above. Nevertheless, this seems to be a minor and trivial deviation from normality. We can accept that the residuals are close to a normal distribution.

```
pnorm r
```



qnorm r



There are also numerical tests for testing normality. One of the tests is the test written by Lawrence C. Hamilton, Dept. of Sociology, Univ. of New Hampshire, called **iqr**. You can get this program from Stata by typing **findit iqr** (see How can I used the findit command to search for programs and get additional help? for more information about using **findit**).

iqr stands for inter-quartile range and assumes the symmetry of the distribution. Severe outliers consist of those points that are either 3 inter-quartile-ranges below the first quartile or 3 inter-quartile-ranges above the third quartile. The presence of any severe outliers should be sufficient evidence to reject normality at a 5% significance level. Mild outliers are common in samples of any size. In our case, we don't have any severe outliers and the distribution seems fairly symmetric. The residuals have an approximately normal distribution.

```
iqr r

   mean=  7.4e-08         std.dev.=   57.6          (n= 400)
 median= -3.657    pseudo std.dev.=  56.69        (IQR=  76.47)
10 trim= -1.083
                                                 low         high
                                                 -------------------
                                  inner fences   -154.7       151.2
                               # mild outliers   1           5
                               % mild outliers   0.25%       1.25%

                                  outer fences   -269.4       265.9
                             # severe outliers   0           0
                             % severe outliers   0.00%       0.00%
```

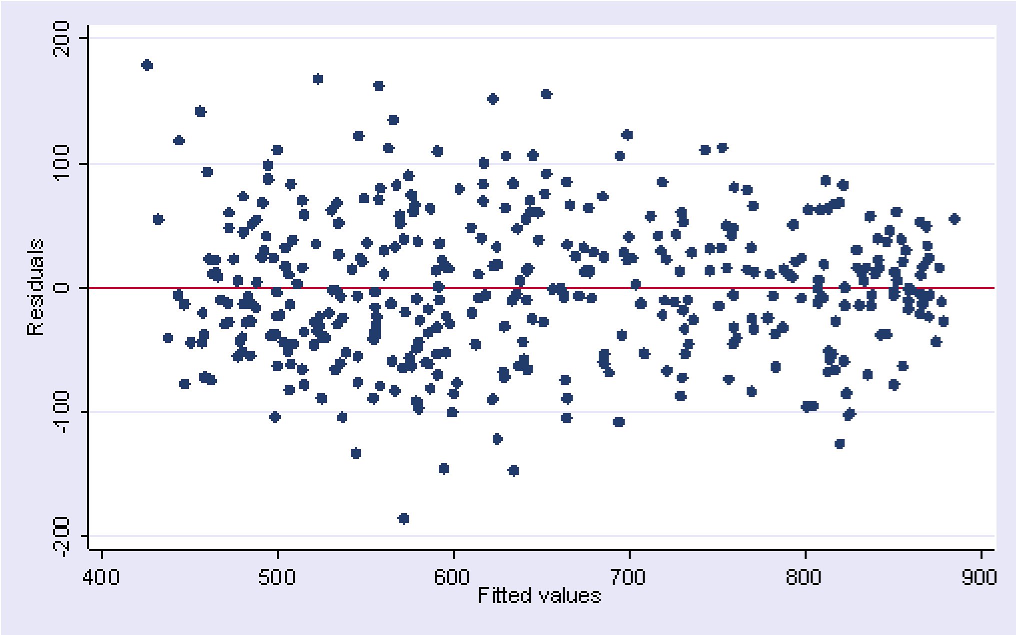
Another test available is the **swilk** test which performs the Shapiro-Wilk W test for normality. The p-value is based on the assumption that the distribution is normal. In our example, it is very large (.51), indicating that we cannot reject that **r** is normally distributed.

```
swilk r

                   Shapiro-Wilk W test for normal data
 Variable |    Obs          W         V          z   Pr > z
---------+-------------------------------------------------
        r |    400     0.99641     0.989     -0.025  0.51006
```

2.3 Checking Homoscedasticity of Residuals

One of the main assumptions for the ordinary least squares regression is the homogeneity of variance of the residuals. If the model is well-fitted, there should be no pattern to the residuals plotted against the fitted values. If the variance of the residuals is non-constant then the residual variance is said to be "heteroscedastic." There are graphical and non-graphical methods for detecting heteroscedasticity. A commonly used graphical method is to plot the residuals versus fitted (predicted) values. We do this by issuing the **rvfplot** command. Below we use the **rvfplot** command with the **yline(0)** option to put a reference line at y=0. We see that the pattern of the data points is getting a little narrower towards the right end, which is an indication of heteroscedasticity.

```
rvfplot, yline(0)
```



Now let's look at a couple of commands that test for heteroscedasticity.

estat imtest

```
Cameron & Trivedi's decomposition of IM-test

---------------------------------------------------
              Source |       chi2     df      p
---------------------+-----------------------------
  Heteroskedasticity |      18.35      9    0.0313
            Skewness |       7.78      3    0.0507
            Kurtosis |       0.27      1    0.6067
---------------------+-----------------------------
               Total |      26.40     13    0.0150
---------------------------------------------------
```

estat hettest

```
Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
         Ho: Constant variance
         Variables: fitted values of api00
         chi2(1)      =     8.75
         Prob > chi2  =   0.0031
```

The first test on heteroskedasticity given by **imest** is the White's test and the second one given by **hettest** is the Breusch-Pagan test. Both test the null hypothesis that the variance of the residuals is homogenous. Therefore, if the p-value is very small, we would have to reject the hypothesis and accept the alternative hypothesis that the variance is not homogenous. So in this case, the evidence is against the null hypothesis that the variance is homogeneous. These tests are very sensitive to model assumptions, such as the assumption of normality. Therefore it is a common practice to combine the tests with diagnostic plots to make a judgment on the severity of the heteroscedasticity and to decide if any correction is needed for heteroscedasticity. In our case, the plot above does not show too strong an evidence. So we are not going to get into details on how to correct for heteroscedasticity even though there are methods available.

2.4 Checking for Multicollinearity

When there is a perfect linear relationship among the predictors, the estimates for a regression model cannot be uniquely computed. The term collinearity implies that two variables are near perfect linear combinations of one another. When more than two variables are involved it is often called multicollinearity, although the two terms are often used interchangeably.

The primary concern is that as the degree of multicollinearity increases, the regression model estimates of the coefficients become unstable and the standard errors for the coefficients can get wildly inflated. In this section, we will explore some Stata commands that help to detect multicollinearity.

We can use the **vif** command after the regression to check for multicollinearity. **vif** stands for *variance inflation factor*. As a rule of thumb, a variable whose VIF values are greater than 10 may merit further investigation. Tolerance, defined as 1/VIF, is used by many researchers to check on the degree of collinearity. A tolerance value lower than 0.1 is comparable to a VIF of 10. It means that the variable could be considered as a linear combination of other independent variables. Let's first look at the regression we did from the last section, the regression model predicting **api00** from **meals, ell** and **emer** and then issue the **vif** command.

```
regress api00 meals ell emer

<-- output omitted -->

vif

Variable |       VIF       1/VIF
---------+----------------------
   meals |      2.73    0.366965
     ell |      2.51    0.398325
    emer |      1.41    0.706805
---------+----------------------
Mean VIF |      2.22
```

The VIFs look fine here. Here is an example where the VIFs are more worrisome.

```
regress api00 acs_k3 avg_ed grad_sch col_grad some_col

  Source |       SS       df       MS                  Number of obs =
379
---------+------------------------------               F(  5,   373) =
143.79
   Model |  5056268.54     5  1011253.71               Prob > F      =
0.0000
Residual |  2623191.21   373  7032.68421               R-squared     =
0.6584
```

```
---------+------------------------------               Adj R-squared =
0.6538
   Total |  7679459.75   378  20316.0311               Root MSE      =
83.861

----------------------------------------------------------------------
-------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+------------------------------------------------------------
-------
  acs_k3 |   11.45725   3.275411      3.498   0.001        5.01667
17.89784
  avg_ed |   227.2638    37.2196      6.106   0.000       154.0773
300.4504
grad_sch |  -2.090898   1.352292     -1.546   0.123      -4.749969
.5681734
col_grad |  -2.967831   1.017812     -2.916   0.004      -4.969199   -
.9664627
some_col |  -.7604543   .8109676     -0.938   0.349      -2.355096
.8341871
   _cons |  -82.60913   81.84638     -1.009   0.313      -243.5473
78.32903
----------------------------------------------------------------------
-------
```

```
vif
```

```
Variable |      VIF       1/VIF
---------+----------------------
  avg_ed |     43.57    0.022951
grad_sch |     14.86    0.067274
col_grad |     14.78    0.067664
some_col |      4.07    0.245993
  acs_k3 |      1.03    0.971867
---------+----------------------
Mean VIF |     15.66
```

In this example, the VIF and tolerance (1/VIF) values for **avg_ed grad_sch** and **col_grad** are worrisome. All of these variables measure education of the parents and the very high VIF values indicate that these variables are possibly redundant. For example, after you know **grad_sch** and **col_grad**, you probably can predict **avg_ed** very well. In this example, multicollinearity arises because we have put in too many variables that measure the same thing, parent education.

Let's omit one of the parent education variables, **avg_ed**. Note that the VIF values in the analysis below appear much better. Also, note how the standard errors are reduced for the parent education variables, **grad_sch** and **col_grad**. This is because the high degree of collinearity caused the standard errors to be inflated. With the multicollinearity eliminated, the coefficient for **grad_sch**, which had been non-significant, is now significant.

```
regress api00 acs_k3 grad_sch col_grad some_col

  Source |       SS       df       MS                  Number of obs =
398
```

```
---------+------------------------------               F(  4,   393) =
107.12
   Model |  4180144.34     4  1045036.09               Prob > F      =
0.0000
Residual |  3834062.79   393  9755.88497               R-squared     =
0.5216
---------+------------------------------               Adj R-squared =
0.5167
   Total |  8014207.14   397  20186.9197               Root MSE      =
98.772

------------------------------------------------------------------------
-------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  acs_k3 |    11.7126   3.664872      3.196   0.002       4.507392
18.91781
grad_sch |   5.634762   .4581979     12.298   0.000       4.733937
6.535588
col_grad |   2.479916   .3395548      7.303   0.000       1.812345
3.147487
some_col |   2.158271   .4438822      4.862   0.000        1.28559
3.030952
   _cons |   283.7446   70.32475      4.035   0.000       145.4849
422.0044
------------------------------------------------------------------------
-------
```

```
vif

Variable |      VIF      1/VIF
---------+----------------------
col_grad |      1.28    0.782726
grad_sch |      1.26    0.792131
some_col |      1.03    0.966696
  acs_k3 |      1.02    0.976666
---------+----------------------
Mean VIF |      1.15
```

Let's introduce another command on collinearity. The **collin** command displays several different measures of collinearity. For example, we can test for collinearity among the variables we used in the two examples above. Note that the **collin** command does not need to be run in connection with a **regress** command, unlike the **vif** command which follows a **regress** command. Also note that only predictor (independent) variables are used with the **collin** command. You can download **collin** from within Stata by typing **findit collin** (see How can I used the findit command to search for programs and get additional help? for more information about using **findit**).

```
collin acs_k3  avg_ed grad_sch col_grad some_col

  Collinearity Diagnostics

                        SQRT                                 Cond
  Variable       VIF    VIF    Tolerance  Eigenval           Index
```

```
    ---------------------------------------------------------------
      acs_k3      1.03    1.01    0.9719     2.4135     1.0000
      avg_ed     43.57    6.60    0.0230     1.0917     1.4869
    grad_sch     14.86    3.86    0.0673     0.9261     1.6144
    col_grad     14.78    3.84    0.0677     0.5552     2.0850
    some_col      4.07    2.02    0.2460     0.0135    13.3729
    ---------------------------------------------------------------
    Mean VIF     15.66              Condition Number   13.3729
```

We now remove **avg_ed** and see the collinearity diagnostics improve considerably.

```
collin acs_k3 grad_sch col_grad some_col

  Collinearity Diagnostics

                         SQRT                               Cond
  Variable        VIF    VIF     Tolerance  Eigenval       Index
---------------------------------------------------------------
    acs_k3       1.02    1.01    0.9767      1.5095      1.0000
  grad_sch       1.26    1.12    0.7921      1.0407      1.2043
  col_grad       1.28    1.13    0.7827      0.9203      1.2807
  some_col       1.03    1.02    0.9667      0.5296      1.6883
---------------------------------------------------------------
  Mean VIF       1.15              Condition Number    1.6883
```

The *condition number* is a commonly used index of the global instability of the regression coefficients -- a large condition number, 10 or more, is an indication of instability.

2.5 Checking Linearity

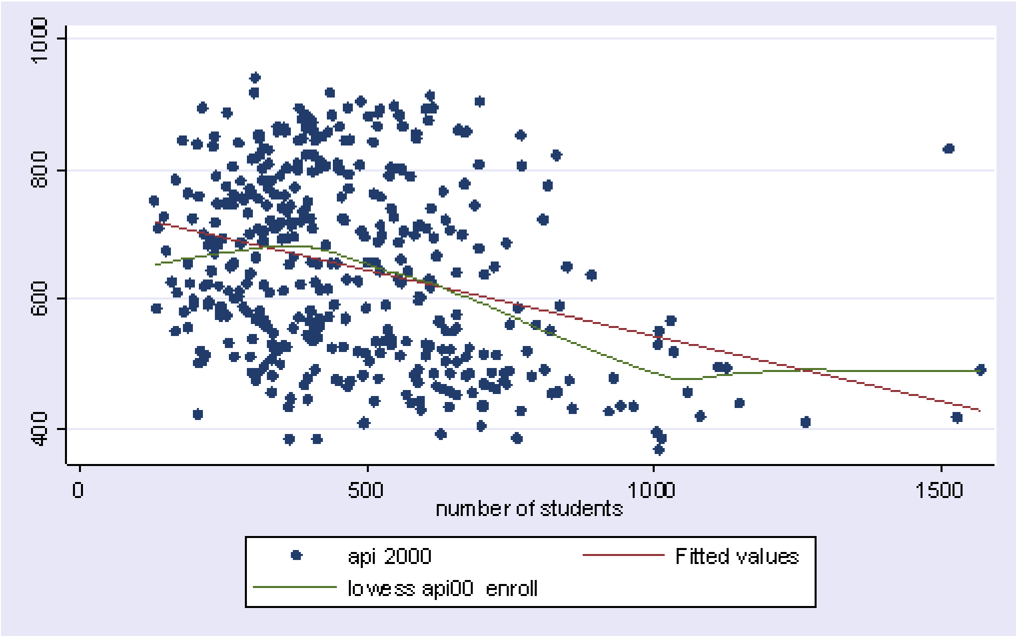
When we do linear regression, we assume that the relationship between the response variable and the predictors is linear. This is the assumption of linearity. If this assumption is violated, the linear regression will try to fit a straight line to data that does not follow a straight line. Checking the linear assumption in the case of simple regression is straightforward, since we only have one predictor. All we have to do is a scatter plot between the response variable and the predictor to see if nonlinearity is present, such as a curved band or a big wave-shaped curve. For example, recall we did a simple linear regression in Chapter 1 using dataset **elemapi2**.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2
regress api00 enroll

  Source |       SS       df       MS                  Number of obs =
400
---------+------------------------------               F(  1,   398) =
44.83
   Model |  817326.293     1  817326.293               Prob > F      =
0.0000
Residual |  7256345.70   398  18232.0244               R-squared     =
0.1012
---------+------------------------------               Adj R-squared =
0.0990
   Total |  8073672.00   399  20234.7669               Root MSE      =
135.03
```

```
------------------------------------------------------------------------------
       api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
      enroll |  -.1998674   .0298512     -6.695   0.000      -.2585532   -.1411817
       _cons |   744.2514   15.93308     46.711   0.000       712.9279    775.5749
------------------------------------------------------------------------------
```

Below we use the **scatter** command to show a scatterplot predicting **api00** from **enroll** and use **lfit** to show a linear fit, and then **lowess** to show a lowess smoother predicting **api00** from **enroll**. We clearly see some degree of nonlinearity.

```
twoway (scatter api00 enroll) (lfit api00 enroll) (lowess api00 enroll)
```



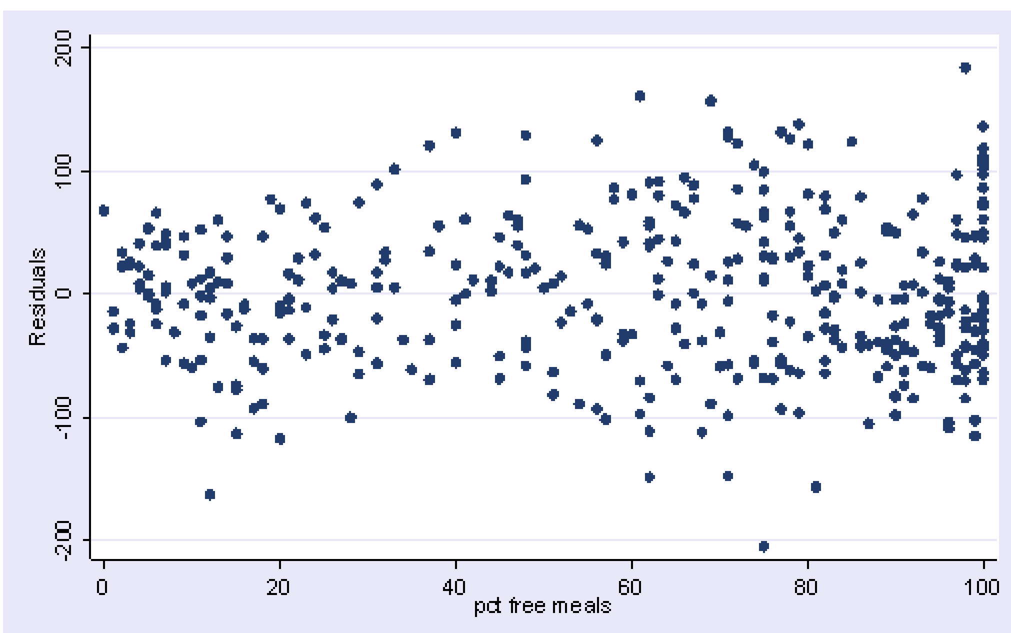
Checking the linearity assumption is not so straightforward in the case of multiple regression. We will try to illustrate some of the techniques that you can use. The most straightforward thing to do is to plot the standardized residuals against each of the predictor variables in the regression model. If there is a clear nonlinear pattern, there is a problem of nonlinearity. Otherwise, we should see for each of the plots just a random scatter of points. Let's continue to use dataset **elemapi2** here. Let's use a different model.

```
regress api00 meals some_col

      Source |       SS       df       MS                  Number of obs = 400
```

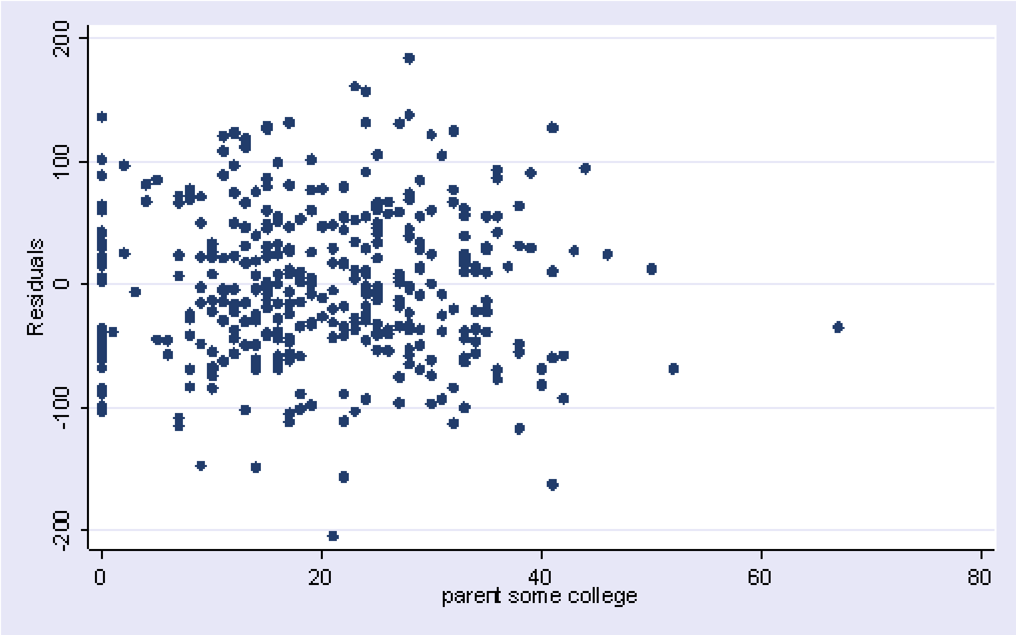
```
---------+------------------------------                F(  2,   397) =
877.98
   Model |  6584905.75     2  3292452.87                Prob > F      =
0.0000
Residual |  1488766.25   397  3750.04094                R-squared     =
0.8156
---------+------------------------------                Adj R-squared =
0.8147
   Total |  8073672.00   399  20234.7669                Root MSE      =
61.238

------------------------------------------------------------------------
-------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
   meals |     -3.949   .0984576    -40.109   0.000      -4.142563   -
3.755436
some_col |   .8476549   .2771428      3.059   0.002        .302804
1.392506
   _cons |    869.097   9.417734     92.283   0.000       850.5822
887.6119
------------------------------------------------------------------------
-------
```

predict r, resid

scatter r meals



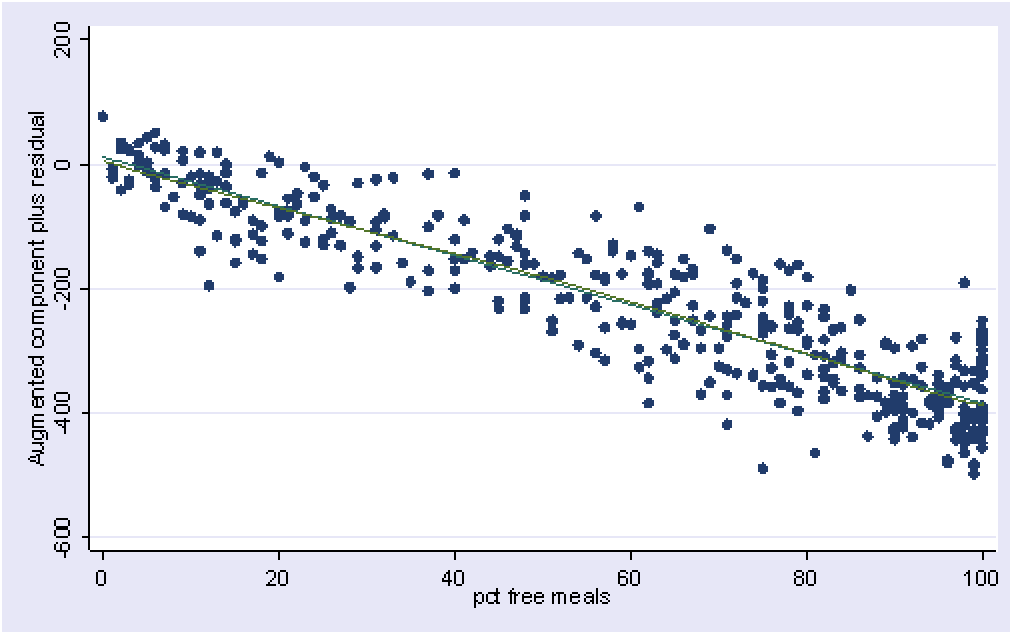
scatter r some_col



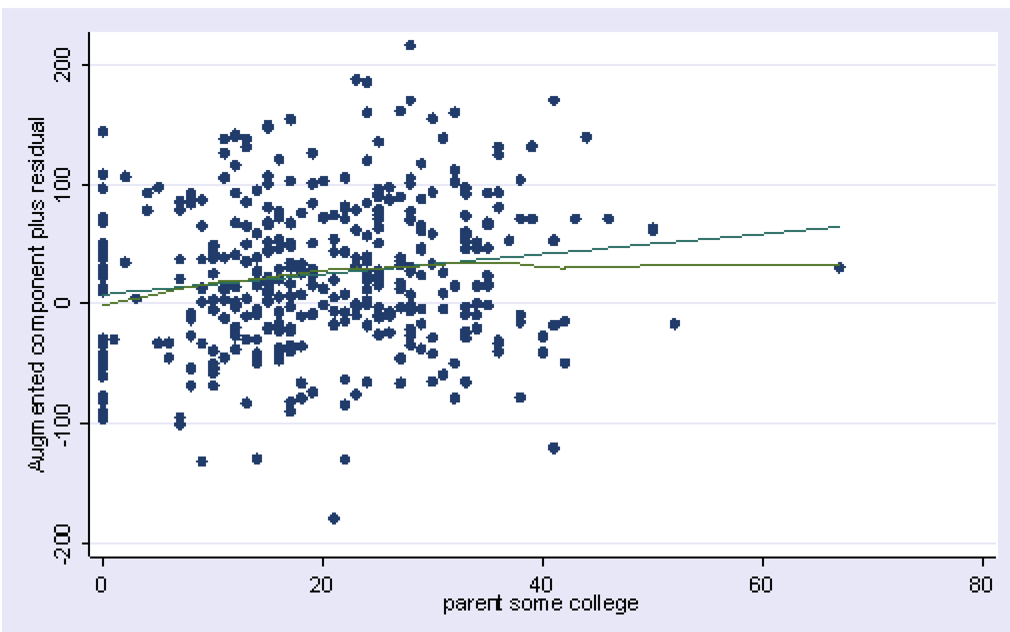
The two residual versus predictor variable plots above do not indicate strongly a clear departure from linearity. Another command for detecting non-linearity is **acprplot**. **acprplot** graphs an augmented component-plus-residual plot, a.k.a. augmented partial residual plot. It can be used to identify nonlinearities in the data. Let's use the **acprplot** command for **meals** and **some_col** and use the **lowess lsopts(bwidth(1))** options to request lowess smoothing with a bandwidth of 1.

In the first plot below the smoothed line is very close to the ordinary regression line, and the entire pattern seems pretty uniform. The second plot does seem more problematic at the right end. This may come from some potential influential points. Overall, they don't look too bad and we shouldn't be too concerned about non-linearities in the data.

```
acprplot meals, lowess lsopts(bwidth(1))
```




```
acprplot some_col, lowess lsopts(bwidth(1))
```



We have seen how to use **acprplot** to detect nonlinearity. However our last example didn't show much nonlinearity. Let's look at a more interesting example. This example is taken from "Statistics with Stata 5" by Lawrence C. Hamilton (1997, Duxbery Press). The dataset we will use is called **nations.dta**. We can get the dataset from the Internet.

```
use http://www.ats.ucla.edu/stat/stata/examples/sws5/nations
(Data on 109 countries)
```

```
describe

Contains data from
http://www.ats.ucla.edu/stat/stata/examples/sws5/nations.dta
  obs:           109                          Data on 109 countries
 vars:            15                          22 Dec 1996 20:12
 size:         4,033 (98.3% of memory free)
-----------------------------------------------------------------------
--------
   1. country   str8   %9s                    Country
   2. pop       float  %9.0g                  1985 population in
millions
   3. birth     byte   %8.0g                  Crude birth rate/1000
people
   4. death     byte   %8.0g                  Crude death rate/1000
people
   5. chldmort  byte   %8.0g                  Child (1-4 yr) mortality
1985
   6. infmort   int    %8.0g                  Infant (<1 yr) mortality
1985
   7. life      byte   %8.0g                  Life expectancy at birth
1985
   8. food      int    %8.0g                  Per capita daily calories
1985
   9. energy    int    %8.0g                  Per cap energy consumed,
kg oil
  10. gnpcap    int    %8.0g                  Per capita GNP 1985
  11. gnpgro    float  %9.0g                  Annual GNP growth % 65-85
  12. urban     byte   %8.0g                  % population urban 1985
  13. school1   int    %8.0g                  Primary enrollment % age-
group
  14. school2   byte   %8.0g                  Secondary enroll % age-
group
  15. school3   byte   %8.0g                  Higher ed. enroll % age-
group
-----------------------------------------------------------------------
--------
Sorted by:
```

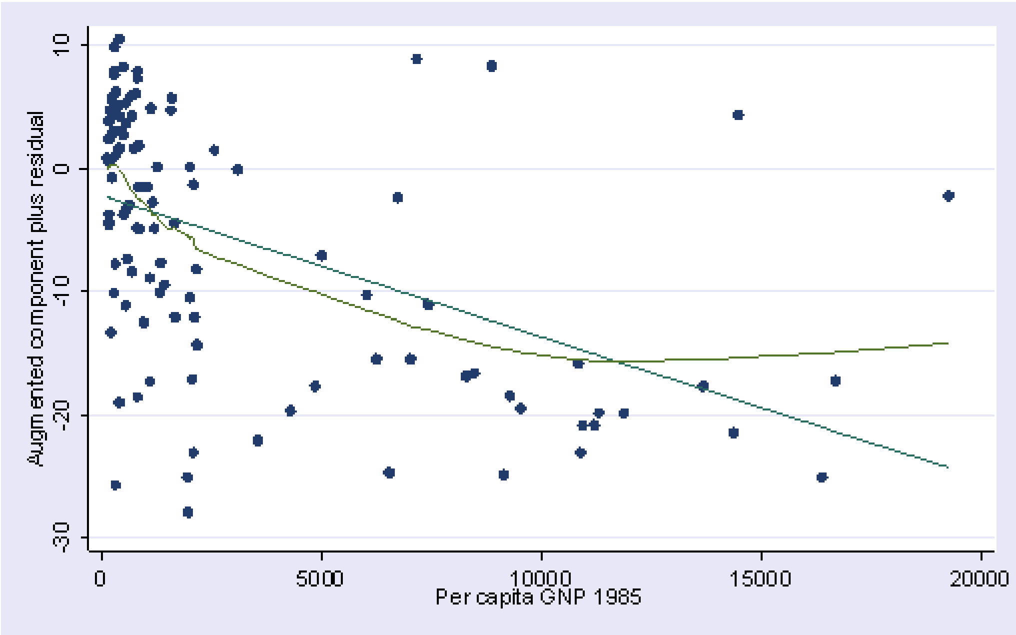
Let's build a model that predicts birth rate (**birth**), from per capita gross national product (**gnpcap**), and urban population (**urban**). If this were a complete regression analysis, we would start with examining the variables, but for the purpose of illustrating nonlinearity, we will jump directly to the regression.

```
regress birth gnpcap urban
  Source |       SS       df       MS               Number of obs =
108
---------+------------------------------            F(  2,   105) =
64.22
   Model |   10796.488     2  5398.24399            Prob > F      =
0.0000
Residual |   8825.5861   105   84.053201            R-squared     =
0.5502
---------+------------------------------            Adj R-squared =
0.5417
```

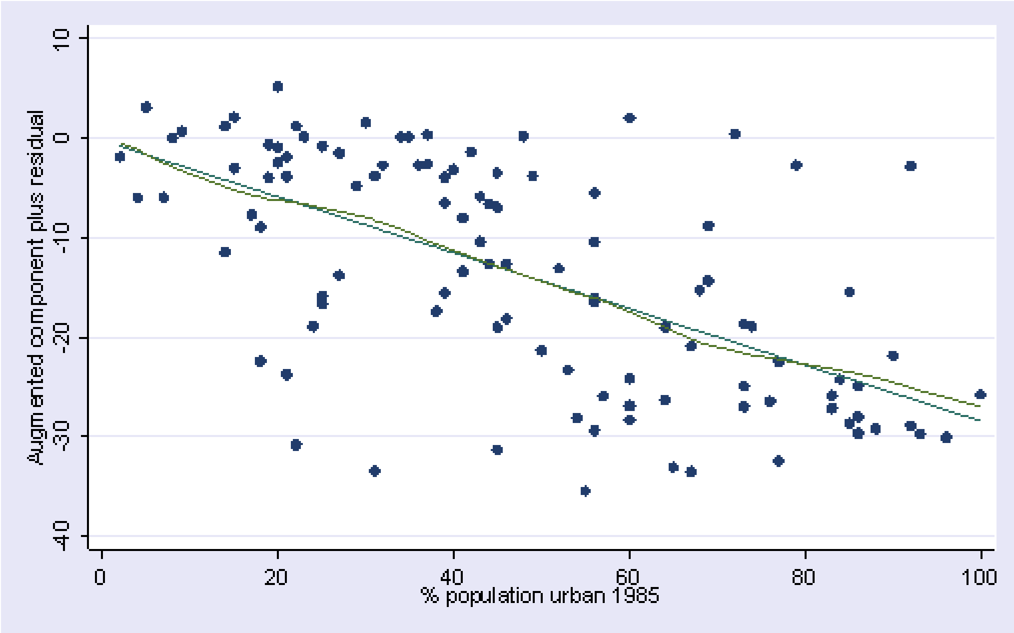
```
   Total |  19622.0741   107   183.38387               Root MSE      =
9.1681

------------------------------------------------------------------------
-------
   birth |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  gnpcap |   -.000842   .0002637     -3.193   0.002      -.0013649   -
.0003191
   urban |  -.2823184   .0462191     -6.108   0.000      -.3739624   -
.1906744
   _cons |   48.85603   1.986909     24.589   0.000       44.91635
52.7957
------------------------------------------------------------------------
-------
```

Now, let's do the **acprplot** on our predictors.

```
acprplot gnpcap, lowess
```

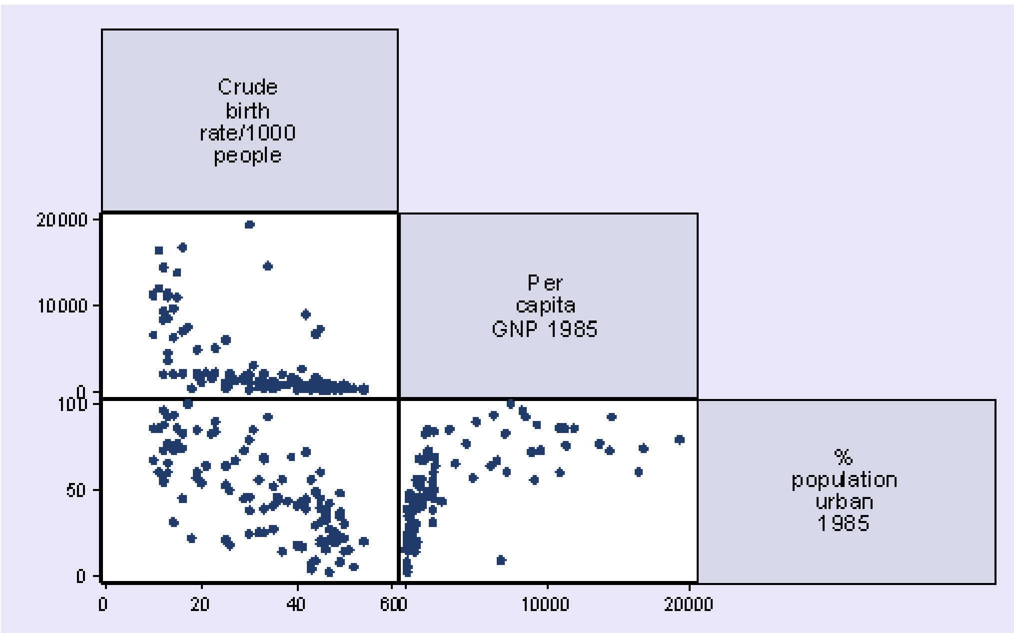



```
acprplot urban, lowess
```



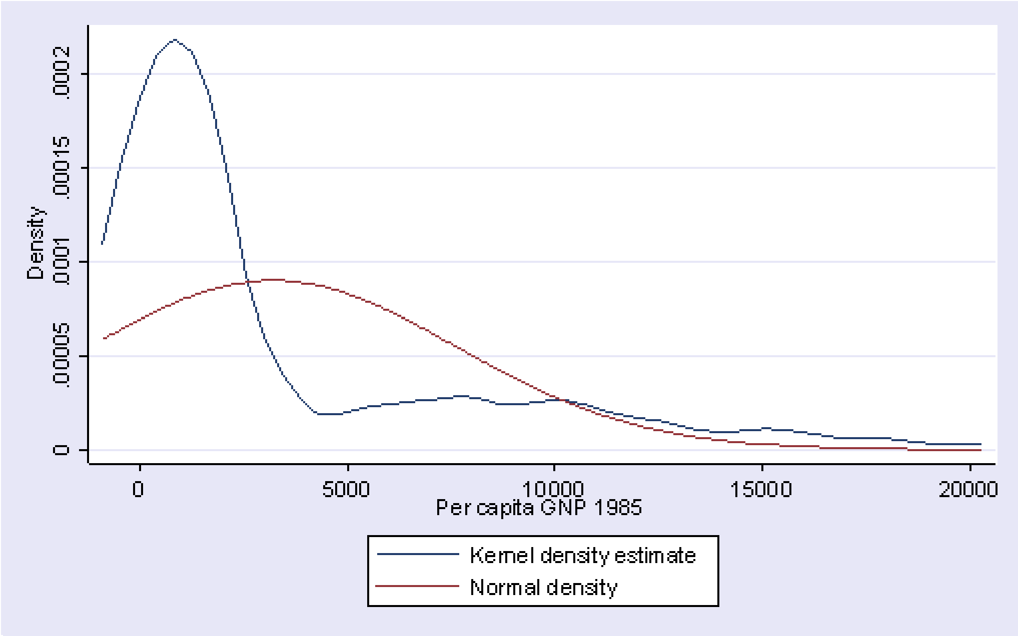
The **acprplot** plot for **gnpcap** shows clear deviation from linearity and the one for **urban** does not show nearly as much deviation from linearity. Now, let's look at these variables more closely.

```
graph matrix birth gnpcap urban, half
```



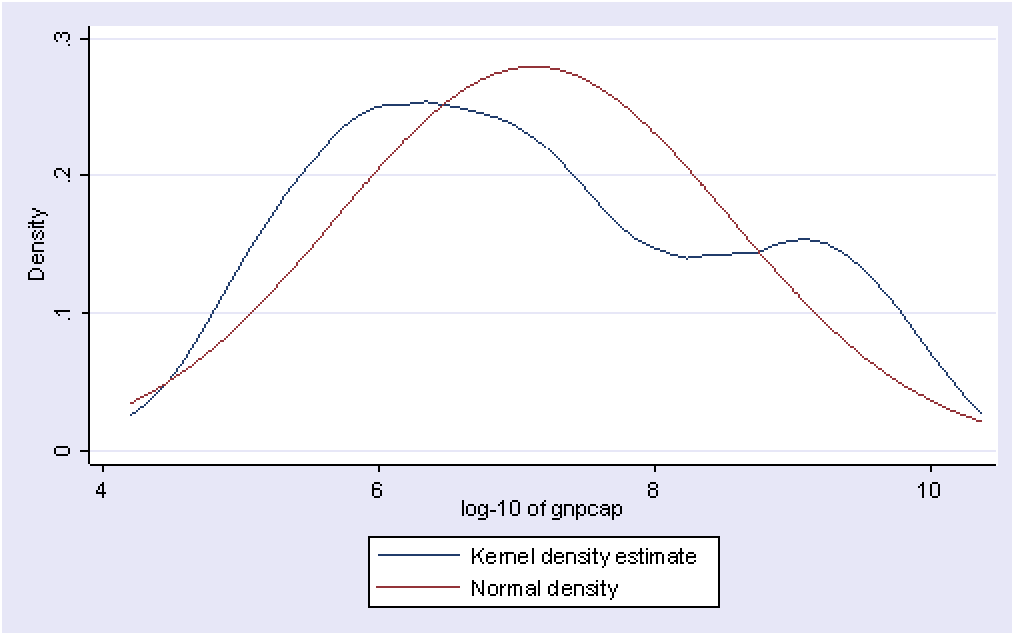
We see that the relation between birth rate and per capita gross national product is clearly nonlinear and the relation between birth rate and urban population is not too far off from being linear. So let's focus on variable **gnpcap**. First let's look at the distribution of **gnpcap**. We suspect that **gnpcap** may be very skewed. This may affect the appearance of the **acprplot**.

```
kdensity gnpcap, normal
```



Indeed, it is very skewed. This suggests to us that some transformation of the variable may be necessary. One of the commonly used transformations is log transformation. Let's try it here.

```
generate lggnp=log(gnpcap)
label variable lggnp "log-10 of gnpcap"
kdensity lggnp, normal
```

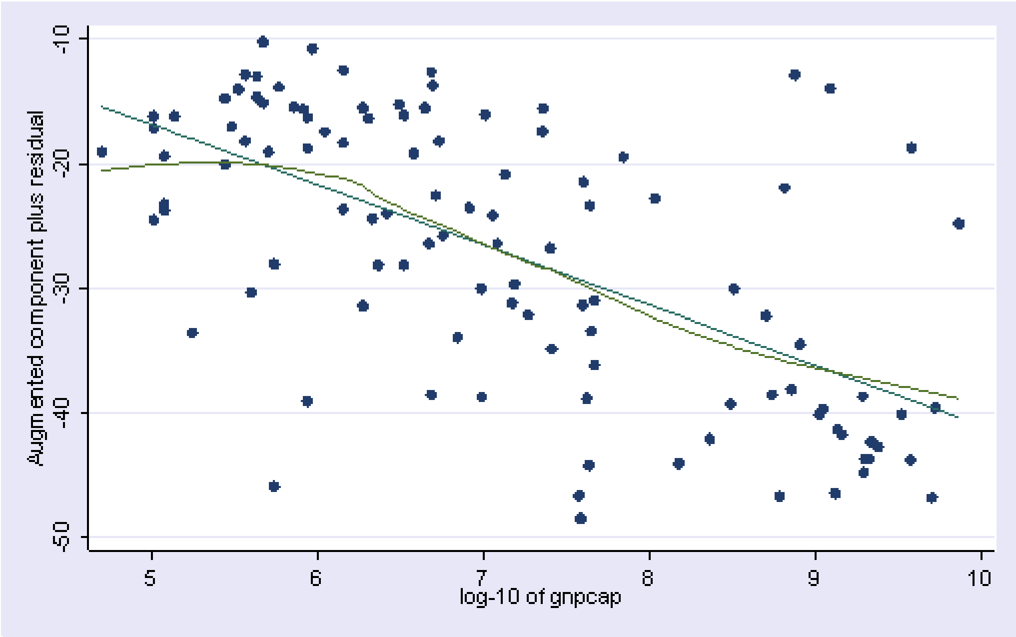


The transformation does seem to help correct the skewness greatly. Next, let's do the regression again replacing **gnpcap** by **lggnp**.

regress birth lggnp urban

```
  Source |       SS       df       MS                  Number of obs =
108
---------+------------------------------               F(  2,   105) =
76.20
   Model |  11618.0395     2  5809.01974               Prob > F      =
0.0000
Residual |   8004.0346   105  76.2289009               R-squared     =
0.5921
---------+------------------------------               Adj R-squared =
0.5843
   Total |  19622.0741   107   183.38387               Root MSE      =
8.7309

------------------------------------------------------------------------
-------
   birth |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
   lggnp |  -4.877688   1.039477     -4.692   0.000       -6.93878   -
2.816596
   urban |   -.156254   .0579632     -2.696   0.008      -.2711843   -
.0413237
   _cons |   74.87778   5.439654     13.765   0.000       64.09196
85.66361
------------------------------------------------------------------------
-------
```

```
acprplot lggnp, lowess
```



The plot above shows less deviation from nonlinearity than before, though the problem of nonlinearity has not been completely solved yet.

2.6 Model Specification

A model specification error can occur when one or more relevant variables are omitted from the model or one or more irrelevant variables are included in the model. If relevant variables are omitted from the model, the common variance they share with included variables may be wrongly attributed to those variables, and the error term is inflated. On the other hand, if irrelevant variables are included in the model, the common variance they share with included variables may be wrongly attributed to them. Model specification errors can substantially affect the estimate of regression coefficients.

Consider the model below. This regression suggests that as class size increases the academic performance increases. Before we publish results saying that increased class size is associated with higher academic performance, let's check the model specification.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2

regress api00  acs_k3

      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  1,   396) =
11.93
       Model |  234353.831     1  234353.831           Prob > F      =
0.0006
```

```
    Residual |  7779853.31   396  19646.0942           R-squared     =
0.0292
-------------+------------------------------           Adj R-squared =
0.0268
       Total |  8014207.14   397  20186.9197           Root MSE      =
140.16

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
      acs_k3 |   17.75148   5.139688     3.45   0.001     7.646998
27.85597
       _cons |   308.3372   98.73085     3.12   0.002      114.235
502.4393
----------------------------------------------------------------------
-------
```

There are a couple of methods to detect specification errors. The **linktest** command performs a model specification link test for single-equation models. **linktest** is based on the idea that if a regression is properly specified, one should not be able to find any additional independent variables that are significant except by chance. **linktest** creates two new variables, the variable of prediction, **_hat**, and the variable of squared prediction, **_hatsq**. The model is then refit using these two variables as predictors. **_hat** should be significant since it is the predicted value. On the other hand, **_hatsq** shouldn't, because if our model is specified correctly, the squared predictions should not have much explanatory power. That is we wouldn't expect **_hatsq** to be a significant predictor if our model is specified correctly. So we will be looking at the p-value for **_hatsq**.

```
linktest

      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  2,   395) =
7.09
       Model |  277705.911     2  138852.955           Prob > F      =
0.0009
    Residual |  7736501.23   395  19586.0791           R-squared     =
0.0347
-------------+------------------------------           Adj R-squared =
0.0298
       Total |  8014207.14   397  20186.9197           Root MSE      =
139.95

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
        _hat |  -11.05006   8.104639    -1.36   0.174    -26.98368
4.883562
```

```
      _hatsq |   .0093318   .0062724     1.49   0.138    -.0029996
.0216631
       _cons |    3884.48   2617.695     1.48   0.139    -1261.877
9030.837
------------------------------------------------------------------------------
```

From the above **linktest**, the test of **_hatsq** is not significant. This is to say that **linktest** has failed to reject the assumption that the model is specified correctly. Therefore, it seems to us that we don't have a specification error. But now, let's look at another test before we jump to the conclusion.

The **ovtest** command performs another test of regression model specification. It performs a regression specification error test (RESET) for omitted variables. The idea behind **ovtest** is very similar to **linktest**. It also creates new variables based on the predictors and refits the model using those new variables to see if any of them would be significant. Let's try **ovtest** on our model.

ovtest

```
Ramsey RESET test using powers of the fitted values of api00
       Ho:  model has no omitted variables
                 F(3, 393) =      4.13
                  Prob > F =      0.0067
```

The **ovtest** command indicates that there are omitted variables. So we have tried both the **linktest** and **ovtest**, and one of them (**ovtest**) tells us that we have a specification error. We therefore have to reconsider our model.

Let's try adding the variable **full** to the model. Now, both the **linktest** and **ovtest** are significant, indicating we have a specification error.

regress api00 acs_k3 full

```
      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  2,   395) =
101.19
       Model |  2715101.89     2  1357550.95           Prob > F      =
0.0000
    Residual |  5299105.24   395  13415.4563           R-squared     =
0.3388
-------------+------------------------------           Adj R-squared =
0.3354
       Total |  8014207.14   397  20186.9197           Root MSE      =
115.83

------------------------------------------------------------------------------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
```

```
-------------+----------------------------------------------------------------
-------
     acs_k3 |   8.355681   4.303023     1.94   0.053    -.1040088
16.81537
       full |   5.389788   .3963539    13.60   0.000     4.610561
6.169015
      _cons |   32.21346   84.07525     0.38   0.702    -133.0775
197.5044
------------------------------------------------------------------------------
-------
```

linktest

```
      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  2,   395) =
108.32
       Model |  2838564.40     2  1419282.20           Prob > F      =
0.0000
    Residual |  5175642.74   395   13102.893           R-squared     =
0.3542
-------------+------------------------------           Adj R-squared =
0.3509
       Total |  8014207.14   397  20186.9197           Root MSE      =
114.47

------------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-------
        _hat |  -1.868895   .9371889    -1.99   0.047    -3.711397   -
.0263936
      _hatsq |   .0023436   .0007635     3.07   0.002     .0008426
.0038447
       _cons |   858.8726   283.4594     3.03   0.003     301.5948
1416.15
------------------------------------------------------------------------------
-------
```

ovtest

```
Ramsey RESET test using powers of the fitted values of api00
       Ho:  model has no omitted variables
                 F(3, 392) =      4.09
                  Prob > F =      0.0071
```

Let's try adding one more variable, **meals**, to the above model.

regress api00 acs_k3 full meals

```
      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  3,   394) =
615.55
```

```
       Model |  6604966.18     3  2201655.39           Prob > F      =
0.0000
    Residual |  1409240.96   394   3576.7537           R-squared     =
0.8242
-------------+------------------------------           Adj R-squared =
0.8228
       Total |  8014207.14   397  20186.9197           Root MSE      =
59.806

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      acs_k3 |  -.7170622   2.238821    -0.32   0.749    -5.118592
3.684468
        full |   1.327138   .2388739     5.56   0.000      .857511
1.796765
       meals |  -3.686265   .1117799   -32.98   0.000    -3.906024   -
3.466505
       _cons |   771.6581   48.86071    15.79   0.000     675.5978
867.7184
------------------------------------------------------------------------
-------
```

linktest

```
      Source |       SS       df       MS              Number of obs =
398
-------------+------------------------------           F(  2,   395) =
931.68
       Model |  6612479.76     2  3306239.88           Prob > F      =
0.0000
    Residual |  1401727.38   395  3548.67691           R-squared     =
0.8251
-------------+------------------------------           Adj R-squared =
0.8242
       Total |  8014207.14   397  20186.9197           Root MSE      =
59.571

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
        _hat |    1.42433   .2925374     4.87   0.000      .849205
1.999455
      _hatsq |  -.0003172    .000218    -1.46   0.146    -.0007458
.0001114
       _cons |  -136.5102   95.05904    -1.44   0.152    -323.3951
50.3747
------------------------------------------------------------------------
-------
```

ovtest

```
Ramsey RESET test using powers of the fitted values of api00
       Ho:  model has no omitted variables
                 F(3, 391) =      2.56
                  Prob > F =      0.0545
```

The **linktest** is once again non-significant while the p-value for **ovtest** is slightly greater than .05. Note that after including **meals** and **full**, the coefficient for class size is no longer significant. While **acs_k3** does have a positive relationship with **api00** when no other variables are in the model, when we include, and hence control for, other important variables, **acs_k3** is no longer significantly related to **api00** and its relationship to **api00** is no longer positive**.**

linktest and **ovtest** are tools available in Stata for checking specification errors, though **linktest** can actually do more than check omitted variables as we used here, e.g., checking the correctness of link function specification. For more details on those tests, please refer to Stata manual.

2.7 Issues of Independence

The statement of this assumption that the errors associated with one observation are not correlated with the errors of any other observation cover several different situations. Consider the case of collecting data from students in eight different elementary schools. It is likely that the students within each school will tend to be more like one another than students from different schools, that is, their errors are not independent. We will deal with this type of situation in Chapter 4 when we demonstrate the **regress** command with **cluster** option.

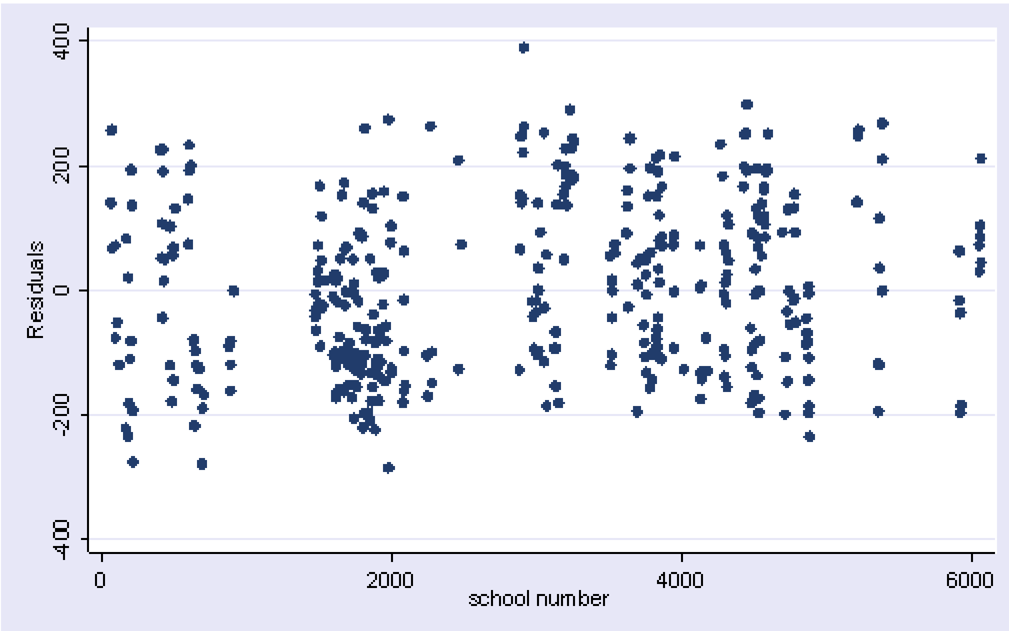
Another way in which the assumption of independence can be broken is when data are collected on the same variables over time. Let's say that we collect truancy data every semester for 12 years. In this situation it is likely that the errors for observation between adjacent semesters will be more highly correlated than for observations more separated in time. This is known as autocorrelation. When you have data that can be considered to be time-series you should use the **dwstat** command that performs a Durbin-Watson test for correlated residuals.

We don't have any time-series data, so we will use the **elemapi2** dataset and pretend that **snum** indicates the time at which the data were collected. We will also need to use the **tsset** command to let Stata know which variable is the time variable.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2
tsset snum
        time variable:  snum, 58 to 6072, but with gaps

regress api00 enroll

( output omitted )

dwstat

Number of gaps in sample:  311
Durbin-Watson d-statistic(  2,   400) =  .2892712
```

The Durbin-Watson statistic has a range from 0 to 4 with a midpoint of 2. The observed value in our example is very small, close to zero, which is not surprising since our data are not truly time-series. A simple visual check would be to plot the residuals versus the time variable.

```
. predict r, resid

scatter r snum
```



2.8 Summary

In this chapter, we have used a number of tools in Stata for determining whether our data meets the regression assumptions. Below, we list the major commands we demonstrated organized according to the assumption the command was shown to test.

- **Detecting Unusual and Influential Data**
  - **predict** -- used to create predicted values, residuals, and measures of influence.
  - **rvpplot** --- graphs a residual-versus-predictor plot.
  - **rvfplot --** graphs residual-versus-fitted plot.
  - **lvr2plot** -- graphs a leverage-versus-squared-residual plot.
  - **dfbeta --** calculates DFBETAs for all the independent variables in the linear model.
  - **avplot** -- graphs an added-variable plot, a.k.a. partial regression plot.
- **Tests for Normality of Residuals**
  - **kdensity** -- produces kernel density plot with normal distribution overlayed.
  - **pnorm** -- graphs a standardized normal probability (P-P) plot.
  - **qnorm** --- plots the quantiles of varname against the quantiles of a normal distribution.
  - **iqr** -- resistant normality check and outlier identification.
  - **swilk** -- performs the Shapiro-Wilk W test for normality.
- **Tests for Heteroscedasticity**
  - **rvfplot --** graphs residual-versus-fitted plot.

- o **hettest** -- performs Cook and Weisberg test for heteroscedasticity.
- o **whitetst** -- computes the White general test for Heteroscedasticity.

- **Tests for Multicollinearity**
  - o **vif** -- calculates the variance inflation factor for the independent variables in the linear model.
  - o **collin** -- calculates the variance inflation factor and other multicollinearity diagnostics
- **Tests for Non-Linearity**
  - o **acprplot** -- graphs an augmented component-plus-residual plot.
  - o **cprplot** --- graphs component-plus-residual plot, a.k.a. residual plot.
- **Tests for Model Specification**
  - o **linktest** -- performs a link test for model specification.
  - o **ovtest** -- performs regression specification error test (RESET) for omitted variables.

See the Stata Topics: Regression page for more information and resources on regression diagnostics in Stata.

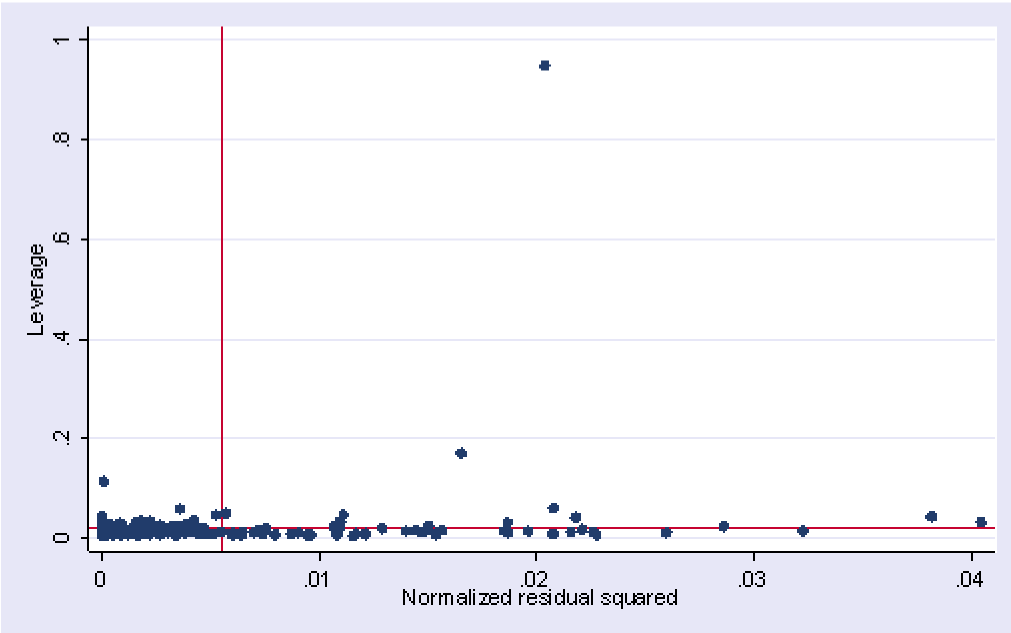
2.9 Self Assessment

1. The following data set consists of measured weight, measured height, reported weight and reported height of some 200 people. You can get it from within Stata by typing **use http://www.ats.ucla.edu/stat/stata/webbooks/reg/davis** We tried to build a model to predict measured weight by reported weight, reported height and measured height. We did an lvr2plot after the regression and here is what we have. Explain what you see in the graph and try to use other STATA commands to identify the problematic observation(s). What do you think the problem is and what is your solution?

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/davis
. regress  measwt measht reptwt reptht
  Source |       SS       df       MS                  Number of obs =
181
---------+------------------------------               F(  3,   177) =
1640.88
   Model |  40891.9594     3  13630.6531               Prob > F      =
0.0000
Residual |   1470.3279   177  8.30693727               R-squared     =
0.9653
---------+------------------------------               Adj R-squared =
0.9647
   Total |  42362.2873   180  235.346041               Root MSE      =
2.8822

-----------------------------------------------------------------------------
-
  measwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
  measht |  -.9607757   .0260189    -36.926   0.000      -1.012123   -
.9094285
  reptwt |    1.01917   .0240778     42.328   0.000        .971654
1.066687
```

```
  reptht |   .8184156   .0419658     19.502   0.000       .7355979
.9012334
   _cons |    24.8138   4.888302      5.076   0.000       15.16695
34.46065
------------------------------------------------------------------------------
-
```

lvr2plot



2. Using the data from the last exercise, what measure would you use if you want to know how much change an observation would make on a coefficient for a predictor? For example, show how much change would it be for the coefficient of predictor **reptht** if we omit observation 12 from our regression analysis? What are the other measures that you would use to assess the influence of an observation on regression? What are the cut-off values for them?

3. The following data file is called **bbwt.dta** and it is from Weisberg's Applied Regression Analysis. You can obtain it from within Stata by typing **use http://www.ats.ucla.edu/stat/stata/webbooks/reg/bbwt** It consists of the body weights and brain weights of some 60 animals. We want to predict the brain weight by body weight, that is, a simple linear regression of brain weight against body weight. Show what you have to do to verify the linearity assumption. If you think that it violates the linearity assumption, show some possible remedies that you would consider.

use http://www.ats.ucla.edu/stat/stata/webbooks/reg/bbwt, clear
regress brainwt bodywt

```
  Source |       SS       df       MS                  Number of obs =
62
---------+------------------------------               F(  1,    60) =
411.12
```

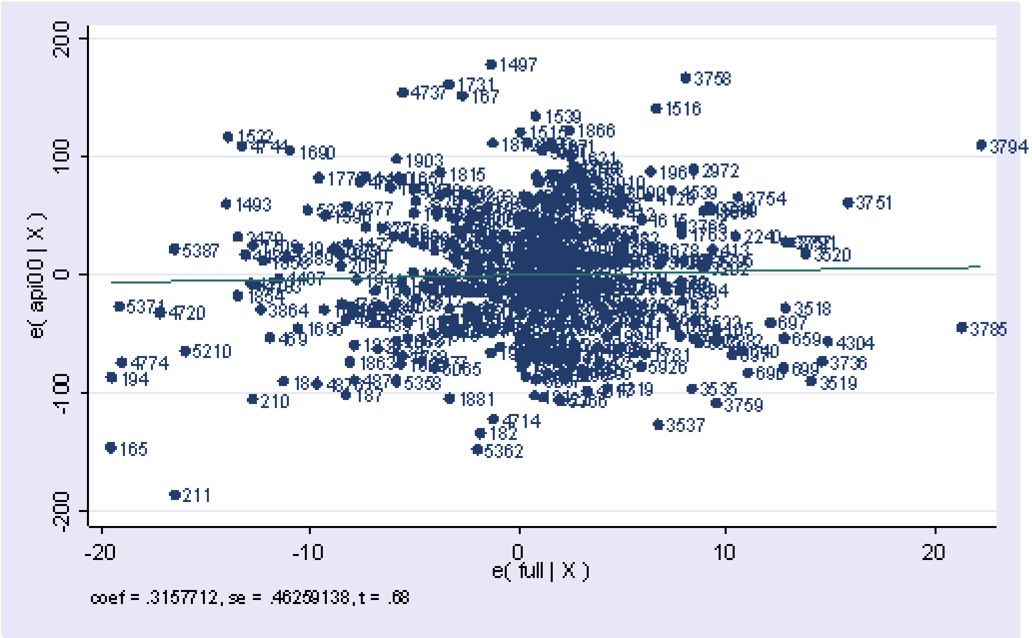
```
   Model |  46067326.8     1  46067326.8               Prob > F      =
0.0000
Residual |  6723217.18    60   112053.62               R-squared     =
0.8726
---------+------------------------------               Adj R-squared =
0.8705
   Total |  52790543.9    61  865418.753               Root MSE      =
334.74

------------------------------------------------------------------------------
-
 brainwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
  bodywt |   .9664599   .0476651     20.276   0.000       .8711155
1.061804
   _cons |   91.00865   43.55574      2.089   0.041       3.884201
178.1331
------------------------------------------------------------------------------
-
```

4. We did a regression analysis using the data file **elemapi2** in chapter 2. Continuing with the analysis we did, we did an avplot here. Explain what an avplot is and what type of information you would get from the plot. If variable **full** were put in the model, would it be a significant predictor?

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
regress api00 meals ell emer
  Source |       SS       df       MS                  Number of obs =
400
---------+------------------------------               F(  3,   396) =
673.00
   Model |  6749782.75     3  2249927.58               Prob > F      =
0.0000
Residual |  1323889.25   396  3343.15467               R-squared     =
0.8360
---------+------------------------------               Adj R-squared =
0.8348
   Total |  8073672.00   399  20234.7669               Root MSE      =
57.82

------------------------------------------------------------------------------
-
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
   meals |  -3.159189   .1497371    -21.098   0.000      -3.453568   -
2.864809
     ell |  -.9098732   .1846442     -4.928   0.000      -1.272878   -
.5468678
    emer |  -1.573496    .293112     -5.368   0.000      -2.149746   -
.9972456
```

```
   _cons |   886.7033    6.25976    141.651   0.000      874.3967
899.0098
------------------------------------------------------------------------------
-
```

avplot full, mlabel(snum)



5. The data set **wage.dta** is from a national sample of 6000 households with a male head earning less than $15,000 annually in 1966. You can get this data file by typing **use http://www.ats.ucla.edu/stat/stata/webbooks/reg/wage** from within Stata**.** The data were classified into 39 demographic groups for analysis. We tried to predict the average hours worked by average age of respondent and average yearly non-earned income.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/wage, clear
. regress HRS AGE NEIN
  Source |       SS       df       MS                  Number of obs =
39
---------+------------------------------               F(  2,    36) =
39.72
   Model |  107205.109     2  53602.5543               Prob > F      =
0.0000
Residual |  48578.1222    36  1349.39228               R-squared     =
0.6882
---------+------------------------------               Adj R-squared =
0.6708
   Total |  155783.231    38   4099.5587               Root MSE      =
36.734

------------------------------------------------------------------------------
-
     HRS |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
```

```
     AGE |  -8.281632   1.603736     -5.164   0.000      -11.53416   -
5.029104
    NEIN |   .4289202   .0484882      8.846   0.000       .3305816
.5272588
   _cons |    2321.03   57.55038     40.330   0.000       2204.312
2437.748
-----------------------------------------------------------------------------
-
```

Both predictors are significant. Now if we add ASSET to our predictors list, neither NEIN nor ASSET is significant.

```
regress HRS AGE NEIN ASSET
  Source |       SS       df       MS                  Number of obs =
39
---------+------------------------------               F(  3,    35) =
25.83
   Model |   107317.64     3  35772.5467               Prob > F      =
0.0000
Residual |  48465.5908    35  1384.73117               R-squared     =
0.6889
---------+------------------------------               Adj R-squared =
0.6622
   Total |  155783.231    38   4099.5587               Root MSE      =
37.212

-----------------------------------------------------------------------------
-
     HRS |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
     AGE |  -8.007181    1.88844     -4.240   0.000      -11.84092   -
4.173443
    NEIN |   .3338277    .337171      0.990   0.329      -.3506658
1.018321
   ASSET |   .0044232    .015516      0.285   0.777       -.027076
.0359223
   _cons |   2314.054   63.22636     36.600   0.000       2185.698
2442.411
-----------------------------------------------------------------------------
-
```

Can you explain why?

6. Continue to use the previous data set. This time we want to predict the average hourly wage by average percent of white respondents. Carry out the regression analysis and list the STATA commands that you can use to check for heteroscedasticity. Explain the result of your test(s).

Now we want to build another model to predict the average percent of white respondents by the average hours worked. Repeat the analysis you performed on the previous regression model. Explain your results.

7. We have a data set that consists of volume, diameter and height of some objects. Someone did a regression of volume on diameter and height.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/tree, clear
regress vol dia height
  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  2,    28) =
254.97
   Model |  7684.16254     2  3842.08127               Prob > F      =
0.0000
Residual |  421.921306    28  15.0686181               R-squared     =
0.9480
---------+------------------------------               Adj R-squared =
0.9442
   Total |  8106.08385    30  270.202795               Root MSE      =
3.8818

-----------------------------------------------------------------------------
-
     vol |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
     dia |   4.708161   .2642646     17.816   0.000       4.166839
5.249482
  height |   .3392513   .1301512      2.607   0.014       .0726487
.6058538
   _cons |  -57.98766   8.638225     -6.713   0.000      -75.68226   -
40.29306
-----------------------------------------------------------------------------
-
```

Explain what tests you can use to detect model specification errors and if there is any, your solution to correct it.

Click here for our answers to these self assessment questions.

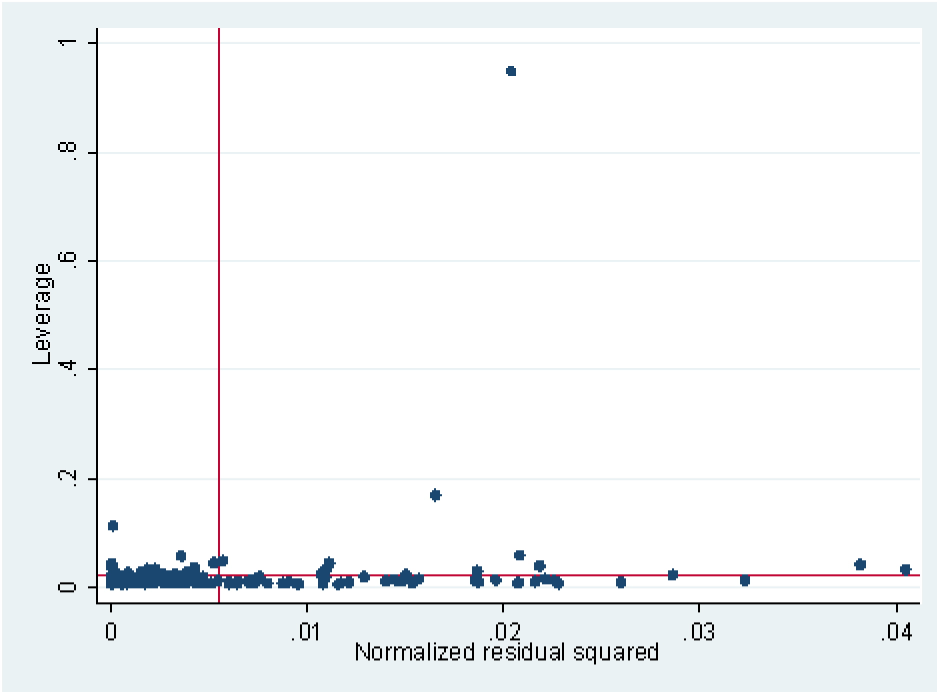
2.10 For more information

- Stata Manuals
  - **[R] regress**
  - **[R] regression diagnostics**

Chapter 2
Self Assessment

1. The following data set consists of measured weight, measured height, reported weight and reported height of some 200 people. You can get it from within Stata by typing **use http://www.ats.ucla.edu/stat/stata/webbooks/reg/davis** We tried to build a model to predict measured weight by reported weight, reported height and measured height. We did an lvr2plot after the regression and here is what we have. Explain what you see in the graph and try to use other STATA commands to identify the problematic observation(s). What do you think the problem is and what is your solution?

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/davis
regress  measwt measht reptwt reptht
  Source |       SS       df       MS                  Number of obs =
181
---------+------------------------------               F(  3,   177) =
1640.88
   Model |  40891.9594     3  13630.6531               Prob > F      =
0.0000
Residual |   1470.3279   177  8.30693727               R-squared     =
0.9653
---------+------------------------------               Adj R-squared =
0.9647
   Total |  42362.2873   180  235.346041               Root MSE      =
2.8822

------------------------------------------------------------------------------
-
  measwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
  measht |  -.9607757   .0260189    -36.926   0.000      -1.012123   -
.9094285
  reptwt |    1.01917   .0240778     42.328   0.000        .971654
1.066687
  reptht |   .8184156   .0419658     19.502   0.000       .7355979
.9012334
   _cons |    24.8138   4.888302      5.076   0.000       15.16695
34.46065
------------------------------------------------------------------------------
-
lvr2plot
```



2. Using the data from the last exercise, what measure would you use if you want to know how much change an observation would make on a coefficient for a predictor? For example, show how much change would it be for the coefficient of predictor **reptht** if we omit observation 12 from our regression analysis? What are the other measures that you would use to assess the influence of an observation on regression? What are the cut-off values for them?

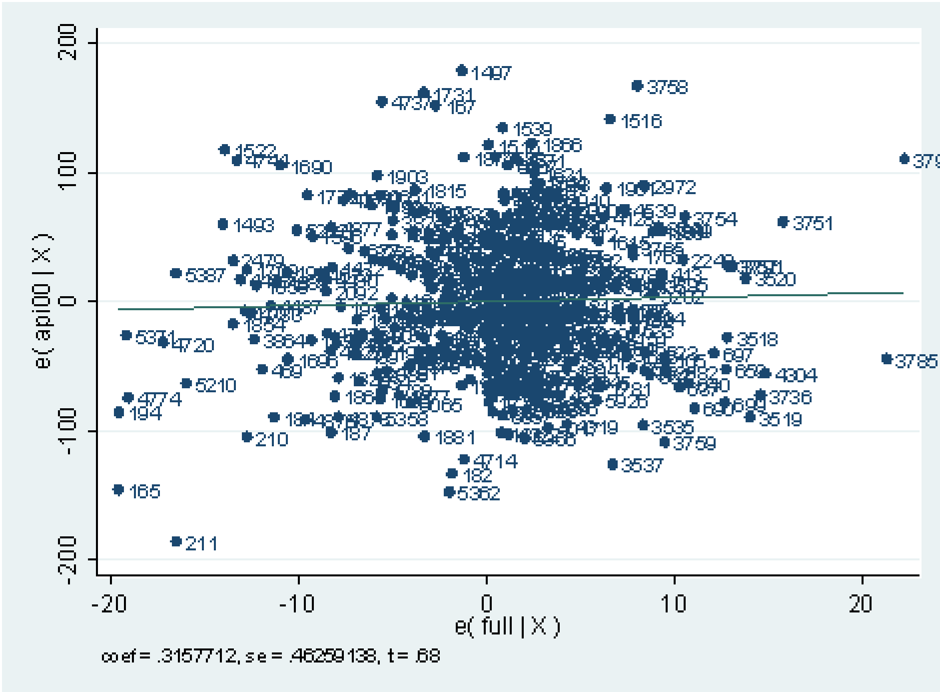
3. The following data file is called **bbwt.dta** and it is from Weisberg's Applied Regression Analysis. You can obtain it from within Stata by typing **use http://www.ats.ucla.edu/stat/stata/webbooks/reg/bbwt** It consists of the body weights and brain weights of some 60 animals. We want to predict the brain weight by body weight, that is, a simple linear regression of brain weight against body weight. Show what you have to do to verify the linearity assumption. If you think that it violates the linearity assumption, show some possible remedies that you would consider.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/bbwt, clear
regress brainwt bodywt
  Source |       SS       df       MS                  Number of obs =
62
---------+------------------------------               F(  1,    60) =
411.12
   Model |  46067326.8     1  46067326.8               Prob > F      =
0.0000
Residual |  6723217.18    60  112053.62                R-squared     =
0.8726
---------+------------------------------               Adj R-squared =
0.8705
   Total |  52790543.9    61  865418.753               Root MSE      =
334.74
```

```
------------------------------------------------------------------------------
-
 brainwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
  bodywt |   .9664599   .0476651     20.276   0.000       .8711155
1.061804
   _cons |   91.00865   43.55574      2.089   0.041       3.884201
178.1331
------------------------------------------------------------------------------
-
```

4. We did a regression analysis using the data file **elemapi2** in chapter 2. Continuing with the analysis we did, we did an avplot here. Explain what an avplot is and what type of information you would get from the plot. If variable **full** were put in the model, would it be a significant predictor?

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
regress api00 meals ell emer
  Source |       SS       df       MS                  Number of obs =
400
---------+------------------------------               F(  3,   396) =
673.00
   Model |  6749782.75     3  2249927.58               Prob > F      =
0.0000
Residual |  1323889.25   396  3343.15467               R-squared     =
0.8360
---------+------------------------------               Adj R-squared =
0.8348
   Total |  8073672.00   399  20234.7669               Root MSE      =
57.82

------------------------------------------------------------------------------
-
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
   meals |  -3.159189   .1497371    -21.098   0.000      -3.453568   -
2.864809
     ell |  -.9098732   .1846442     -4.928   0.000      -1.272878   -
.5468678
    emer |  -1.573496    .293112     -5.368   0.000      -2.149746   -
.9972456
   _cons |   886.7033    6.25976    141.651   0.000       874.3967
899.0098
------------------------------------------------------------------------------
-
avplot full, mlabel(snum)
```



5. The data set **wage.dta** is from a national sample of 6000 households with a male head earning less than $15,000 annually in 1966. You can get this data file by typing **use http://www.ats.ucla.edu/stat/stata/webbooks/reg/wage** from within Stata**.** The data were classified into 39 demographic groups for analysis. We tried to predict the average hours worked by average age of respondent and average yearly non-earned income.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/wage, clear
regress HRS AGE NEIN
  Source |       SS       df       MS               Number of obs =
39
---------+------------------------------               F(  2,    36) =
39.72
   Model |  107205.109     2  53602.5543               Prob > F      =
0.0000
Residual |  48578.1222    36  1349.39228               R-squared     =
0.6882
---------+------------------------------               Adj R-squared =
0.6708
   Total |  155783.231    38   4099.5587               Root MSE      =
36.734

------------------------------------------------------------------------------
-
     HRS |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
     AGE |  -8.281632   1.603736     -5.164   0.000      -11.53416   -
5.029104
```

```
    NEIN |   .4289202   .0484882      8.846   0.000       .3305816
.5272588
   _cons |    2321.03   57.55038     40.330   0.000       2204.312
2437.748
-----------------------------------------------------------------------------
-
```

Both predictors are significant. Now if we add ASSET to our predictors list, neither NEIN nor ASSET is significant.

```
regress HRS AGE NEIN ASSET
  Source |       SS       df       MS                  Number of obs =
39
---------+------------------------------               F(  3,    35) =
25.83
   Model |  107317.64     3  35772.5467               Prob > F      =
0.0000
Residual |  48465.5908    35  1384.73117               R-squared     =
0.6889
---------+------------------------------               Adj R-squared =
0.6622
   Total |  155783.231    38   4099.5587               Root MSE      =
37.212

-----------------------------------------------------------------------------
-
     HRS |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
     AGE |  -8.007181    1.88844     -4.240   0.000      -11.84092   -
4.173443
    NEIN |   .3338277    .337171      0.990   0.329      -.3506658
1.018321
   ASSET |   .0044232    .015516      0.285   0.777       -.027076
.0359223
   _cons |   2314.054   63.22636     36.600   0.000       2185.698
2442.411
-----------------------------------------------------------------------------
-
```

Can you explain why?

6. Continue to use the previous data set. This time we want to predict the average hourly wage by average percent of white respondents. Carry out the regression analysis and list the STATA commands that you can use to check for heteroscedasticity. Explain the result of your test(s).

Now we want build another model to predict the average percent of white respondents by the average hours worked. Repeat the analysis you performed on the previous regression model. Explain your results.

7. We have a data set that consists of volume, diameter and height of some objects. Someone did a regression of volume on diameter and height.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/tree, clear
regress vol dia height
  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  2,    28) =
254.97
   Model |  7684.16254     2  3842.08127               Prob > F      =
0.0000
Residual |  421.921306    28  15.0686181               R-squared     =
0.9480
---------+------------------------------               Adj R-squared =
0.9442
   Total |  8106.08385    30  270.202795               Root MSE      =
3.8818

-----------------------------------------------------------------------------
-
     vol |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
     dia |   4.708161   .2642646     17.816   0.000       4.166839
5.249482
  height |   .3392513   .1301512      2.607   0.014       .0726487
.6058538
   _cons |  -57.98766   8.638225     -6.713   0.000      -75.68226   -
40.29306
-----------------------------------------------------------------------------
-
```

Explain what tests you can use to detect model specification errors and if there is any, your solution to correct it.

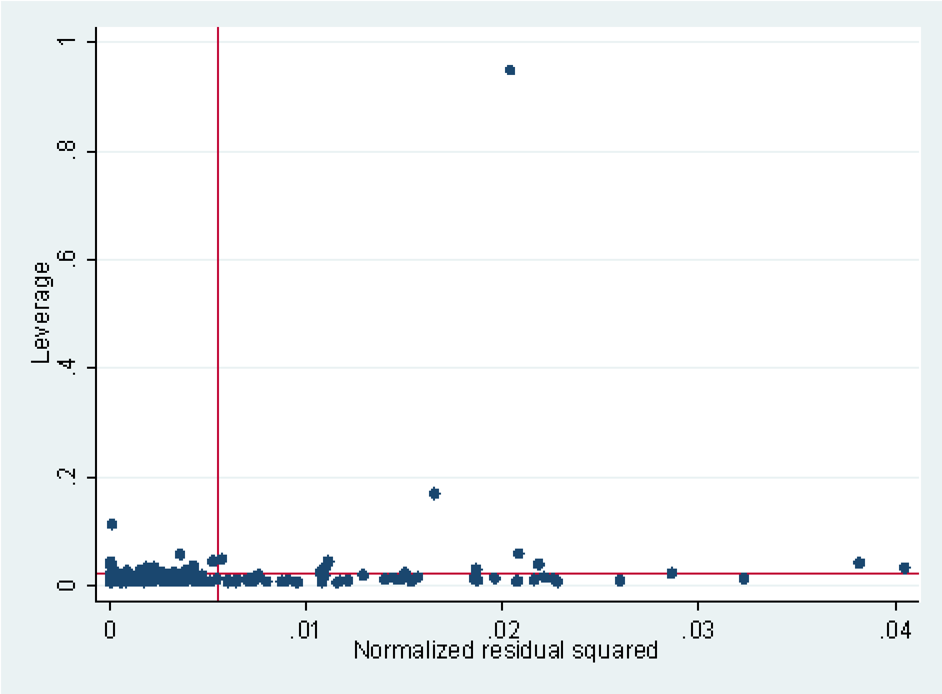
Chapter 2
Self Assessment Answers

1. The following data set consists of measured weight, measured height, reported weight and reported height of some 200 people. We tried to build a model to predict measured weight by reported weight, reported height and measured height. We did an lvr2plot after the regression and here is what we have. Explain what you see in the graph and try to use other STATA commands to identify the problematic observation(s). What do you think the problem is and what is your solution?

```
use davis, clear
regress  measwt measht reptwt reptht
```

```
Source |       SS       df       MS                         Number of obs =
181
---------+------------------------------                    F(  3,   177) =
1640.88
   Model |  40891.9594     3  13630.6531                    Prob > F      =
0.0000
Residual |   1470.3279   177  8.30693727                    R-squared     =
0.9653
---------+------------------------------                    Adj R-squared =
0.9647
   Total |  42362.2873   180  235.346041                    Root MSE      =
2.8822

------------------------------------------------------------------------------
-
  measwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
  measht |  -.9607757   .0260189    -36.926   0.000      -1.012123   -
.9094285
  reptwt |    1.01917   .0240778     42.328   0.000        .971654
1.066687
  reptht |   .8184156   .0419658     19.502   0.000       .7355979
.9012334
   _cons |    24.8138   4.888302      5.076   0.000       15.16695
34.46065
------------------------------------------------------------------------------
-
```

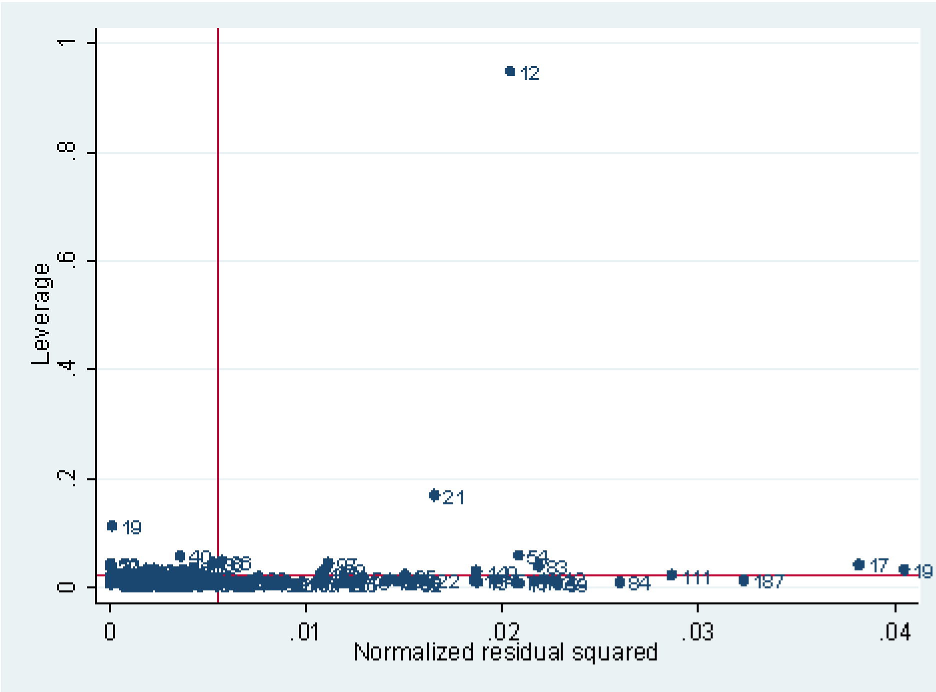
```
lvr2plot
```



Answer:

lvr2plot is the leverage against residual squared plot. The upper left corner of the plot will be points that are high in leverage and the lower right corner will be points that are high in the absolute of residuals. The upper right portion will be those points that are both high in leverage and in the absolute of residuals. There is one point in this plot that stands out so much differently from any other point. There are many ways of figuring out what this point is. First of all, graphically, we can add an option in our lvr2plot command to see which observation is associated with the extreme point on the plot.

```
lvr2plot, ml(subject)
```



There are also numerical measures that we can deploy. Since it is obviously very high on leverage, we can first generate **leverage** and list the extreme ones.

```
predict l, leverage
hilo l  measwt measht reptwt reptht  subject, high show(5)
5 highest observations on l

        l      measwt      measht      reptwt      reptht     subject
0578113          65         187          67         188          40
0596073         102         185         107         185          54
1136993          76         197          75         200          19
1702566         119         180         124         178          21
9481246         166          57          56         163          12
```

The other way is to use **Cook's D** since Cook's D is the combination of leverage and residual.

```
predict c, cooksd
hilo c  measwt measht reptwt reptht  subject, high show(5)
5 highest observations on c

        c      measwt      measht      reptwt      reptht     subject
0619987         102         185         107         185          54
0628549          88         185          93         188         191
0779325          92         187         101         185          17
1808358         119         180         124         178          21
 317.8551         166          57          56         163          12
```

We can also look at **studentized residuals**.

```
predict rstu, rstu
hilo  rstu measwt measht reptwt reptht subject, show(5)
5 lowest and highest observations on rstu
```

```
     rstu     measwt     measht     reptwt     reptht    subject
-2.772892         88        185         93        188        191
-2.703085         92        187        101        185         17
-2.305224         84        183         90        183        111
-2.023018         53        169         52        175         83
-1.994573        102        185        107        185         54

     rstu     measwt     measht     reptwt     reptht    subject
 2.030575         58        161         51        159          2
 2.031815         75        172         70        169         59
 2.176899         60        167         55        163         84
 2.440577         60        172         55        168        187
 10.67515        166         57         56        163         12
```

In all of the above, we see that **subject 12** is a problematic point. Is it an entry error? Yes. Apparently for subject 12 the measured weight has been switched with measured height. We can be very much sure on this case. Therefore, we can switch them back. We then perform the same analysis again.

```
replace measwt=57 if  subject==12
(1 real change made)

replace measht=166 if subject==12
(1 real change made)

list  subject measwt  measht reptwt reptht in 12/12

       subject     measwt     measht     reptwt     reptht
 12.        59         75        172         70        169

regress  measwt measht reptwt reptht

  Source |       SS       df       MS                  Number of obs =
181
---------+------------------------------               F(  3,   177) =
2085.02
   Model |  31551.0849     3  10517.0283               Prob > F      =
0.0000
Residual |  892.804651   177  5.04409407               R-squared     =
0.9725
---------+------------------------------               Adj R-squared =
0.9720
   Total |  32443.8895   180  180.243831               Root MSE      =
2.2459

----------------------------------------------------------------------------
-
  measwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+------------------------------------------------------------------
-
  measht |  -.0364477    .088613     -0.411   0.681      -.2113216
.1384262
  reptwt |    .963793   .0194467     49.561   0.000       .9254157
1.00217
  reptht |   .0225427   .0811435      0.278   0.781      -.1375904
.1826759
```

```
   _cons |   4.821849   4.242671      1.137   0.257      -3.550881
13.19458
----------------------------------------------------------------------------
-
```

We now see that both measured height and reported height are no longer significant predictors. This is because that the predictors are collinear to each other since we have corrected the entry error. Let's do another regression with only reported weight as a single predictor. Notice that adjusted R-square is actually the highest among all the regression analysis we have done so far. This shows that data entry error could really distort the regression analysis sometimes.

regress measwt reptwt

```
  Source |       SS       df       MS                  Number of obs =
181
---------+------------------------------               F(  1,   179) =
6315.72
   Model |  31549.7087     1  31549.7087               Prob > F      =
0.0000
Residual |  894.180797   179  4.99542345               R-squared     =
0.9724
---------+------------------------------               Adj R-squared =
0.9723
   Total |  32443.8895   180  180.243831               Root MSE      =
2.235

----------------------------------------------------------------------------
-
  measwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+------------------------------------------------------------------
-
  reptwt |   .9569886   .0120419     79.472   0.000       .9332262
.9807509
   _cons |   2.847071   .8081664      3.523   0.001       1.252311
4.44183
----------------------------------------------------------------------------
-
```

2. Continue with the first model we run in our last exercise. What measure and its corresponding STATA command would you use if you want to know how much change an observation would make on a predictor? For example, how much change would it be for the coefficient of predictor **reptht** if we omit observation 12 from our regression analysis? What are the other measures that you would use to assess the strength of an observation on regression? What are the commonly suggested cut-off values for them?

Answer: The measure that measures how much impact each observation has on a particular predictor is DFBETAs. The DFBETA for a predictor and for a particular observation is the difference between the regression coefficient calculated for all of the data and the regression coefficient calculated with the observation deleted, scaled by the standard error calculated with the observation deleted. The cut-off value for DFBETAs is 2/sqrt(n), where n is the number of observations. In our case, it will be the absolute value of DFBETAs greater than 2/sqrt(181)=.14866. From our list below, we can see we have several troublesome points with observation 12 the most troublesome one. For observation 12, the DFreptht is 24.25463. That means that including observation 12 in the regression, the regression coefficient for **reptht** will increase by about 24 times the standard error than the case with the observation excluded.

```
dfbeta
DFmeasht:  DFbeta(measht)
DFreptwt:  DFbeta(reptwt)
DFreptht:  DFbeta(reptht)

hilo  DFreptht measwt measht reptwt reptht subject, show(5)
5 lowest and highest observations on DFreptht

 DFreptht     measwt     measht     reptwt     reptht    subject
-.3410896         53        169         52        175         83
-.2115161         88        185         93        188        191
-.1834869         59        182         61        183         86
-.1629187         65        187         67        188         40
-.1510676         79        179         79        171        112

 DFreptht     measwt     measht     reptwt     reptht    subject
0904913        63        160        64        158         78
1255461        69        167        73        165        122
1791834        85        191        83        188        140
4119168       119        180       124        178         21
 24.25463        166         57         56        163         12
```

DFBETAs are calculation intensive as it is for computed each predictor and each observation. DFITS and Cook's D, on the other hand, are summary information of the influence (leverage and residual) and are much less computation intensive. For example, we can look at DFITS after the regression, similar to what we did in Exercise 1. The cut-off values of DFITS and Cook's D is 2*sqrt(k/n) and 4/n respectively. Observations with DFITS or Cook's D value greater than these cut-off values deserve further investigation.

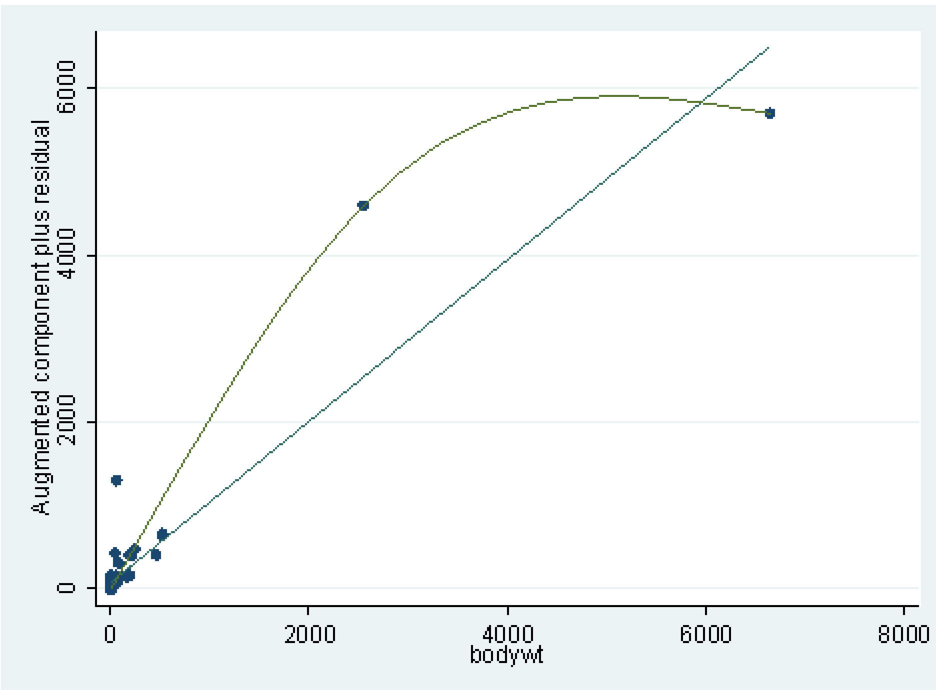
3. The following data file is called **bbwt.dta** and it is from Weisberg's Applied Regression Analysis. It consists of the body weights and brain weight of some 60 animals. We want to predict the brain weight by body weight, that is, a simple linear regression of brain weight against body weight. Show what you have to do to verify the linearity assumption. If you think that it violates the linearity assumption, show some possible remedies that you would consider.

```
use bbwt, clear
regress brainwt bodywt
  Source |       SS       df       MS                  Number of obs =
62
---------+------------------------------               F(  1,    60) =
411.12
   Model |  46067326.8     1  46067326.8               Prob > F      =
0.0000
Residual |  6723217.18    60   112053.62               R-squared     =
0.8726
---------+------------------------------               Adj R-squared =
0.8705
   Total |  52790543.9    61  865418.753               Root MSE      =
334.74
```

```
------------------------------------------------------------------------------
-
 brainwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
  bodywt |   .9664599   .0476651     20.276   0.000       .8711155
1.061804
   _cons |   91.00865   43.55574      2.089   0.041       3.884201
178.1331
------------------------------------------------------------------------------
-
```

Answer: In general, we can use **acprplot** to verify the linearity assumption against a predictor. For example, we can do after the regression above the acprplot against our only predictor **bodywt**.

acprplot bodywt, mspline



The graph does not look very linear. In our chapter, we did some logarithm transformations. We'll try it here and the results are shown below. Notice the plot is much nicer this time. The adjusted R-square is also up by .05.

gen lbdwt=log(bodywt)

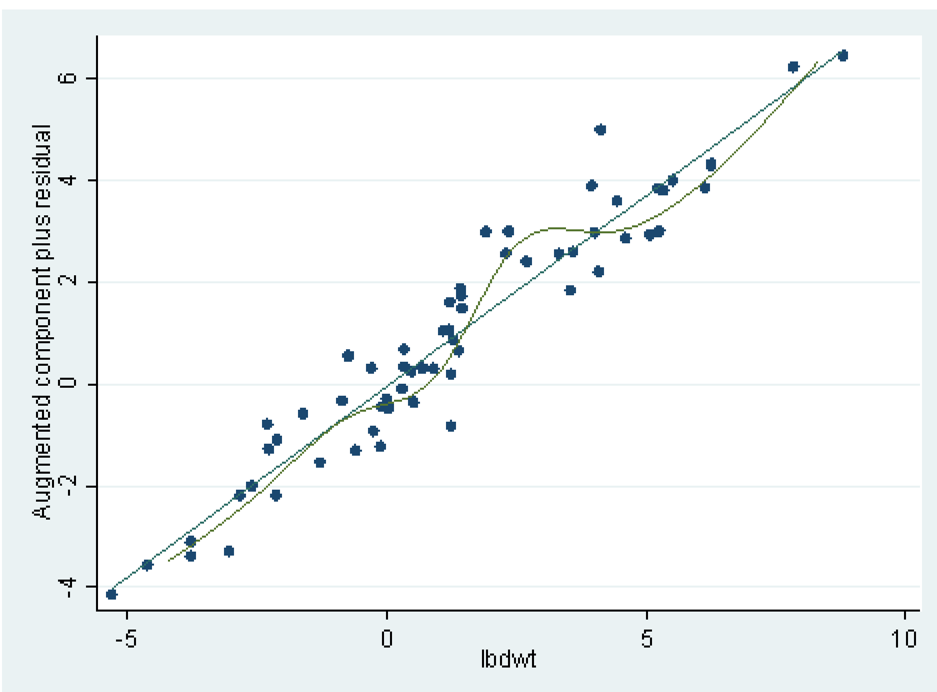
gen lbrwt=log(brainwt)

regress lbrwt lbdwt

```
  Source |       SS       df       MS                  Number of obs =
62
```

```
---------+------------------------------                F(  1,    60) =
697.42
   Model |  336.188605     1  336.188605                Prob > F      =
0.0000
Residual |  28.9226087    60  .482043478                R-squared     =
0.9208
---------+------------------------------                Adj R-squared =
0.9195
   Total |  365.111213    61  5.98542973                Root MSE      =
.69429

-----------------------------------------------------------------------------
-
   lbrwt |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
   lbdwt |   .7516861   .0284635     26.409   0.000       .6947507
.8086216
   _cons |   2.134788   .0960432     22.227   0.000       1.942673
2.326903
-----------------------------------------------------------------------------
-
```

acprplot lbdwt, mspline

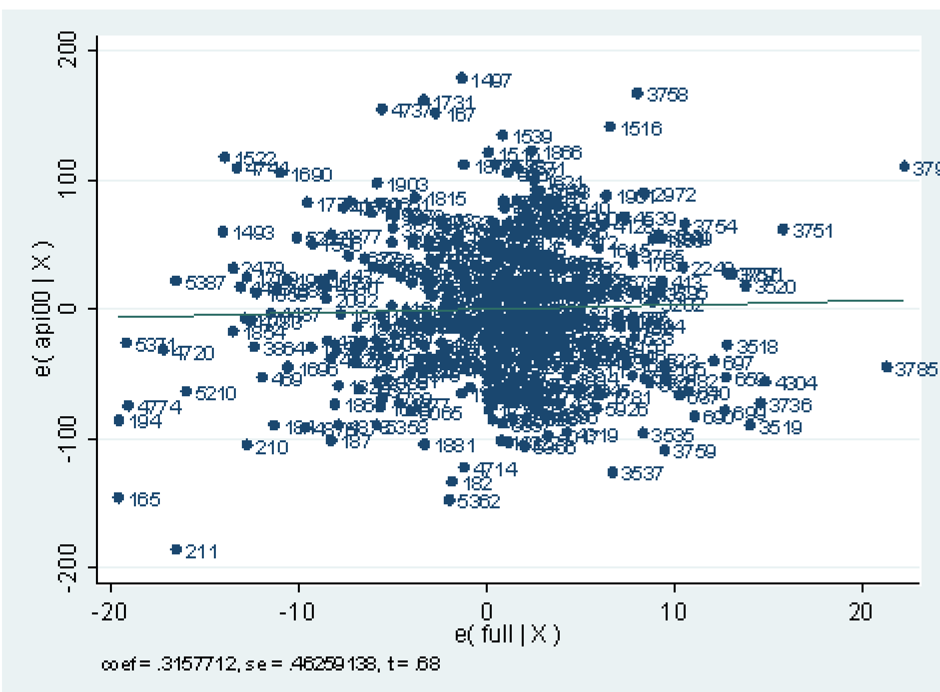


4. We did a regression analysis using data file **elemapi** in chapter 2. Continuing with the analysis we did, we did an avplot here. Explain what an avplot is and how you would interpret the avplot below. If **full** were put in the model, would it be a significant predictor?

use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
regress api00 meals ell emer

```
  Source |       SS       df       MS                  Number of obs =
400
---------+------------------------------               F(  3,   396) =
673.00
   Model |  6749782.75     3  2249927.58               Prob > F      =
0.0000
Residual |  1323889.25   396  3343.15467               R-squared     =
0.8360
---------+------------------------------               Adj R-squared =
0.8348
   Total |  8073672.00   399  20234.7669               Root MSE      =
57.82

------------------------------------------------------------------------------
-
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
   meals |  -3.159189   .1497371    -21.098   0.000      -3.453568   -
2.864809
     ell |  -.9098732   .1846442     -4.928   0.000      -1.272878   -
.5468678
    emer |  -1.573496    .293112     -5.368   0.000      -2.149746   -
.9972456
   _cons |   886.7033    6.25976    141.651   0.000       874.3967
899.0098
------------------------------------------------------------------------------
-
```

avplot full, mlabel(snum)



<u>Answer</u>: A group of points can be jointly influential. An avplot is an attractive graphic method to present multiple influential points on a predictor. What we are looking for in an avplot are those points that can exert substantial change to the regression line. For example, in the plot above, the observation with school number 211 is very low at the left corner of the plot. Deleting it would flatten the regression line a lot, in other words, it would decrease the regression coefficient for variable **full** significantly. You can compare the regression that includes the variable full and the entire data set and the model without the observation with snum 211.

regress api00 meals ell emer full

```
  Source |       SS       df       MS                  Number of obs =
400
---------+------------------------------               F(  4,   395) =
504.18
   Model |  6751342.63     4  1687835.66               Prob > F      =
0.0000
Residual |  1322329.37   395  3347.66928               R-squared     =
0.8362
---------+------------------------------               Adj R-squared =
0.8346
   Total |  8073672.00   399  20234.7669               Root MSE      =
57.859

----------------------------------------------------------------------------
-
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+------------------------------------------------------------------
-
   meals |  -3.156558   .1498877    -21.059   0.000      -3.451236   -
2.861881
     ell |  -.8981675   .1855628     -4.840   0.000      -1.262982   -
.5333532
    emer |  -1.225015     .58877     -2.081   0.038       -2.38253   -
.0675008
    full |   .3157712   .4625914      0.683   0.495      -.5936778
1.22522
   _cons |   855.0671   46.76702     18.284   0.000       763.1237
947.0105
----------------------------------------------------------------------------
-
```

regress api00 meals ell emer full if snum !=211

```
  Source |       SS       df       MS                  Number of obs =
399
---------+------------------------------               F(  4,   394) =
513.16
   Model |  6715948.02     4  1678987.01               Prob > F      =
0.0000
Residual |  1289106.10   394  3271.84289               R-squared     =
0.8390
---------+------------------------------               Adj R-squared =
0.8373
```

```
   Total |  8005054.12   398  20113.2013              Root MSE      =
57.20

------------------------------------------------------------------------------
-
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
   meals |  -3.164431   .1482011    -21.352   0.000      -3.455795   -
2.873067
     ell |  -.8930366   .1834563     -4.868   0.000      -1.253712   -
.5323609
    emer |  -1.411583    .585001     -2.413   0.016      -2.561697   -
.2614692
    full |   .1213333   .4613751      0.263   0.793      -.7857315
1.028398
   _cons |   874.6425   46.64066     18.753   0.000       782.9468
966.3382
------------------------------------------------------------------------------
-
```

Of course, there are other points that are in similar nature as the observation with snum 211 shown in the avplot that are worth paying more attention to. On the other hand, if we look at the t-value on top of the avplot, it is only 68. The p-value corresponding to it will be the probability for t-distribution with degree of freedom being the total degree of freedom. :

```
di tprob(399, .683)
49500322
```

which is not significant. The equation on top of the avplot is actually the regression coefficient and its standard error if the variable were a predictor. In our regression which includes **full** and all the data, we see that coefficient for **full** is .3157712 and the standard error for it is .4625914. They are exactly the same as shown on top of the avplot.

5. The data set **wage.dta** is from a national sample of 6000 households with a male head earning less than $15,000 annually in 1966. The data were classified into 39 demographic groups for analysis. We tried to predict the average hours worked by average age of respondent and average yearly non-earned income.

```
use wage, clear
regress HRS AGE NEIN
  Source |       SS       df       MS                  Number of obs =
39
---------+-----------------------------                F(  2,    36) =
39.72
   Model |  107205.109     2  53602.5543               Prob > F      =
0.0000
Residual |  48578.1222    36  1349.39228               R-squared     =
0.6882
---------+-----------------------------                Adj R-squared =
0.6708
   Total |  155783.231    38   4099.5587               Root MSE      =
36.734
```

```
------------------------------------------------------------------------------
-
     HRS |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
     AGE |  -8.281632   1.603736     -5.164   0.000      -11.53416   -
5.029104
    NEIN |   .4289202   .0484882      8.846   0.000       .3305816
.5272588
   _cons |    2321.03   57.55038     40.330   0.000       2204.312
2437.748
------------------------------------------------------------------------------
-
```

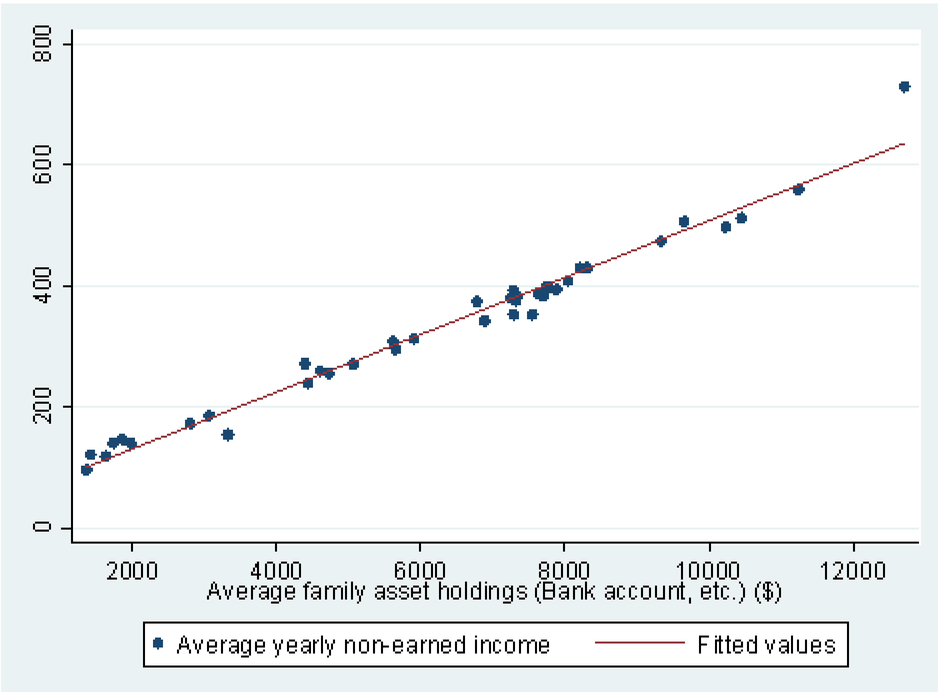
Both predictors are significant. Now if we add ASSET to our predictors list, neither NEIN nor ASSET is significant.

```
regress HRS AGE NEIN ASSET
  Source |       SS       df       MS                  Number of obs =
39
---------+------------------------------               F(  3,    35) =
25.83
   Model |  107317.64     3  35772.5467                Prob > F      =
0.0000
Residual |  48465.5908    35  1384.73117               R-squared     =
0.6889
---------+------------------------------               Adj R-squared =
0.6622
   Total |  155783.231    38   4099.5587               Root MSE      =
37.212

------------------------------------------------------------------------------
-
     HRS |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
     AGE |  -8.007181    1.88844     -4.240   0.000      -11.84092   -
4.173443
    NEIN |   .3338277    .337171      0.990   0.329      -.3506658
1.018321
   ASSET |   .0044232    .015516      0.285   0.777       -.027076
.0359223
   _cons |   2314.054   63.22636     36.600   0.000       2185.698
2442.411
------------------------------------------------------------------------------
-
```

Can you explain why?

Answer: If we look at our data set more carefully, for example, we can do a describe at the beginning of regression analysis, we would notice that variable NEIN and ASSET are very closed related. Therefore, we would expect that these two variables are strongly correlated. We can also do a scatter plot to check on this. Here is what we have done:

```
describe  NEIN ASSET
              storage  display     value
variable name   type   format      label      variable label
-----------------------------------------------------------------------------
--
NEIN            float  %9.0g                  Average yearly non-earned
                                                income
ASSET           float  %9.0g                  Average family asset holdings
                                                (Bank account, etc.) ($)
twoway (scatter  NEIN ASSET) (lfit NEIN ASSET)
```



Another useful command introduced in this chapter is **vif**.

```
regress HRS AGE NEIN
(Output is shown above.)

vif

Variable |       VIF       1/VIF
---------+----------------------
     AGE |      1.29    0.774467
    NEIN |      1.29    0.774467
---------+----------------------
Mean VIF |      1.29

regress HRS AGE NEIN ASSET
(Output is shown above.)

vif
```

```
Variable |       VIF       1/VIF
---------+----------------------
    NEIN |     60.84    0.016436
   ASSET |     56.07    0.017836
     AGE |      1.74    0.573178
---------+----------------------
Mean VIF |     39.55
```

So we see that in the first regression, there is no evidence of collinearity since the variance inflation factors are fairly small. But in the second regression analysis, the **vif** for NEIN and ASSET jumped up to around 60, which indicates strongly the appearance of collinearity among the predictors. The collinearity can also be detected by using the command **collin**.

```
collin  NEIN ASSET AGE

  Collinearity Diagnostics

                           SQRT                              Cond
  Variable        VIF      VIF    Tolerance  Eigenval       Index
-------------------------------------------------------------
      NEIN      60.84     7.80     0.0164      2.2855      1.0000
     ASSET      56.07     7.49     0.0178      0.7059      1.7994
       AGE       1.74     1.32     0.5732      0.0086     16.3386
-------------------------------------------------------------
  Mean VIF      39.55               Condition Number   16.3386
```

6. Continue to use the previous data set. This time we want to predict the average hourly wage by the average percent of white respondents. Carry out the regression analysis and list the STATA commands that you can use to check for heteroscedasticity. Explain the results of the test(s).

```
use wage, clear
regress  RATE RACE

  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  1,    29) =
22.82
   Model |  2.16442894     1  2.16442894               Prob > F      =
0.0000
Residual |  2.75013286    29  .094832168               R-squared     =
0.4404
---------+------------------------------               Adj R-squared =
0.4211
   Total |  4.91456181    30  .163818727               Root MSE      =
.30795

-----------------------------------------------------------------------------
-
    RATE |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
    RACE |  -.0142697   .0029869     -4.777   0.000      -.0203786   -
.0081608
```

```
   _cons |   3.367147   .1261571     26.690   0.000       3.109127
3.625168
-----------------------------------------------------------------------------
-
```

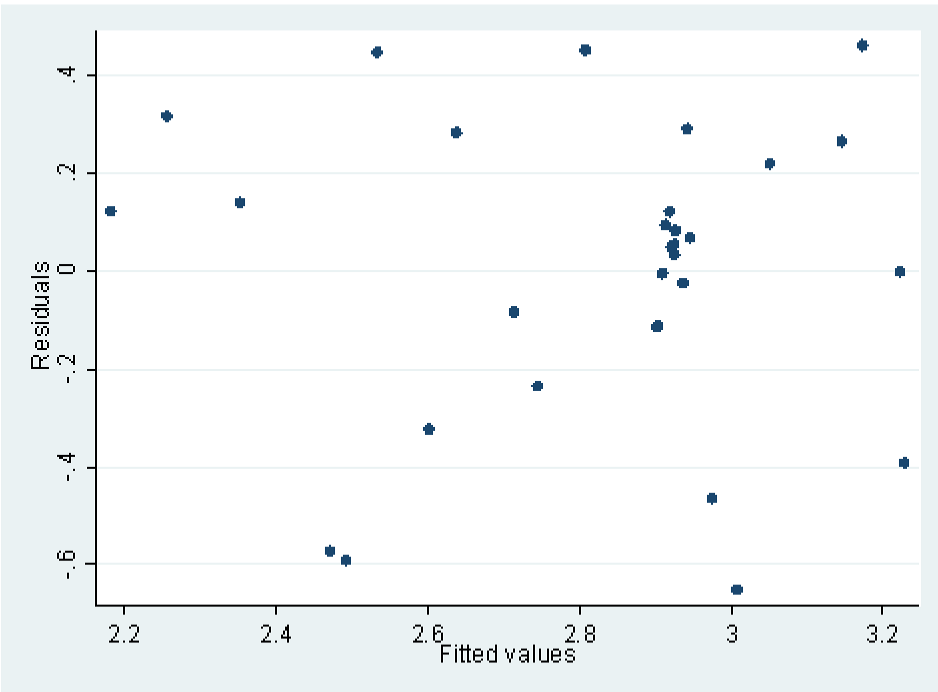
hettest

```
Cook-Weisberg test for heteroscedasticity using fitted values of RATE
     Ho: Constant variance
         chi2(1)      =      0.42
         Prob > chi2  =      0.5186
```

whitetst

```
(8 missing values generated)
(8 missing values generated)

White's general test statistic :  .5617374  Chi-sq( 2)  P-value =  .7551
```

rvfplot



The **hettest** and **whitetst** are based on the null hypothesis that the variance is constant. Therefore, when the probability is large, we will accept the null hypothesis of constant variance. The **rvfplot** also shows that the variance across fitted values does not change a lot, as overall speaking we see a band of equal width. On the other hand, the regression below is different. Both hettest and whitetst are significant, indicating heteroscedasticity. This can also be seen from the rvfplot below, we see that the band is getting wider to the right.

regress RACE HRS

```
  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  1,    29) =
65.14
```

```
   Model |  7355.07438     1  7355.07438               Prob > F      =
0.0000
Residual |   3274.4589    29  112.912376               R-squared     =
0.6919
---------+------------------------------               Adj R-squared =
0.6813
   Total |  10629.5333    30  354.317776               Root MSE      =
10.626

-----------------------------------------------------------------------------
-
    RACE |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
     HRS |  -.2801356   .0347093     -8.071   0.000       -.351124   -
.2091472
   _cons |   639.3401   74.53629      8.578   0.000       486.8963
791.7839
-----------------------------------------------------------------------------
-
```

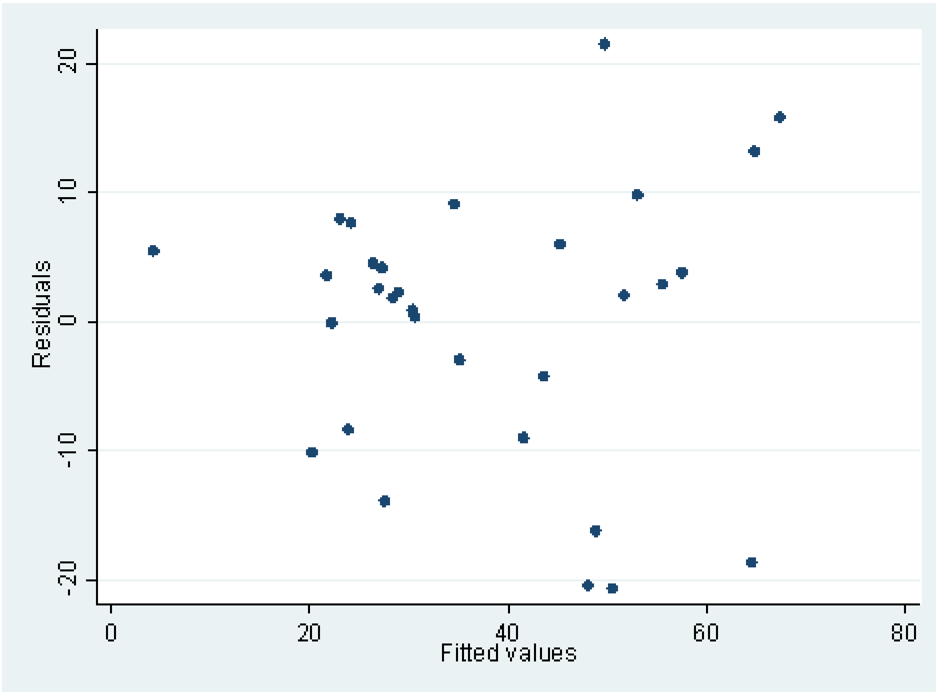
hettest

```
Cook-Weisberg test for heteroscedasticity using fitted values of RACE
     Ho: Constant variance
         chi2(1)      =      6.60
         Prob > chi2  =      0.0102
```

whitetst

```
White's general test statistic :  7.889606  Chi-sq( 2)  P-value =  .0194
```

rvfplot



7. We have a data set that consists of volume, diameter and height of some objects. Someone did a regression of volume on diameter and height.

```
use tree, clear
regress vol dia height
  Source |       SS        df       MS                         Number of obs =
31
---------+------------------------------                       F(  2,    28) =
254.97
   Model |  7684.16254     2  3842.08127                       Prob > F      =
0.0000
Residual |  421.921306    28  15.0686181                       R-squared     =
0.9480
---------+------------------------------                       Adj R-squared =
0.9442
   Total |  8106.08385    30  270.202795                       Root MSE      =
3.8818

------------------------------------------------------------------------------
-
     vol |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
     dia |   4.708161   .2642646     17.816   0.000       4.166839
5.249482
  height |   .3392513   .1301512      2.607   0.014       .0726487
.6058538
   _cons |  -57.98766   8.638225     -6.713   0.000      -75.68226   -
40.29306
```

```
------------------------------------------------------------------------------
-
```

Explain what tests you can use to detect model specification errors and if there is any, your solution to correct it.

Answer: We can use **linktest** and **ovtest** to detect model specification errors.

linktest

```
  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  2,    28) =
594.19
   Model |  7919.48998     2  3959.74499               Prob > F      =
0.0000
Residual |  186.593864    28  6.66406657               R-squared     =
0.9770
---------+------------------------------               Adj R-squared =
0.9753
   Total |  8106.08385    30  270.202795               Root MSE      =
2.5815

------------------------------------------------------------------------------
-
     vol |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------------
-
    _hat |   .3606632   .1115454      3.233   0.003       .1321728
.5891537
  _hatsq |   .0094227   .0015856      5.942   0.000       .0061746
.0126707
   _cons |   8.376438   1.729554      4.843   0.000       4.833608
11.91927
------------------------------------------------------------------------------
-
```

ovtest

```
Ramsey RESET test using powers of the fitted values of vol
       Ho:  model has no omitted variables
                  F(3, 25) =     11.54
                  Prob > F =      0.0001
```

For linktest we look for p-value for the square term and both the linktest and ovtest are significant indicating that our model is **not** specified correctly. It is actually easy to understand in this case, since we look for the relationship between volume, which is 3-dimensional and diameter and height, which are 1-dimensional. So it is reasonable to put in higher degree terms. One solution is to put the squared diameter term into our regression as shown below. Both the linktest and ovtest are no longer significant.

gen dia2=dia*dia

```
regress  vol dia dia2 height

  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  3,    27) =
383.20
   Model |  7920.07197     3  2640.02399               Prob > F      =
0.0000
Residual |  186.011883    27    6.889329               R-squared     =
0.9771
---------+------------------------------               Adj R-squared =
0.9745
   Total |  8106.08385    30  270.202795               Root MSE      =
2.6248

-----------------------------------------------------------------------------
-
     vol |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
     dia |  -2.885077   1.309851     -2.203   0.036      -5.572669   -
.1974846
    dia2 |   .2686224   .0459048      5.852   0.000       .1744335
.3628112
  height |   .3763873    .088232      4.266   0.000       .1953502
.5574244
   _cons |  -9.920417   10.07912     -0.984   0.334      -30.60105
10.76022
-----------------------------------------------------------------------------
-
linktest

  Source |       SS       df       MS                  Number of obs =
31
---------+------------------------------               F(  2,    28) =
596.13
   Model |  7920.08338     2  3960.04169               Prob > F      =
0.0000
Residual |   186.00047    28  6.64287391               R-squared     =
0.9771
---------+------------------------------               Adj R-squared =
0.9754
   Total |  8106.08385    30  270.202795               Root MSE      =
2.5774

-----------------------------------------------------------------------------
-
     vol |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------------
-
    _hat |   1.004811   .1199563      8.376   0.000       .7590915
1.25053
  _hatsq |  -.0000621   .0015027     -0.041   0.967      -.0031403
.0030161
```

```
   _cons |  -.0727509   2.019028     -0.036   0.972      -4.208542
4.06304
-----------------------------------------------------------------------------
-
```

ovtest

```
Ramsey RESET test using powers of the fitted values of vol
       Ho:  model has no omitted variables
                  F(3, 24) =      0.43
                  Prob > F =      0.7312
```

Chapter 3 - Regression with Categorical Predictors

Chapter Outline

Please note: This page makes use of the program **xi3** which is no longer being maintained and has been from our archives. References to **xi3** will be left on this page because they illustrate specific principles of coding categorical variables.

3.0 Introduction

In the previous two chapters, we have focused on regression analyses using continuous variables. However, it is possible to include categorical predictors in a regression analysis, but it requires some extra work in performing the analysis and extra work in properly interpreting the results. This chapter will illustrate how you can use Stata for including categorical predictors in your analysis and describe how to interpret the results of such analyses. Stata has some great tools that really ease the process of including categorical variables in your regression analysis, and we will emphasize the use of these timesaving tools.

This chapter will use the **elemapi2** data that you have seen in the prior chapters. We will focus on four variables **api00**, **some_col**, **yr_rnd** and **mealcat**, which takes **meals** and breaks it up into 3 categories. Let's have a quick look at these variables.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2
describe api00  some_col yr_rnd  mealcat

               storage display     value
variable name   type   format      label      variable label
-----------------------------------------------------------------------
--------
api00           int    %6.0g                  api 2000
some_col        byte   %4.0f                  parent some college
yr_rnd          byte   %4.0f       yr_rnd     year round school
mealcat         byte   %18.0g      mealcat    Percentage free meals in
3
                                                categories
```

The variable **api00** is a measure of the performance of the schools. Below we see the codebook information for **api00**

```
codebook api00
api00 ---------------------------------------------------------------
api 2000
                  type:  numeric (int)

                 range:  [369,940]                    units:  1
         unique values:  271                  coded missing:  0 / 400

                  mean:   647.622
              std. dev:   142.249

           percentiles:        10%       25%       50%       75%
90%
                             465.5     523.5       643     762.5
850
```

The variable **some_col** is a continuous variable that measures the percentage of the parents in the school who have attended college, and the codebook information is shown below.

```
codebook some_col
some_col -------------------------------------------------- parent some
college
                  type:  numeric (byte)

                 range:  [0,67]                       units:  1
         unique values:  49                   coded missing:  0 / 400

                  mean:   19.7125
              std. dev:   11.3369

           percentiles:        10%       25%       50%       75%
90%
                               2.5        12        19        28
34
```

The variable **yr_rnd** is a categorical variable that is coded 0 if the school is not year round, and 1 if year round, see below.

```
codebook yr_rnd
yr_rnd ------------------------------------------------------ year
round school
                  type:  numeric (byte)
                 label:  yr_rnd

                 range:  [0,1]                        units:  1
         unique values:  2                    coded missing:  0 / 400

           tabulation:  Freq.   Numeric  Label
                          308         0  No
                           92         1  Yes
```

The variable **meals** is the percentage of students who are receiving state sponsored free meals and can be used as an indicator of poverty. This was broken into 3 categories (to make equally sized groups) creating the variable **mealcat**. The codebook information for **mealcat** is shown below.

```
codebook mealcat
mealcat ----------------------------------------------------------
(unlabeled)
                  type:  numeric (float)
                 label:  mealcat

                 range:  [1,3]                         units:  1
         unique values:  3                     coded missing:  0 / 400

            tabulation:  Freq.   Numeric  Label
                           131         1  0-46% free meals
                           132         2  47-80% free meals
                           137         3  81-100% free meals
```

3.1 Regression with a 0/1 variable

The simplest example of a categorical predictor in a regression analysis is a 0/1 variable, also called a dummy variable. Let's use the variable **yr_rnd** as an example of a dummy variable. We can include a dummy variable as a predictor in a regression analysis as shown below.

```
regress api00 yr_rnd
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
116.24
       Model |  1825000.56     1  1825000.56           Prob > F      =
0.0000
    Residual |  6248671.43   398  15700.1795           R-squared     =
0.2260
-------------+------------------------------           Adj R-squared =
0.2241
       Total |  8073672.00   399  20234.7669           Root MSE      =
125.30

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      yr_rnd |  -160.5064    14.8872   -10.78   0.000    -189.7737    -
131.239
       _cons |    684.539    7.13965    95.88   0.000     670.5028
698.5751
------------------------------------------------------------------------
-------
```

This may seem odd at first, but this is a legitimate analysis. But what does this mean? Let's go back to basics and write out the regression equation that this model implies.

```
api00 = _cons + Byr_rnd * yr_rnd
```

where **_cons** is the intercept (or constant) and we use **Byr_rnd** to represent the coefficient for variable **yr_rnd**. Filling in the values from the regression equation, we get

```
api00 = 684.539 + -160.5064 * yr_rnd
```

If a school is not a year-round school (i.e. **yr_rnd** is 0) the regression equation would simplify to

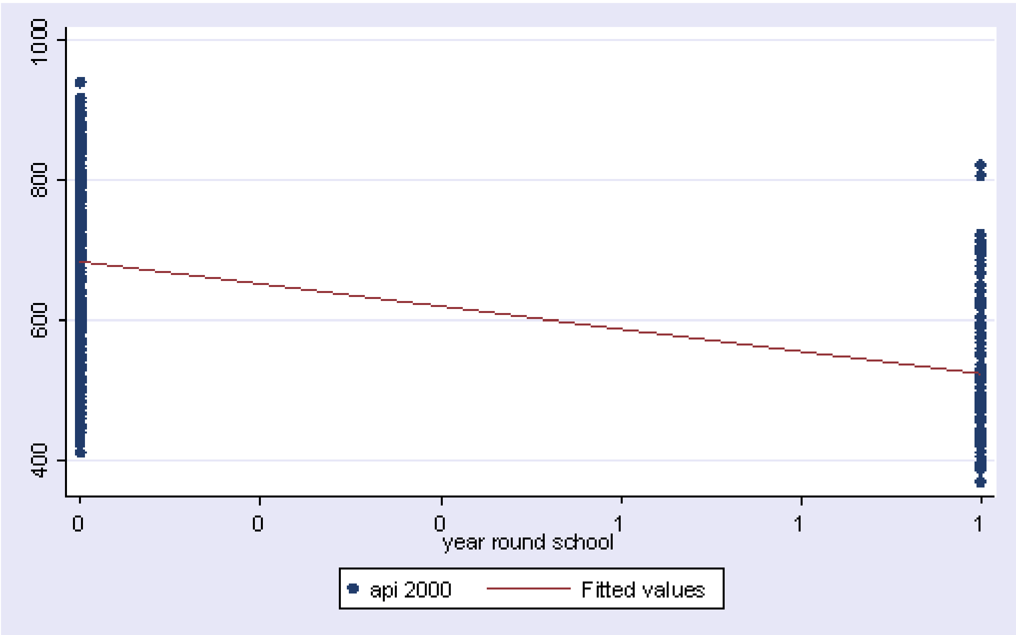
```
api00 = constant    + 0 * Byr_rnd
api00 = 684.539     + 0 * -160.5064
api00 = 684.539
```

If a school is a year-round school, the regression equation would simplify to

```
api00 = constant + 1 * Byr_rnd
api00 = 684.539  + 1 * -160.5064
api00 = 524.0326
```

We can graph the observed values and the predicted values using the **scatter** command as shown below. Although **yr_rnd** only has 2 values, we can still draw a regression line showing the relationship between **yr_rnd** and **api00**. Based on the results above, we see that the predicted value for non-year round schools is 684.539 and the predicted value for the year round schools is 524.032, and the slope of the line is negative, which makes sense since the coefficient for **yr_rnd** was negative (-160.5064).

```
twoway (scatter api00 yr_rnd) (lfit api00 yr_rnd)
```



Let's compare these predicted values to the mean **api00** scores for the year-round and non-year-round schools.

```
tabulate yr_rnd, sum(api00)
 year round |         Summary of api 2000
     school |        Mean   Std. Dev.       Freq.
------------+------------------------------------
         No |   684.53896   132.11253         308
        Yes |   524.03261   98.916043          92
------------+------------------------------------
      Total |    647.6225   142.24896         400
```

As you see, the regression equation predicts that the value of **api00** will be the mean value, depending on whether a school is a year round school or non-year round school.

Let's relate these predicted values back to the regression equation. For the non-year-round schools, their mean is the same as the intercept (684.539). The coefficient for **yr_rnd** is the amount we need to add to get the mean for the year-round schools, i.e., we need to add -160.5064 to get 524.0326, the mean for the non year-round schools. In other words, **Byr_rnd** is the mean **api00** score for the year-round schools minus the mean **api00** score for the non year-round schools, i.e., mean(year-round) - mean(non year-round).

It may be surprising to note that this regression analysis with a single dummy variable is the same as doing a t-test comparing the mean **api00** for the year-round schools with the non year-round schools (see below). You can see that the t value below is the same as the t value for **yr_rnd** in the regression above. This is because **Byr_rnd** compares the year-rounds and non year-rounds (since the coefficient is mean(year round)-mean(non year-round)).

```
ttest api00, by(yr_rnd)

Two-sample t test with equal variances

----------------------------------------------------------------------
-------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf.
Interval]
---------+------------------------------------------------------------
-------
      No |     308     684.539     7.52781    132.1125    669.7263
699.3516
     Yes |      92    524.0326    10.31271    98.91604    503.5477
544.5175
---------+------------------------------------------------------------
-------
combined |     400    647.6225    7.112448     142.249    633.6399
661.6051
---------+------------------------------------------------------------
-------
    diff |            160.5064     14.8872                 131.239
189.7737
----------------------------------------------------------------------
-------
```

```
Degrees of freedom: 398

                     Ho: mean(No) - mean(Yes) = diff = 0

     Ha: diff < 0               Ha: diff ~= 0               Ha: diff > 0
       t =  10.7815                t =  10.7815                t =
10.7815
   P < t =   1.0000          P > |t| =   0.0000          P > t =
0.0000
```

Since a t-test is the same as doing an **anova**, we can get the same results using the **anova** command as well.

`anova api00 yr_rnd`

```
                           Number of obs =     400     R-squared     =
0.2260
                           Root MSE      =  125.30     Adj R-squared =
0.2241

                  Source |  Partial SS    df       MS           F
Prob > F
              -----------+-------------------------------------------------
-------
                   Model |  1825000.56     1  1825000.56     116.24
0.0000
                         |
                  yr_rnd |  1825000.56     1  1825000.56     116.24
0.0000
                         |
                Residual |  6248671.43   398  15700.1795
              -----------+-------------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

If we square the t-value from the t-test, we get the same value as the F-value from the anova.

`di 10.7815^2`

```
116.24074
```

3.2 Regression with a 1/2 variable

A categorical predictor variable does not have to be coded 0/1 to be used in a regression model. It is easier to understand and interpret the results from a model with dummy variables, but the results from a variable coded 1/2 yield essentially the same results.

Lets make a copy of the variable **yr_rnd** called **yr_rnd2** that is coded 1/2, 1=non year-round and 2=year-round.

```
generate yr_rnd2=yr_rnd
recode yr_rnd2 0=1 1=2
```

```
(400 changes made)
```

Let's perform a regression predicting **api00** from **yr_rnd2**.

```
regress api00 yr_rnd2

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
116.24
       Model |  1825000.56     1  1825000.56           Prob > F      =
0.0000
    Residual |  6248671.43   398  15700.1795           R-squared     =
0.2260
-------------+------------------------------           Adj R-squared =
0.2241
       Total |  8073672.00   399  20234.7669           Root MSE      =
125.30

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
     yr_rnd2 |  -160.5064    14.8872   -10.78   0.000    -189.7737    -
131.239
       _cons |   845.0453   19.35336    43.66   0.000     806.9977
883.0929
------------------------------------------------------------------------
-------
```

Note that the coefficient for **yr_rnd** is the same as **yr_rnd2**. So, you can see that if you code **yr_rnd** as 0/1 or as 1/2, the regression coefficient works out to be the same. However the intercept (_cons) is a bit less intuitive. When we used **yr_rnd**, the intercept was the mean for the non year-rounds. When using **yr_rnd2**, the intercept is the mean for the non year-rounds minus **Byr_rnd2**, i.e., 684.539 - (-160.506) = 845.045

Note that you can use 0/1 or 1/2 coding and the results for the coefficient come out the same, but the interpretation of the constant in the regression equation is different. It is often easier to interpret the estimates for 0/1 coding.

In summary, these results indicate that the **api00** scores are significantly different for the schools depending on the type of school, year round school vs. non-year round school. Non year-round schools have significantly higher API scores than year-round schools. Based on the regression results, non year- round schools have scores that are 160.5 points higher than year- round schools.

3.3 Regression with a 1/2/3 variable

3.3.1 Manually Creating Dummy Variables

Say, that we would like to examine the relationship between the amount of poverty and api scores. We don't have a measure of poverty, but we can use **mealcat** as a proxy for a measure of poverty. Below we repeat the codebook info for **mealcat** showing the values for the three categories.

```
codebook mealcat

mealcat -----------------------------------------------------------
(unlabeled)
                  type:  numeric (float)
                 label:  mealcat

                 range:  [1,3]                        units:  1
         unique values:  3                    coded missing:  0 / 400

            tabulation:  Freq.   Numeric  Label
                           131         1  0-46% free meals
                           132         2  47-80% free meals
                           137         3  81-100% free meals
```

You might be tempted to try including **mealcat** in a regression like this.

```
regress api00 mealcat

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
1207.74
       Model |  6072527.52     1  6072527.52           Prob > F      =
0.0000
    Residual |  2001144.48   398   5028.0012           R-squared     =
0.7521
-------------+------------------------------           Adj R-squared =
0.7515
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.908

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
     mealcat |  -150.5533   4.332147   -34.75   0.000    -159.0701   -
142.0365
       _cons |   950.9874   9.421798   100.93   0.000     932.4647
969.5101
------------------------------------------------------------------------
-------
```

But this is looking at the linear effect of **mealcat** with **api00**, but **mealcat** is not an interval variable. Instead, you will want to code the variable so that all the information concerning the three levels is accounted for. You can dummy code **mealcat** like this.

```
tabulate mealcat, gen(mealcat)

           mealcat |      Freq.     Percent        Cum.
-------------------+-----------------------------------
  0-46% free meals |        131       32.75       32.75
 47-80% free meals |        132       33.00       65.75
81-100% free meals |        137       34.25      100.00
-------------------+-----------------------------------
             Total |        400      100.00
```

We now have created **mealcat1** that is 1 if **mealcat** is 1, and 0 otherwise. Likewise, **mealcat2** is 1 if **mealcat** is 2, and 0 otherwise and likewise **mealcat3** was created. We can see this below.

```
list mealcat mealcat1 mealcat2 mealcat3 in 1/10, nolabel

                mealcat  mealcat1  mealcat2  mealcat3
  1.                  1         1         0         0
  2.                  2         0         1         0
  3.                  3         0         0         1
  4.                  1         1         0         0
  5.                  1         1         0         0
  6.                  1         1         0         0
  7.                  1         1         0         0
  8.                  1         1         0         0
  9.                  1         1         0         0
 10.                  1         1         0         0
```

We can now use two of these dummy variables (**mealcat2** and **mealcat3**) in the regression analysis.

```
regress api00 mealcat2 mealcat3

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
       Model |  6094197.67     2  3047098.83           Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143           R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.612

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    mealcat2 |  -166.3236   8.708331   -19.10   0.000    -183.4438   -
149.2034
    mealcat3 |   -301.338   8.628815   -34.92   0.000    -318.3019   -
284.3741
```

```
       _cons |   805.7176   6.169416   130.60   0.000     793.5887
817.8464
------------------------------------------------------------------------
-------
```

We can test the overall differences among the three groups by using the **test** command as shown below. This shows that the overall differences among the three groups are significant.

```
test mealcat2 mealcat3

 ( 1)  mealcat2 = 0.0
 ( 2)  mealcat3 = 0.0

       F(  2,   397) =  611.12
            Prob > F =    0.0000
```

The interpretation of the coefficients is much like that for the binary variables. Group 1 is the omitted group, so **_cons** is the mean for group 1. The coefficient for **mealcat2** is the mean for group 2 minus the mean of the omitted group (group 1). And the coefficient for **mealcat3** is the mean of group 3 minus the mean of group 1. You can verify this by comparing the coefficients with the means of the groups.

```
tabulate mealcat, summarize(api00)

            |         Summary of api 2000
    mealcat |        Mean   Std. Dev.       Freq.
------------+------------------------------------
  0-46% fre |   805.71756   65.668664         131
  47-80% fr |   639.39394    82.13513         132
  81-100% f |   504.37956   62.727015         137
------------+------------------------------------
      Total |    647.6225   142.24896         400
```

Based on these results, we can say that the three groups differ in their **api00** scores, and that in particular group2 is significantly different from group1 (because **mealcat2** was significant) and group 3 is significantly different from group 1 (because **mealcat3** was significant).

3.3.2 Using the xi command

We can use the **xi** command to do the work for us to create the indicator variables and run the regression all in one command, as shown below.

```
xi : regress api00 i.mealcat

i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_1
omitted)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
```

```
       Model |  6094197.67     2  3047098.83           Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143           R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.612

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
 _Imealcat_2 |  -166.3236   8.708331   -19.10   0.000    -183.4438   -
149.2034
 _Imealcat_3 |   -301.338   8.628815   -34.92   0.000    -318.3019   -
284.3741
       _cons |   805.7176   6.169416   130.60   0.000     793.5887
817.8464
------------------------------------------------------------------------
-------
```

When we use **xi** and include the term **i.mealcat** in the model, Stata creates the variables **_Imealcat_2** and **_Imealcat_3** that are dummy variables just like **mealcat2** and **mealcat3** that we created before. There really is no difference between **mealcat2** and **_Imealcat_2**.

As you can see, the results are the same as in the prior analysis. If we want to test the overall effect of **mealcat** we use the test command as shown below, which also gives us the same results as we found using the dummy variables **mealcat2** and **mealcat3**.

```
test _Imealcat_2 _Imealcat_3

 ( 1)  _Imealcat_2 = 0.0
 ( 2)  _Imealcat_3 = 0.0

       F(  2,   397) =  611.12
            Prob > F =    0.0000
```

Note that if you are doing this in Stata version 6 the variables would be named **Imealc_2** and **Imealc_3** instead of **_Imealcat_2** and **_Imealcat_3**. One of the improvements in Stata 7 is that variable names can be longer than 8 characters, so the names of the variables created by the **xi** command are easier to understand than in version 6. From this point forward, we will use the variable names that would be created in version 7.

What if we wanted a different group to be the **reference group**? If we create dummy variables via **tabulate , generate()** then we can easily choose which variable will be the omitted group, for example, let's omit group 3.

```
regress api00 mealcat1 mealcat2
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
       Model |  6094197.67     2  3047098.83           Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143           R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.612

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    mealcat1 |    301.338   8.628815    34.92   0.000     284.3741
318.3019
    mealcat2 |   135.0144    8.61209    15.68   0.000     118.0834
151.9454
       _cons |   504.3796   6.032807    83.61   0.000     492.5193
516.2398
------------------------------------------------------------------------
-------
```

With group 3 omitted, the constant is now the mean of group 3 and **mealcat1** is group1-group3 and **mealcat2** is group2-group3. We see that both of these coefficients are significant, indicating that group 1 is significantly different from group 3 and group 2 is significantly different from group 3.

When we use the **xi** command, how can we choose which group is the omitted group? By default, the first group is omitted, but say we want group 3 to be omitted. We can use the **char** command as shown below to tell Stata that we want the third group to be the omitted group for the variable **mealcat**.

```
char mealcat[omit] 3
```

Then, when we use the **xi** command using **mealcat** the mealcat=3 group will be omitted. If you save the data file, Stata will remember this for future Stata sessions.

```
xi : regress api00 i.mealcat
```

```
i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_3
omitted)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
```

```
       Model |  6094197.67     2  3047098.83           Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143           R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.612

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
 _Imealcat_1 |    301.338   8.628815    34.92   0.000     284.3741
318.3019
 _Imealcat_2 |   135.0144    8.61209    15.68   0.000     118.0834
151.9454
       _cons |   504.3796   6.032807    83.61   0.000     492.5193
516.2398
----------------------------------------------------------------------
-------
```

You can compare and see that these results are identical to those found using **mealcat1** and **mealcat2** as predictors.

3.3.3 Using the anova command

We can also do this analysis using the **anova** command. The benefit of the **anova** command is that it gives us the test of the overall effect of **mealcat** without needing to subsequently use the **test** command as we did with the **regress** command.

```
anova api00 mealcat

                           Number of obs =     400     R-squared     =
0.7548
                           Root MSE      = 70.6122     Adj R-squared =
0.7536

                  Source |  Partial SS    df       MS           F
Prob > F
              -----------+--------------------------------------------
-------
                   Model |  6094197.67     2  3047098.83     611.12
0.0000
                         |
                 mealcat |  6094197.67     2  3047098.83     611.12
0.0000
                         |
                Residual |  1979474.33   397  4986.08143
              -----------+--------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

We can see the **anova** test of the effect of **mealcat** is the same as the **test** command from the regress command.

We can even follow this with the **anova, regress** command and compare the parameter estimates with those we performed previously.

```
anova, regress

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
       Model |  6094197.67     2  3047098.83           Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143           R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.612

----------------------------------------------------------------------
-------
       api00        Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
----------------------------------------------------------------------
-------
_cons            504.3796   6.032807    83.61   0.000     492.5193
516.2398
mealcat
           1      301.338   8.628815    34.92   0.000     284.3741
318.3019
           2     135.0144    8.61209    15.68   0.000     118.0834
151.9454
           3    (dropped)
----------------------------------------------------------------------
-------
```

Note: the parameter estimates are the same because **mealcat** is coded the same way in the **regress** command and in the **anova** command, in both cases the last category (category 3) being dropped. While you can control which category is the omitted category when you use the **regress** command, the **anova, regress** command always drops the last category.

3.3.4 Other coding schemes

It is generally very convenient to use dummy coding but that is not the only kind of coding that can be used. As you have seen, when you use dummy coding one of the groups becomes the reference group and all of the other groups are compared to that group. This may not be the most interesting set of comparisons.

Say you want to compare group 1 with groups 2 and 3, and for a second comparison compare group 2 with group 3. You need to generate a coding scheme that forms these 2 comparisons. We will illustrate this using a Stata program, **xi3**, (an enhanced version of **xi**) that will create the variables you would need for such comparisons (as well as a variety of other common comparisons).

The comparisons that we have described (comparing group 1 with 2 and 3, and then comparing groups 2 and 3) correspond to Helmert comparisons (see Chapter 5 for more details). We use the **h.** prefix (instead of the **i.** prefix) to indicate that we desire Helmert comparisons on the variable mealcat. Otherwise, you see that **xi3** works much like the **xi** command.

xi3: regress api00 h.mealcat

```
h.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_3
omitted)

      Source |       SS       df       MS               Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
       Model |  6094197.67     2  3047098.83            Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143            R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669            Root MSE      =
70.612

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t     P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
 _Imealcat_1 |   233.8308   7.523544    31.08   0.000     219.0398
248.6218
 _Imealcat_2 |   135.0144    8.61209    15.68   0.000     118.0834
151.9454
       _cons |   649.8304   3.531285   184.02   0.000      642.888
656.7727
----------------------------------------------------------------------
-----
```

If you compare the parameter estimates with the means (see below) you can verify that the coefficient for **_Imealcat_1** is the mean of group 1 minus the mean of groups 2 and 3 (805.71756 - (639.39394 + 504.37956) / 2 = 233.83081) and the coefficient for **_Imealcat_2** is the mean of group 2 minus group 3 (639.39 - 504.37 = 135.01). Both of these comparisons are significant, indicating that group 1 differs significantly from groups 2 and 3 combined, and group 2 differs significantly from group 3.

tabulate mealcat, sum(api00)

```
            |         Summary of api 2000
    mealcat |        Mean   Std. Dev.       Freq.
```

```
------------+-----------------------------------
  0-46% fre |   805.71756   65.668664         131
  47-80% fr |   639.39394    82.13513         132
  81-100% f |   504.37956   62.727015         137
------------+-----------------------------------
      Total |    647.6225   142.24896         400
```

And the value of **_cons** is the unweighted average of the means of the 3 groups.

```
display (805.71756 +639.39394 +504.37956)/3
649.83035
```

Using the coding scheme provided by **xi3**, we were able to form perhaps more interesting tests than those provided by dummy coding. The **xi3** program can create variables according to other coding schemes, as well as custom coding schemes that you create, see **help xi3** and Chapter 5 for more information.

3.4 Regression with two categorical predictors

3.4.1 Using the xi: command

Previously we looked at using **yr_rnd** to predict **api00**

```
regress api00 yr_rnd
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
116.24
       Model |  1825000.56     1  1825000.56           Prob > F      =
0.0000
    Residual |  6248671.43   398  15700.1795           R-squared     =
0.2260
-------------+------------------------------           Adj R-squared =
0.2241
       Total |  8073672.00   399  20234.7669           Root MSE      =
125.30

-----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------
-------
      yr_rnd |  -160.5064    14.8872   -10.78   0.000    -189.7737    -
131.239
       _cons |    684.539    7.13965    95.88   0.000     670.5028
698.5751
-----------------------------------------------------------------------
-------
```

And we have also looked at **mealcat** using the **xi** command

```
xi : regress api00 i.mealcat
```

```
i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_3
omitted)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
611.12
       Model |  6094197.67     2  3047098.83           Prob > F      =
0.0000
    Residual |  1979474.33   397  4986.08143           R-squared     =
0.7548
-------------+------------------------------           Adj R-squared =
0.7536
       Total |  8073672.00   399  20234.7669           Root MSE      =
70.612

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
 _Imealcat_1 |    301.338   8.628815    34.92   0.000     284.3741
318.3019
 _Imealcat_2 |   135.0144    8.61209    15.68   0.000     118.0834
151.9454
       _cons |   504.3796   6.032807    83.61   0.000     492.5193
516.2398
------------------------------------------------------------------------
-------
```

We can include both **yr_rnd** and **mealcat** together in the same model.

```
xi : regress api00 i.mealcat yr_rnd

i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_3
omitted)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
435.02
       Model |  6194144.30     3  2064714.77           Prob > F      =
0.0000
    Residual |  1879527.69   396  4746.28206           R-squared     =
0.7672
-------------+------------------------------           Adj R-squared =
0.7654
       Total |  8073672.00   399  20234.7669           Root MSE      =
68.893

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
```

```
------------+------------------------------------------------------------
-------
 _Imealcat_1 |   281.6832   9.445676    29.82   0.000     263.1132    
300.2531
 _Imealcat_2 |   117.9458   9.188911    12.84   0.000     99.88066     
136.011
      yr_rnd |  -42.96006   9.361761    -4.59   0.000    -61.36502   -
24.55509
       _cons |     526.33   7.584533    69.40   0.000      511.419    
541.2409
-------------------------------------------------------------------------
-------
```

We can test the overall effect of **mealcat** with the test command, which is significant.

```
test  _Imealcat_1 _Imealcat_2

 ( 1)  _Imealcat_1 = 0.0
 ( 2)  _Imealcat_2 = 0.0

       F(  2,   396) =  460.27
            Prob > F =    0.0000
```

Because this model has only main effects (no interactions) you can interpret **Byr_rnd** as the difference between the year round and non-year round group. The coefficient for **I_mealcat_1** (which we will call **B_Imealcat_1**) is the difference between mealcat=1 and mealcat=3, and **B_Imealcat_2** as the difference between mealcat=2 and mealcat=3.

Let's dig below the surface and see how the coefficients relate to the predicted values. Let's view the cells formed by crossing **yr_rnd** and **mealcat** and number the cells from cell1 to cell6.

```
          mealcat=1     mealcat=2      mealcat=3
 yr_rnd=0  cell1         cell2          cell3
 yr_rnd=1  cell4         cell5          cell6
```

With respect to **mealcat**, the group **mealcat=3** is the reference category, and with respect to **yr_rnd** the group **yr_rnd=0** is the reference category. As a result, cell3 is the reference cell. The constant is the predicted value for this cell.

The coefficient for **yr_rnd** is the difference between **cell3** and **cell6**. Since this model has only main effects, it is also the difference between cell2 and cell5, or from cell1 and cell4. In other words, **Byr_rnd** is the amount you add to the predicted value when you go from non-year round to year round schools.

The coefficient for **_Imealcat_1** is the predicted difference between cell1 and cell3. Since this model only has main effects, it is also the predicted difference between cell4 and cell6. Likewise, **B_Imealcat_2** is the predicted difference between cell2 and cell3, and also the predicted difference between cell5 and cell6.

So, the predicted values, in terms of the coefficients, would be

```
          mealcat=1         mealcat=2         mealcat=3
         ----------------------------------------------
yr_rnd=0  _cons             _cons             _cons
          +BImealcat1       +BImealcat2
         ----------------------------------------------
yr_rnd=1  _cons             _cons             _cons
          +Byr_rnd          +Byr_rnd          +Byr_rnd
          +BImealcat1       +BImealcat2
```

We should note that if you computed the predicted values for each cell, they would not exactly match the means in the 6 cells. The predicted means would be close to the observed means in the cells, but not exactly the same. This is because our model only has main effects and assumes that the difference between cell1 and cell4 is exactly the same as the difference between cells 2 and 5 which is the same as the difference between cells 3 and 6. Since the observed values don't follow this pattern, there is some discrepancy between the predicted means and observed means.

3.4.2 Using the anova command

We can run the same analysis using the **anova** command with just main effects

```
anova api00 yr_rnd mealcat

                           Number of obs =     400     R-squared     =
0.7672
                           Root MSE      = 68.8933     Adj R-squared =
0.7654

                  Source |  Partial SS    df       MS           F
Prob > F
              -----------+-----------------------------------------------
-------
                   Model |  6194144.30     3  2064714.77     435.02
0.0000
                         |
                  yr_rnd |  99946.6332     1  99946.6332      21.06
0.0000
                 mealcat |  4369143.74     2  2184571.87     460.27
0.0000
                         |
                Residual |  1879527.69   396  4746.28206
              -----------+-----------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

Note that we get the same information that we do from the **xi : regress** command, followed by the **test** command. The **anova** command automatically provides the information provided by the **test** command. If we like, we can also request the parameter estimates later just by doing this.

```
anova, regress

      Source |       SS       df       MS              Number of obs =
400
```

```
-------------+------------------------------           F(  3,   396) =
435.02
       Model |  6194144.30     3  2064714.77           Prob > F      =
0.0000
    Residual |  1879527.69   396  4746.28206           R-squared     =
0.7672
-------------+------------------------------           Adj R-squared =
0.7654
       Total |  8073672.00   399  20234.7669           Root MSE      =
68.893

----------------------------------------------------------------------
-------
       api00        Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
----------------------------------------------------------------------
-------
_cons            483.3699    7.45694    64.82   0.000     468.7098
498.03
yr_rnd
          1      42.96006   9.361761     4.59   0.000     24.55509
61.36502
          2    (dropped)
mealcat
          1      281.6832   9.445676    29.82   0.000     263.1132
300.2531
          2      117.9458   9.188911    12.84   0.000     99.88066
136.011
          3    (dropped)
----------------------------------------------------------------------
-------
```

anova will display the parameter estimates from the last anova model. However, the **anova** command is rigid in its determination of which group will be the omitted group and the last group is dropped. Since this differs from the coding we used in the regression commands above, the parameter estimates from this anova command will differ from the regress command above.

In summary, these results indicate the differences between year round and non-year round schools is significant, and the differences among the three **mealcat** groups are significant.

3.5 Categorical predictor with interactions

3.5.1 using xi

Let's perform the same analysis that we performed above, this time let's include the interaction of **mealcat** by **yr_rnd**. When using **xi**, it is easy to include an interaction term, as shown below.

```
xi : regress api00 i.mealcat*yr_rnd

i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_3
omitted)
i.meal~t*yr_rnd   _ImeaXyr_rn_#       (coded as above)
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  5,   394) =
261.61
       Model |  6204727.82     5  1240945.56           Prob > F      =
0.0000
    Residual |  1868944.18   394  4743.51314           R-squared     =
0.7685
-------------+------------------------------           Adj R-squared =
0.7656
       Total |  8073672.00   399  20234.7669           Root MSE      =
68.873

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
 _Imealcat_1 |   288.1929   10.44284    27.60   0.000     267.6623
308.7236
 _Imealcat_2 |    123.781   10.55185    11.73   0.000      103.036
144.5259
      yr_rnd |  -33.49254   11.77129    -2.85   0.005    -56.63492   -
10.35015
_ImeaXyr_r~1 |  -40.76438   29.23118    -1.39   0.164    -98.23297
16.70422
_ImeaXyr_r~2 |  -18.24763   22.25624    -0.82   0.413    -62.00347
25.5082
       _cons |   521.4925   8.414197    61.98   0.000     504.9502
538.0349
------------------------------------------------------------------------
-------
```

We can test the overall interaction with the **test** command. This interaction effect is not significant.

```
test  _ImeaXyr_rn_1 _ImeaXyr_rn_2

 ( 1)  _ImeaXyr_rn_1 = 0.0
 ( 2)  _ImeaXyr_rn_2 = 0.0

       F(  2,   394) =    1.12
            Prob > F =    0.3288
```

It is important to note how the meaning of the coefficients change in the presence of these interaction terms. For example, in the prior model, with only main effects, we could interpret **Byr_rnd** as the difference between the year round and non year round schools. However, now that we have added the interaction term, the term **Byr_rnd** represents the difference between cell3 and cell6, or the difference between the year round and non-year round schools when **mealcat**=3 (because **mealcat**=3 was the omitted group). The presence of an interaction would imply that the difference between year round and non-year round schools depends on the level of **mealcat**. The interaction terms **B_ImeaXyr_rn_1** and **B_ImeaXyr_rn_2** represent the extent to which the difference between the year round/non year round schools changes when mealcat=1 and when mealcat=2 (as compared to the reference group, mealcat=3). For example the term **B_ImeaXyr_rn_1** represents the difference between year round and non-year round for mealcat=1 vs. the difference for mealcat=3. In other words, **B_ImeaXyr_rn_1** in this design is (cell1-cell4) - (cell3-cell6), or it represents how much the effect of **yr_rnd** differs between mealcat=1 and mealcat=3.

Below we have shown the predicted values for the six cells in terms of the coefficients in the model. If you compare this to the main effects model, you will see that the predicted values are the same except for the addition of **_ImeaXyr_rn_1** (in cell 4) and **_ImeaXyr_rn_2** (in cell 5).

```
           mealcat=1           mealcat=2          mealcat=3
           -------------------------------------------------
 yr_rnd=0  _cons               _cons              _cons
           +BImealcat1         +BImealcat2
           -------------------------------------------------
 yr_rnd=1  _cons               _cons              _cons
           +Byr_rnd            +Byr_rnd           +Byr_rnd
           +BImealcat1         +BImealcat2
           +B_ImeaXyr_rn_1     +B_ImeaXyr_rn_2
```

It can be very tricky to interpret these interaction terms if you wish to form specific comparisons. For example, if you wanted to perform a test of the simple main effect of **yr_rnd** when **mealcat**=1, i.e., comparing cell1 with cell4, you would want to compare **_cons**+ **BImealcat1** vs. **_cons** + **B yr_rnd + BImealcat1+ BImeaXyr_rn_1** and since **_cons** and **Imealcat1** would drop out, we would test

```
test _b[yr_rnd] +_b[_ImeaXyr_rn_1] ==0

 ( 1)  yr_rnd + _ImeaXyr_rn_1 = 0.0

       F(  1,   394) =    7.70
            Prob > F =    0.0058
```

This test is significant, indicating that the effect of **yr_rnd** is significant for the **mealcat** = 1 group.

As we will see, such tests can be more easily done via anova.

3.5.2 Using anova

Constructing these interactions can be somewhat easier when using the **anova** command. As you see below, the **anova** command gives us the test of the overall main effects and interactions without the need to perform subsequent **test** commands.

```
anova api00 yr_rnd mealcat yr_rnd*mealcat

                           Number of obs =     400     R-squared     =
0.7685
                           Root MSE      = 68.8732     Adj R-squared =
0.7656
```

```
                  Source |  Partial SS    df       MS           F
Prob > F
              -----------+------------------------------------------
-------
                   Model |  6204727.82     5  1240945.56     261.61
0.0000
                         |
                  yr_rnd |  99617.3706     1  99617.3706      21.00
0.0000
                 mealcat |  1796232.80     2  898116.399     189.34
0.0000
          yr_rnd*mealcat |  10583.5187     2  5291.75936       1.12
0.3288
                         |
                Residual |  1868944.18   394  4743.51314
              -----------+------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

It is easy to perform tests of simple main effects using the **sme** command. You can download **sme** from within Stata by typing **findit sme** (see How can I used the findit command to search for programs and get additional help? for more information about using **findit**).

Now we can test the simple main effects of **yr_rnd** at each level of **mealcat**.

```
sme yr_rnd mealcat

Test of yr_rnd at mealcat(1): F(1/394)  = 7.7023296
Test of yr_rnd at mealcat(2): F(1/394)  = 7.5034121
Test of yr_rnd at mealcat(3): F(1/394)  = 8.0955856


Critical value of F for alpha = .05 using ...
--------------------------------------------------
Dunn's procedure               = 4.7435944
Marascuilo & Levin             = 5.4561926
per family error rate          = 5.7804
simultaneous test procedure    = 8.1680324
```

The results from **sme** show us the effect of **yr_rnd** at each of the 3 levels of **mealcat**. We can see that the comparison for **mealcat** = 1 matches those we computed above using the test statement, however, it was much easier and less error prone using the **sme** command.

Although this section has focused on how to handle analyses involving interactions, these particular results show no indication of interaction. We could decide to omit interaction terms from future analyses having found the interactions to be non-significant. This would simplify future analyses, however including the interaction term can be useful to assure readers that the interaction term is non-significant.

3.6 Continuous and Categorical variables

3.6.1 Using regress

Say that we wish to analyze both continuous and categorical variables in one analysis. For example, let's include **yr_rnd** and **some_col** in the same analysis.

```
regress api00 yr_rnd some_col
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
68.54
       Model |  2072201.84     2  1036100.92           Prob > F      =
0.0000
    Residual |  6001470.16   397  15117.0533           R-squared     =
0.2567
-------------+------------------------------           Adj R-squared =
0.2529
       Total |  8073672.00   399  20234.7669           Root MSE      =
122.95

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      yr_rnd |  -149.1591   14.87519   -10.03   0.000    -178.4031   -
119.9151
    some_col |   2.235689   .5528656     4.04   0.000     1.148779
3.322599
       _cons |   637.8581   13.50332    47.24   0.000     611.3111
664.405
------------------------------------------------------------------------
-------
```

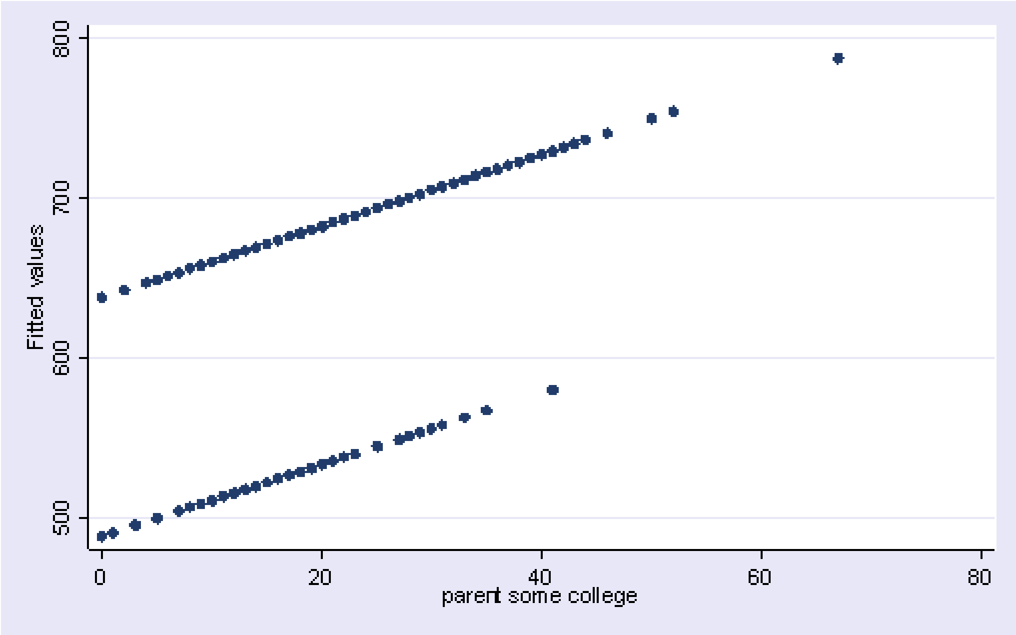
We can create the predicted values using the **predict** command.

```
predict yhat
```

```
(option xb assumed; fitted values)
```

Let's graph the predicted values by **some_col**.

```
scatter yhat some_col
```



The coefficient for **some_col** indicates that for every unit increase in **some_col** the **api00** score is predicted to increase by 2.23 units. This is the slope of the lines shown in the above graph. The graph has two lines, one for the year round schools and one for the non-year round schools. The coefficient for **yr_rnd** is -149.16, indicating that as **yr_rnd** increases by 1 unit, the **api00** score is expected to decrease by about 149 units. As you can see in the graph, the top line is about 150 units higher than the lower line. You can see that the intercept is 637 and that is where the upper line crosses the Y axis when X is 0. The lower line crosses the line about 150 units lower at about 487.

3.6.2 Using anova

We can run this analysis using the **anova** command. The **anova** command assumes that the variables are categorical, thus, we need to use the **continuous()** option (which can be abbreviated as **cont()**) to specify that **some_col** is a continuous variable.

```
anova api00 yr_rnd some_col, cont(some_col)

                           Number of obs =     400     R-squared     =
0.2567
                           Root MSE      = 122.951     Adj R-squared =
0.2529

                  Source |  Partial SS    df       MS           F
Prob > F
              -----------+----------------------------------------------------
-------
                   Model |  2072201.84     2  1036100.92      68.54
0.0000
                         |
                  yr_rnd |  1519992.67     1  1519992.67     100.55
0.0000
```

```
                some_col |  247201.276     1  247201.276      16.35
0.0001
                         |
                Residual |  6001470.16   397  15117.0533
              -----------+----------------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

If we square the t-values from the **regress** command (above), we would find that they match those of the **anova** command.

3.7 Interactions of Continuous by 0/1 Categorical variables

Above we showed an analysis that looked at the relationship between **some_col** and **api00** and also included **yr_rnd**. We saw that this produced a graph where we saw the relationship between **some_col** and **api00** but there were two regression lines, one higher than the other but with equal slope. Such a model assumed that the slope was the same for the two groups. Perhaps the slope might be different for these groups. Let's run the regressions separately for these two groups beginning with the non-year round schools.

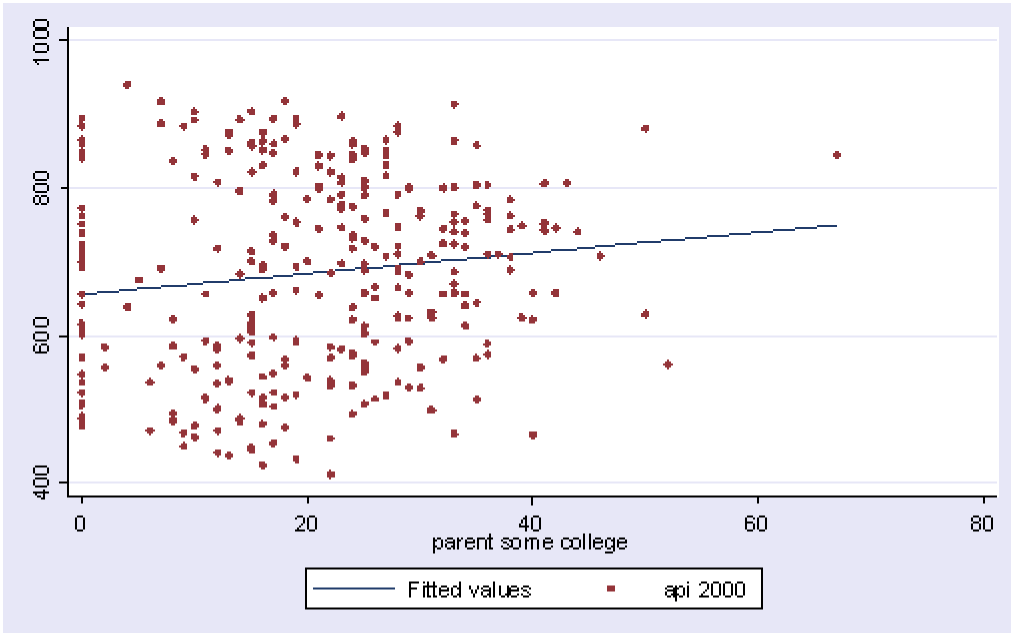
regress api00 some_col if yr_rnd==0

```
      Source |       SS       df       MS              Number of obs =
308
-------------+------------------------------           F(  1,   306) =
4.91
       Model |  84700.8576     1  84700.8576           Prob > F      =
0.0274
    Residual |  5273591.67   306  17233.9597           R-squared     =
0.0158
-------------+------------------------------           Adj R-squared =
0.0126
       Total |  5358292.53   307  17453.7216           Root MSE      =
131.28

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    some_col |   1.409427   .6357572     2.22   0.027     .1584181
2.660436
       _cons |   655.1103   15.23704    42.99   0.000     625.1277
685.0929
------------------------------------------------------------------------
-------
```

predict yhat0 if yr_rnd==0

```
(option xb assumed; fitted values)
(92 missing values generated)
```

scatter yhat0 api00 some_col if yr_rnd==0, connect(l i) msymbol(i o) sort

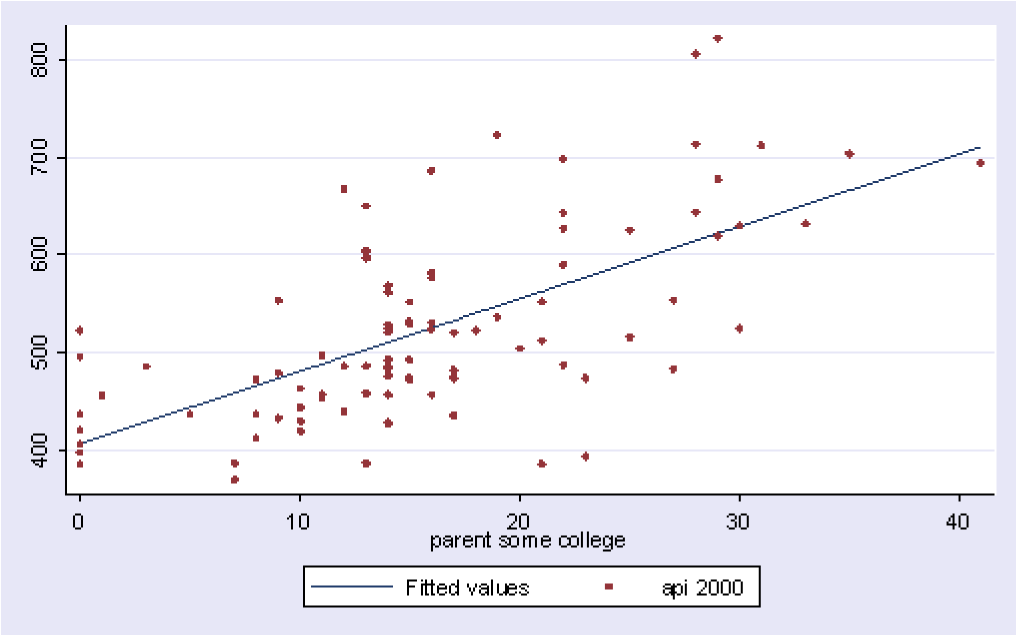


Likewise, let's look at the year round schools.

regress api00 some_col if yr_rnd==1

```
      Source |       SS       df       MS              Number of obs =
92
-------------+------------------------------           F(  1,    90) =
65.08
       Model |  373644.064     1  373644.064           Prob > F      =
0.0000
    Residual |  516734.838    90   5741.4982           R-squared     =
0.4196
-------------+------------------------------           Adj R-squared =
0.4132
       Total |  890378.902    91  9784.38354           Root MSE      =
75.773

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    some_col |   7.402618   .9176327     8.07   0.000      5.57958
9.225655
       _cons |   407.0391   16.51462    24.65   0.000     374.2299
439.8482
------------------------------------------------------------------------
-------
```

```
predict yhat1 if yr_rnd==1

(option xb assumed; fitted values)
(308 missing values generated)
scatter yhat1 api00 some_col if yr_rnd==1, connect(l i) msymbol(i o)
sort
```



Note that the slope of the regression line looks much steeper for the year round schools than for the non-year round schools. This is confirmed by the regression equations that show the slope for the year round schools to be higher (7.4) than non-year round schools (1.3). We can compare these to see if these are significantly different from each other by including the interaction of **some_col** by **yr_rnd**, an interaction of a continuous variable by a categorical variable.

3.7.1 Computing interactions manually

We will start by manually computing the interaction of **some_col** by **yr_rnd**. Let's start fresh and use the **elemapi2** data file using the **, clear** option to clear out any variables we have previously created.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
```

Next, let's make a variable that is the interaction of some college (**some_col**) and year round schools (**yr_rnd**) called **yrXsome**.

```
gen yrXsome = yr_rnd*some_col
```

We can now run the regression that tests whether the coefficient for **some_col** is significantly different for year round schools and non-year round schools. Indeed, the **yrXsome** interaction effect is significant.

```
regress api00 some_col yr_rnd yrXsome

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
52.05
       Model |  2283345.48     3  761115.162           Prob > F      =
0.0000
    Residual |  5790326.51   396  14622.0366           R-squared     =
0.2828
-------------+------------------------------           Adj R-squared =
0.2774
       Total |  8073672.00   399  20234.7669           Root MSE      =
120.92

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    some_col |   1.409427   .5856022     2.41   0.017     .2581494
2.560705
      yr_rnd |  -248.0712   29.85895    -8.31   0.000    -306.7731   -
189.3694
     yrXsome |    5.99319    1.57715     3.80   0.000     2.892557
9.093824
       _cons |   655.1103   14.03499    46.68   0.000     627.5179
682.7027
------------------------------------------------------------------------
-------
```

We can make a graph showing the regression lines for the two types of schools showing how different their regression lines are. We first create the predicted value, we call it **yhata**.

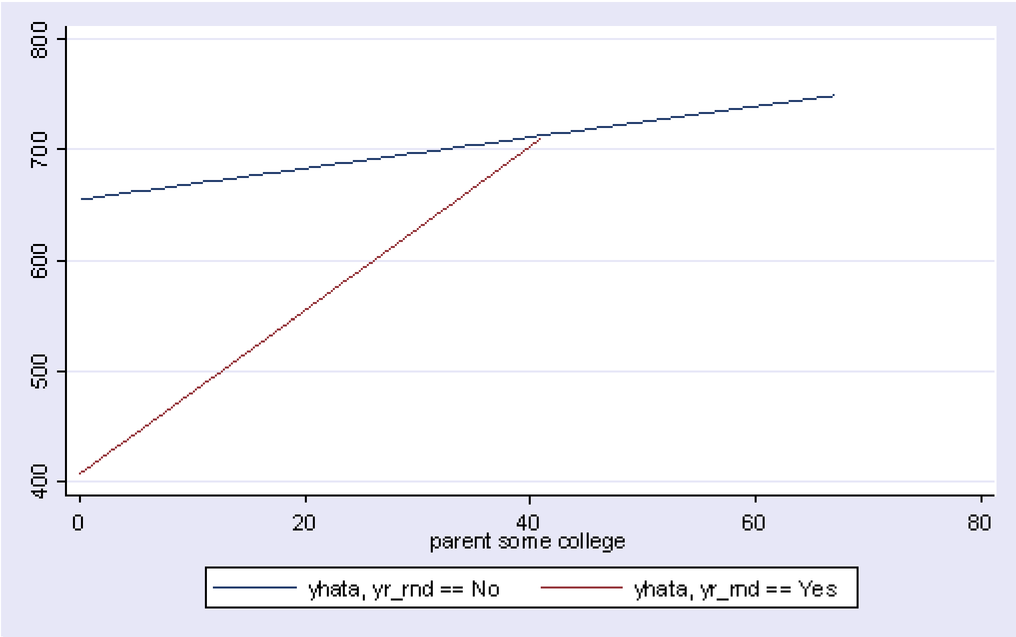
```
predict yhata

(option xb assumed; fitted values)
```

Then, we create separate variables for the two types of schools which will be called **yhata0** for non-year round schools and **yhata1** for year round schools.

```
separate yhata, by(yr_rnd)

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------
--------
yhata0          float  %9.0g                  yhata, yr_rnd == 0
yhata1          float  %9.0g                  yhata, yr_rnd == 1
```

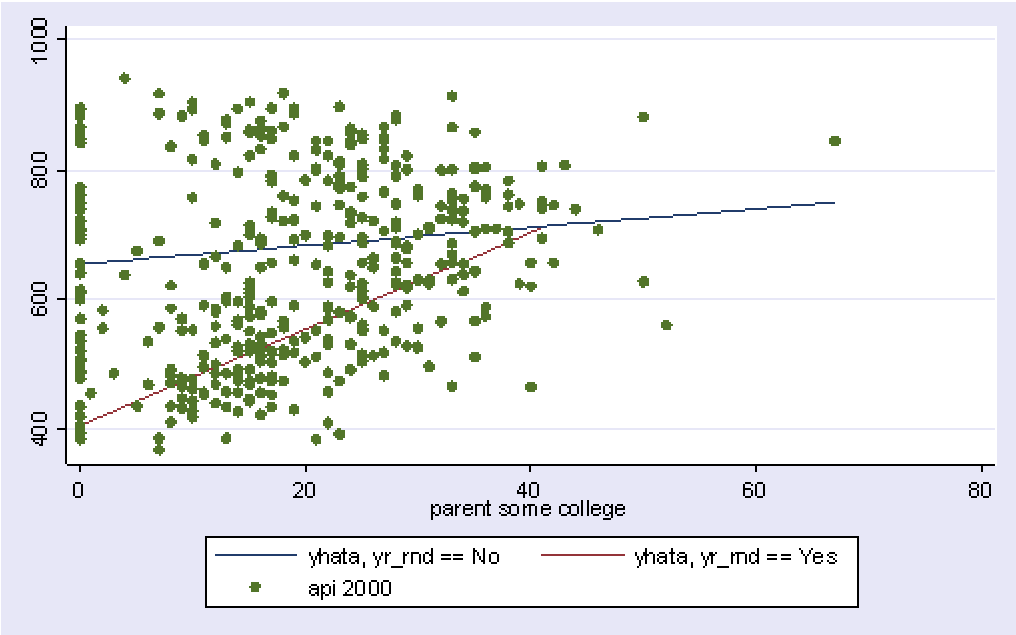
We can then graph the predicted values for the two types of schools by **some_col**. You can see how the two lines have quite different slopes, consistent with the fact that the **yrXsome** interaction was significant. The **c(ll[_])** option indicates that **yhata0** should be connected with a line, and **yhata1** should be connected with dashed lines (because we included **[_]** after the **l**). If we had used **l[.]** it would have made a dotted line. The options to make dashed and dotted lines are new to Stata 7 and you can find more information via **help grsym** .

```
line yhata0 yhata1 some_col, sort
```



We can replot the same graph including the data points.

```
twoway (line yhata0 yhata1 some_col, sort) (scatter api00 some_col)
```

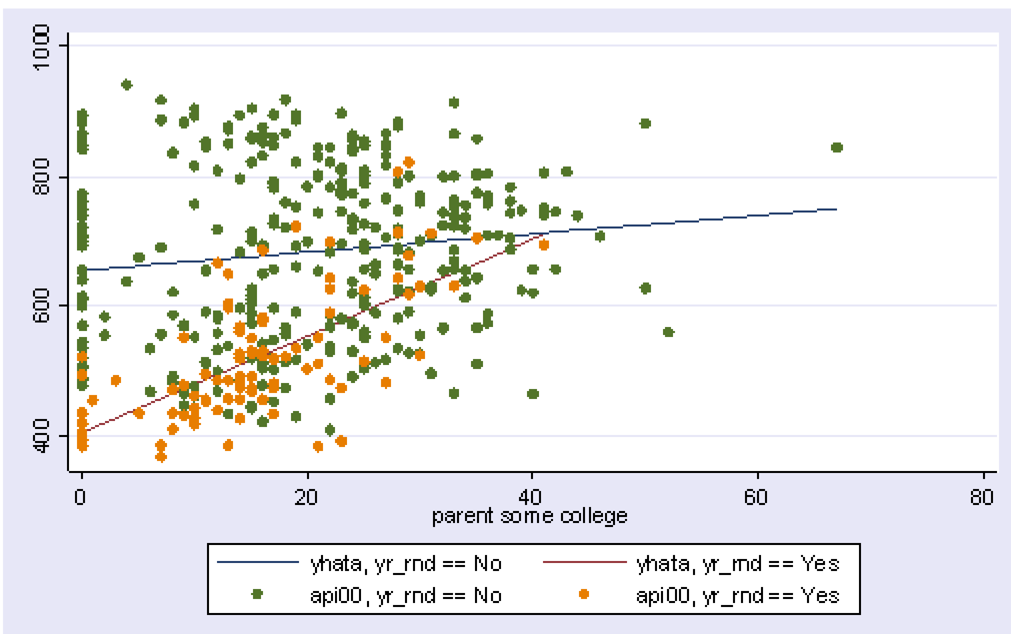


The graph above used the same kind of dots for the data points for both types of schools. Let's make separate variables for the **api00** scores for the two types of schools called **api000** for the non-year round schools and **api001** for the year round schools.

```
separate api00, by(yr_rnd)

                  storage  display     value
variable name       type   format      label      variable label
-----------------------------------------------------------------------
--------
api000              int    %6.0g                  api00, yr_rnd == 0
api001              int    %6.0g                  api00, yr_rnd == 1
```

We can then make the same graph as above except show the points differently for the two types of schools. Below we use small circles for the non-year round schools, and triangles for the year round schools.

```
twoway (line yhata0 yhata1 some_col, sort) (scatter api000 api001
some_col)
```



Let's quickly run the regressions again where we performed separate regressions for the two groups

Non-year round

```
regress api00 some_col if yr_rnd==0

      Source |       SS       df       MS              Number of obs =
308
-------------+------------------------------           F(  1,   306) =
4.91
```

```
       Model |  84700.8576     1  84700.8576           Prob > F      =
0.0274
    Residual |  5273591.67   306  17233.9597           R-squared     =
0.0158
-------------+------------------------------           Adj R-squared =
0.0126
       Total |  5358292.53   307  17453.7216           Root MSE      =
131.28

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
    some_col |   1.409427   .6357572     2.22   0.027     .1584181
2.660436
       _cons |   655.1103   15.23704    42.99   0.000     625.1277
685.0929
----------------------------------------------------------------------
-------
```

Year round

```
regress api00 some_col if yr_rnd==1
```

```
      Source |       SS       df       MS              Number of obs =
92
-------------+------------------------------           F(  1,    90) =
65.08
       Model |  373644.064     1  373644.064           Prob > F      =
0.0000
    Residual |  516734.838    90   5741.4982           R-squared     =
0.4196
-------------+------------------------------           Adj R-squared =
0.4132
       Total |  890378.902    91  9784.38354           Root MSE      =
75.773

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
    some_col |   7.402618   .9176327     8.07   0.000      5.57958
9.225655
       _cons |   407.0391   16.51462    24.65   0.000     374.2299
439.8482
----------------------------------------------------------------------
-------
```

Now, let's show the regression for both types of schools with the interaction term.

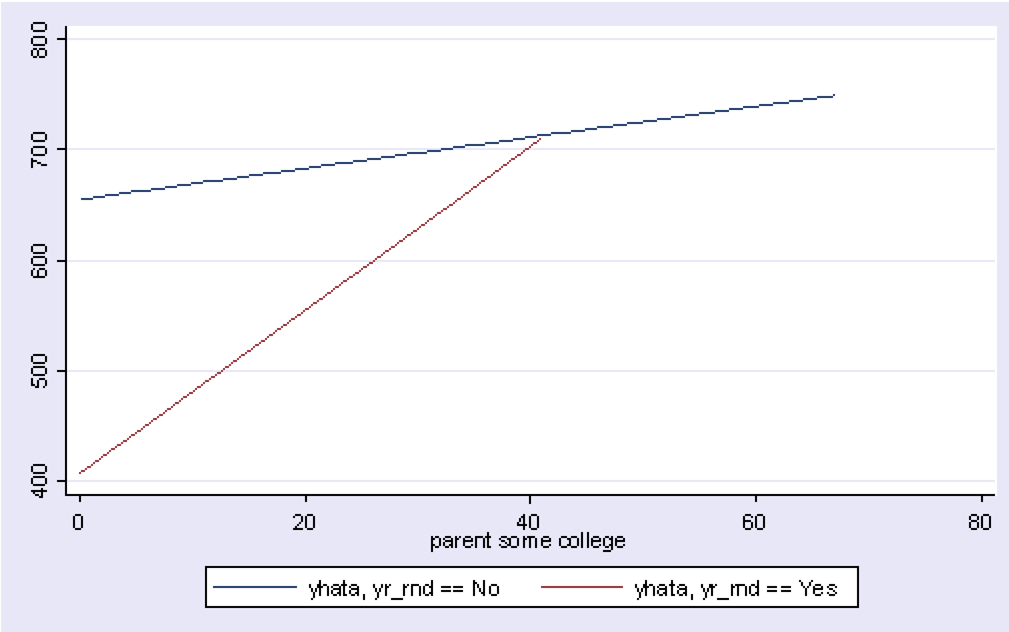
```
regress api00 some_col yr_rnd yrXsome
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
52.05
       Model |  2283345.48     3  761115.162           Prob > F      =
0.0000
    Residual |  5790326.51   396  14622.0366           R-squared     =
0.2828
-------------+------------------------------           Adj R-squared =
0.2774
       Total |  8073672.00   399  20234.7669           Root MSE      =
120.92

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    some_col |   1.409427   .5856022     2.41   0.017     .2581494
2.560705
      yr_rnd |  -248.0712   29.85895    -8.31   0.000    -306.7731   -
189.3694
     yrXsome |    5.99319    1.57715     3.80   0.000     2.892557
9.093824
       _cons |   655.1103   14.03499    46.68   0.000     627.5179
682.7027
------------------------------------------------------------------------
-------
```

Note that the coefficient for **some_col** in the combined analysis is the same as the coefficient for **some_col** for the non-year round schools? This is because non-year round schools are the reference group. Then, the coefficient for the **yrXsome** interaction in the combined analysis is the **Bsome_col** for the year round schools (7.4) minus **Bsome_col** for the non year round schools (1.41) yielding 5.99. This interaction is the difference in the slopes of **some_col** for the two types of schools, and this is why this is useful for testing whether the regression lines for the two types of schools are equal. If the two types of schools had the same regression coefficient for **some_col**, then the coefficient for the **yrXsome** interaction would be 0. In this case, the difference is significant, indicating that the regression lines are significantly different.

So, if we look at the graph of the two regression lines we can see the difference in the slopes of the regression lines (see graph below). Indeed, we can see that the non-year round schools (the solid line) have a smaller slope (1.4) than the slope for the year round schools (7.4). The difference between these slopes is 5.99, the coefficient for **yrXsome**.

`line yhata0 yhata1 some_col, sort`



3.7.2 Computing interactions with xi

We can use the **xi** command for doing this kind of analysis as well. Let's start fresh and use the **elemapi2** file.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
```

We can run a model just like the model we showed above using the **xi** command. You can compare the results to those above and see that we get the exact same results.

```
xi : regress api00 i.yr_rnd*some_col

i.yr_rnd          _Iyr_rnd_1-2        (naturally coded; _Iyr_rnd_1
omitted)
i.yr_rnd*some~l   _Iyr_Xsome__#       (coded as above)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
52.05
       Model |  2283345.48     3  761115.162           Prob > F      =
0.0000
    Residual |  5790326.51   396  14622.0366           R-squared     =
0.2828
-------------+------------------------------           Adj R-squared =
0.2774
       Total |  8073672.00   399  20234.7669           Root MSE      =
120.92

----------------------------------------------------------------------
-------
```

```
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-------
  _Iyr_rnd_2 |  -248.0712   29.85895    -8.31   0.000    -306.7731   -
189.3694
    some_col |   1.409427   .5856022     2.41   0.017     .2581494
2.560705
_Iyr_Xsome~2 |    5.99319    1.57715     3.80   0.000     2.892557
9.093824
       _cons |   655.1103   14.03499    46.68   0.000     627.5179
682.7027
------------------------------------------------------------------------------
-------
```

The **i.yr_rnd*some_col** term creates 3 terms, **some_col**, **_Iyr_rnd_2** an indicator variable for yr_rnd representing whether the school is year round and the variable **_Iyr_Xsome~2** representing the interaction of **yr_rnd** by **some_col**.

As we did above, we can create predicted values and create graphs showing the regression lines for the two types of schools. We omit showing these commands.

3.7.3 Computing interactions with anova

We can also run a model just like the model we showed above using the **anova** command. We include the terms **yr_rnd some_col** and the interaction **yr_rnr*some_col**

```
anova api00 yr_rnd some_col yr_rnd*some_col, contin(some_col)

                           Number of obs =     400     R-squared     =
0.2828
                           Root MSE      = 120.922     Adj R-squared =
0.2774

                  Source |  Partial SS    df       MS           F
Prob > F
         ----------------+--------------------------------------------
-------
                   Model |  2283345.48     3  761115.162      52.05
0.0000
                         |
                  yr_rnd |  1009279.99     1  1009279.99      69.02
0.0000
                some_col |  456473.187     1  456473.187      31.22
0.0000
         yr_rnd*some_col |  211143.646     1  211143.646      14.44
0.0002
                         |
                Residual |  5790326.51   396  14622.0366
         ----------------+--------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

As we illustrated above, we can compute the predicted values using the predict command and graph the separate regression lines. These commands are omitted.

In this section we found that the relationship between **some_col** and **api00** depended on whether the school is a year round school or a non-year round school. For the year round schools, the relationship between **some_col** and **api00** was significantly stronger than for non-year round schools. In general, this type of analysis allows you to test whether the strength of the relationship between two continuous variables varies based on the categorical variable.

3.8 Continuous and Categorical variables, interaction with 1/2/3 variable

The prior examples showed how to do regressions with a continuous variable and a categorical variable that has 2 levels. These examples will extend this further by using a categorical variable with 3 levels, **mealcat**.

3.8.1 using xi

We can use the **xi** command to run a model with **some_col**, **mealcat** and the interaction of these two variables.

```
xi : regress api00 i.mealcat*some_col

i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_1
omitted)
i.meal~t*some~l   _ImeaXsome__#       (coded as above)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  5,   394) =
263.00
       Model |  6212306.88     5  1242461.38           Prob > F      =
0.0000
    Residual |  1861365.12   394  4724.27696           R-squared     =
0.7695
-------------+------------------------------           Adj R-squared =
0.7665
       Total |  8073672.00   399  20234.7669           Root MSE      =
68.733

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
 _Imealcat_2 |    -239.03   18.66502   -12.81   0.000    -275.7255   -
202.3345
 _Imealcat_3 |  -344.9476   17.05743   -20.22   0.000    -378.4825   -
311.4126
    some_col |  -.9473385   .4873679    -1.94   0.053    -1.905505
.0108284
```

```
_ImeaXsome~2 |   3.140945   .7292897     4.31   0.000     1.707159
4.57473
_ImeaXsome~3 |   2.607308   .8960435     2.91   0.004     .8456841
4.368933
       _cons |   825.8937   11.99182    68.87   0.000     802.3177
849.4697
----------------------------------------------------------------------
-------
```

The interaction now has two terms (**_ImeaXsome~2** and **_ImeaXsome~3**). To get an overall test of this interaction, we can use the **test** command.

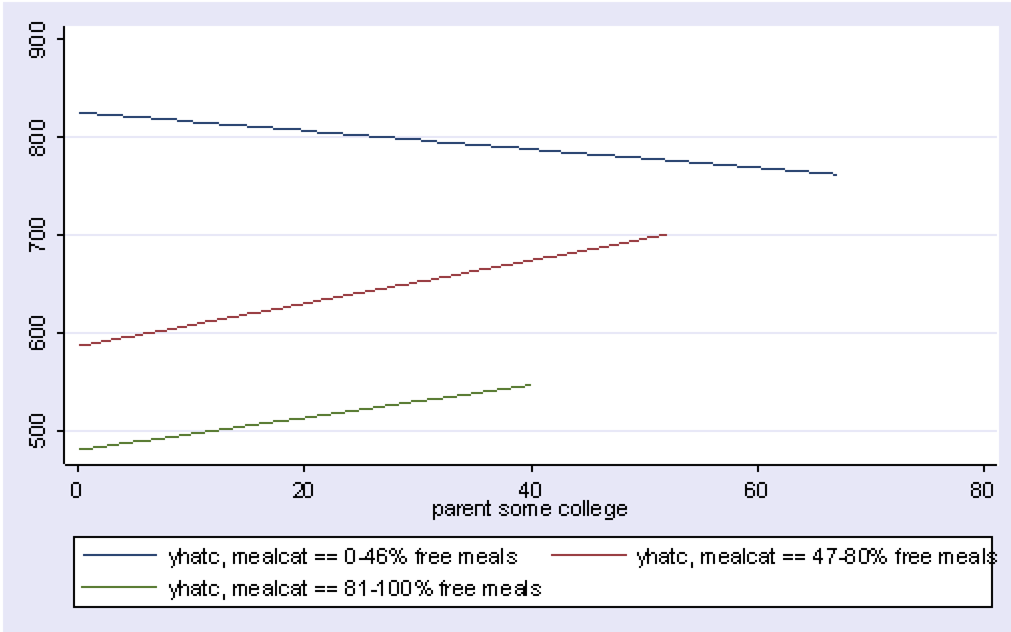
```
test  _ImeaXsome__2 _ImeaXsome__3

 ( 1)  _ImeaXsome__2 = 0.0
 ( 2)  _ImeaXsome__3 = 0.0

       F(  2,   394) =   10.32
            Prob > F =    0.0000
```

These results indicate that the overall interaction is indeed significant. This means that the regression lines from the 3 groups differ significantly. As we have done before, let's compute the predicted values and make a graph of the predicted values so we can see how the regression lines differ.

```
predict yhatc

(option xb assumed; fitted values)

separate yhatc, by(mealcat)

              storage  display     value
variable name   type   format      label      variable label
----------------------------------------------------------------------
--------
yhatc1          float  %9.0g                  yhatc, mealcat == 1
yhatc2          float  %9.0g                  yhatc, mealcat == 2
yhatc3          float  %9.0g                  yhatc, mealcat == 3
```

Since we had three groups, we get three regression lines, one for each category of **mealcat**. The solid line is for group 1, the dashed line for group 2, and the dotted line is for group 3.

```
line yhatc1 yhatc2 yhatc3 some_col, sort
```



Group 1 was the omitted group, therefore the slope of the line for group 1 is the coefficient for **some_col** which is -.94. Indeed, this line has a downward slope. If we add the coefficient for **some_col** to the coefficient for **_ImeaXsome~2** we get the coefficient for group 2, i.e., 3.14 + -.94 yields 2.2, the slope for group 2. Indeed, group 2 shows an upward slope. Likewise, if we add the coefficient for **some_col** to the coefficient for **_ImeaXsome~3** we get the coefficient for group 3, i.e., 2.6 + -.94 yields 1.66, the slope for group 3,. So, the slopes for the 3 groups are

```
group 1: -0.94
group 2:  2.2
group 3:  1.66
```

The test of the coefficient for **_ImeaXsome~2** tested whether the coefficient for group 2 differed from group 1, and indeed this was significant. Likewise, the test of the coefficient for **_ImeaXsome~3** tested whether the coefficient for group 3 differed from group 1, and indeed this was significant. What did the test of the coefficient **some_col** test? This coefficient represents the coefficient for group 1, so this tested whether the coefficient for group 1 (-0.94) was significantly different from 0. This is probably a non-interesting test.

The comparisons in the above analyses don't seem to be as interesting as comparing group 1 vs. 2 and then comparing group 2 vs. 3. These successive comparisons seem much more interesting. We can do this by making group 2 the omitted group, and then each group would be compared to group 2. As we have done before, we will use the **char** command to indicate that we want group 2 to be the omitted category and then rerun the regression.

```
char mealcat[omit] 2
xi : regress api00 i.mealcat*some_col

i.mealcat         _Imealcat_1-3       (naturally coded; _Imealcat_2
omitted)
```

```
i.meal~t*some~l   _ImeaXsome__#       (coded as above)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  5,   394) =
263.00
       Model |  6212306.88     5  1242461.38           Prob > F      =
0.0000
    Residual |  1861365.12   394  4724.27696           R-squared     =
0.7695
-------------+------------------------------           Adj R-squared =
0.7665
       Total |  8073672.00   399  20234.7669           Root MSE      =
68.733

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
 _Imealcat_1 |     239.03   18.66502    12.81   0.000     202.3345
275.7255
 _Imealcat_3 |  -105.9176    18.7545    -5.65   0.000     -142.789    -
69.0462
    some_col |   2.193606   .5425274     4.04   0.000     1.126996
3.260217
_ImeaXsome~1 |  -3.140945   .7292897    -4.31   0.000     -4.57473   -
1.707159
_ImeaXsome~3 |  -.5336362   .9272014    -0.58   0.565    -2.356517
1.289245
       _cons |   586.8637   14.30311    41.03   0.000     558.7438
614.9837
------------------------------------------------------------------------
-------
```

Now, the test of **_ImeaXsome~1** tests whether the coefficient for group 1 differs from group 2, and it does. Then, the test of **_ImeaXsome~3** tests whether the coefficient for group 3 significantly differs from group 2, and it does not. This makes sense given the graph and given the estimates of the coefficients that we have, that -.94 is significantly different from 2.2 but 2.2 is not significantly different from 1.66.

3.8.2 Using Anova

We can perform the same analysis using the **anova** command, as shown below. The **anova** command gives us somewhat less flexibility since we cannot choose which group is the omitted group.

```
use elemapi2, clear
anova api00 mealcat some_col mealcat*some_col, cont(some_col)

      Number of obs =     400     R-squared     =  0.7695
                           Root MSE      = 68.7334     Adj R-squared =
0.7665
```

```
                  Source |  Partial SS    df       MS           F
Prob > F
        -----------------+---------------------------------------------
-------
                   Model |  6212306.88     5  1242461.38     263.00
0.0000
                         |
                 mealcat |  2012065.49     2  1006032.75     212.95
0.0000
                some_col |  36366.3662     1  36366.3662       7.70
0.0058
        mealcat*some_col |  97468.1685     2  48734.0843      10.32
0.0000
                         |
                Residual |  1861365.12   394  4724.27696
        -----------------+---------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

Because the **anova** command omits the 3rd category, and the analysis we showed above omitted the second category, the parameter estimates will not be the same. You can compare the results from below with the results above and see that the parameter estimates are not the same. Because group 3 is dropped, that is the reference category and all comparisons are made with group 3.

anova, regress

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  5,   394) =
263.00
       Model |  6212306.88     5  1242461.38           Prob > F      =
0.0000
    Residual |  1861365.12   394  4724.27696           R-squared     =
0.7695
-------------+------------------------------           Adj R-squared =
0.7665
       Total |  8073672.00   399  20234.7669           Root MSE      =
68.733

------------------------------------------------------------------------
-------
       api00        Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
------------------------------------------------------------------------
-------
_cons            480.9461   12.13063    39.65   0.000     457.0973
504.795
meals3
          1      344.9476   17.05743    20.22   0.000     311.4126
378.4825
          2      105.9176    18.7545     5.65   0.000      69.0462
142.789
          3    (dropped)
```

```
some_col          1.65997   .7519086     2.21   0.028     .1817153
3.138225
meals3*some_col
        1      -2.607308   .8960435    -2.91   0.004    -4.368933   -
.8456841
        2       .5336362   .9272014     0.58   0.565    -1.289245
2.356517
        3      (dropped)
------------------------------------------------------------------------
-------
```

These analyses showed that the relationship between **some_col** and **api00** varied, depending on the level of **mealcat**. In comparing group 1 with group 2, the coefficient for **some_col** was significantly different, but there was no difference in the coefficient for **some_col** in comparing groups 2 and 3.

3.9 Summary

This covered four techniques for analyzing data with categorical variables, 1) manually constructing indicator variables, 2) creating indicator variables using the **xi** command, 3) coding variables using **xi3**, and 4) using the **anova** command. Each method has its advantages and disadvantages, as described below.

Manually constructing indicator variables can be very tedious and even error prone. For very simple models, it is not very difficult to create your own indicator variables, but if you have categorical variables with many levels and/or interactions of categorical variables, it can be laborious to manually create indicator variables. However, the advantage is that you can have quite a bit of control over how the variables are created and the terms that are entered into the model.

The **xi** command can really ease the creation of indicator variables, and make it easier to include interactions in your models by allowing you to include interaction terms such as i.prog*female . The **xi** command also gives you the flexibility to decide which category would be the omitted category (unlike the **anova** command).

The **anova** command eliminates the need to create indicator variables making it easy to include variables that have lots of categories, and making it easy to create interactions by allowing you to include terms like **some_col*mealcat**. It can be easier to perform tests of simple main effects with the **anova** command. However, the **anova** command is not flexible in letting you choose which category is the omitted category (the last category is always the omitted category).

As you will see in the next chapter, the regress command includes additional options like the **robust** option and the **cluster** option that allow you to perform analyses when you don't exactly meet the assumptions of ordinary least squares regression. In such cases, the **regress** command offers features not available in the **anova** command and may be more advantageous to use.

See the [Stata Topics: Regression] page for more information and resources on regression with categorical predictors in Stata.

3.10 Self Assessment

1. Using the **elemapi2** data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) convert the variable **ell** into 2 categories using the following coding, 0-25 on **ell** becomes 0, and 26-100 on **ell** becomes 1. Use this recoded version of **ell** to predict **api00** and interpret the results.

2. Convert the variable **ell** into 3 categories coding those scoring 0-14 on ell as 1, and those 15/41 as 2 and 42/100 as 3. Do an analysis predicting **api00** from the **ell** variable converted to a 1/2/3 variable. Interpret the results.

3. Do a regression analysis predicting **api00** from **yr_rnd** and the **ell** variable converted to a 0/1 variable. Then create an interaction term and run the analysis again. Interpret the results of these analyses.

4. Do a regression analysis predicting **api00** from **ell** coded as 0/1 (from question 1) and **some_col**, and the interaction of these two variables. Interpret the results, including showing a graph of the results.

5. Use the variable **ell** converted into 3 categories (from question 2) and predict api00 from **ell** in 3 categories, from **some_col** and the interaction. of these two variables. Interpret the results, including showing a graph.

Click here for our answers to these self assessment questions.

3.11 For more information

- Stata Manuals
  - **[R] xi**
  - **[R] anova**
  - **[R] test**
- Web Links
  - Creating Dummy Variables
    - Stata FAQ- How can I create dummy variables in Stata
  - Models with interactions of continuous and categorical variables
    - Stata FAQ- How can I compare regression coefficients between 2 groups
    - Stata FAQ- How can I compare regression coefficients across 3 (or more) groups
  - Other
    - Stata FAQ: How can I form various tests comparing the different levels of a categorical variable after anova or regress?
    - Stata FAQ- Why do estimation commands sometimes drop variables (from Stata FAQs)

Chapter 3 - Self Assessment

1. Using the **elemapi2** data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) convert the variable **ell** into 2 categories using the following coding, 0-25 on **ell** becomes 0, and 26-100 on **ell** becomes 1. Use this recoded version of **ell** to predict **api00** and interpret the results.

2. Convert the variable **ell** into 3 categories coding those scoring 0-14 on ell as 1, and those 15/41 as 2 and 42/100 as 3. Do an analysis predicting **api00** from the **ell** variable converted to a 1/2/3 variable. Interpret the results.

3. Do a regression analysis predicting **api00** from **yr_rnd** and the **ell** variable converted to a 0/1 variable. Then create an interaction term and run the analysis again. Interpret the results of these analyses.

4. Do a regression analysis predicting **api00** from **ell** coded as 0/1 (from question 1) and **some_col**, and the interaction of these two variables. Interpret the results, including showing a graph of the results.

5. Use the variable **ell** converted into 3 categories (from question 2) and predict api00 from **ell** in 3 categories, from **some_col** and the interaction. of these two variables. Interpret the results, including showing a graph.

Chapter 3: Self Assessment Answers

1. Using the **elemapi2** data file (use http://www.ats.ucla.edu/stat/stata/examples/ara/elemapi2) convert the variable **ell** into 2 categories using the following coding, 0-25 on **ell** becomes 0, and 26-100 on **ell** becomes 1. Use this recoded version of **ell** to predict **api00** and interpret the results.

Answer 1.
We first use the elemapi2 data file

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
```

We convert **ell** into a 0/1 variable called **ell_bin**.

```
gen ell_bin = ell
recode ell_bin 0/25 = 0 26/100=1
(398 changes made)
```

We tabulate **ell_bin** to see that the recoding looks OK.

```
tab ell_bin
    ell_bin |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        201       50.25       50.25
          1 |        199       49.75      100.00
------------+-----------------------------------
      Total |        400      100.00
```

We now include **ell_bin** in the regression model.

```
regress api00 ell_bin

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  1,   398) =
451.15
       Model |  4289511.71     1  4289511.71           Prob > F      =
0.0000
    Residual |  3784160.29   398  9507.94043           R-squared     =
0.5313
-------------+------------------------------           Adj R-squared =
0.5301
       Total |  8073672.00   399  20234.7669           Root MSE      =
97.509

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
```

```
     ell_bin |   -207.114   9.750989   -21.24   0.000    -226.2838   -
187.9441
       _cons |   750.6617   6.877731   109.14   0.000     737.1405
764.1829
-----------------------------------------------------------------------
-------
```

The coefficient for **_cons** represents the api scores for the schools where **ell_bin** is coded 0 (low number of English language learners). The coefficient for **ell_bin** represents the api scores for the schools with a high number of English language learners minus the api scores for the api scores for the schools with a low number of English language learners. When broken into these two categories, the schools with the high number of English language learners score 207 points lower on the api scores than schools with a low number of English language learners.

2. Convert the variable **ell** into 3 categories coding those scoring 0-14 on ell as 1, and those 15/41 as 2 and 42/100 as 3. Do an analysis predicting **api00** from the **ell** variable converted to a 1/2/3 variable. Interpret the results.

Answer 2.
First we create the categorical variable called **ell_cat**.

```
generate ell_cat = ell
recode ell_cat 0/14=1 15/41=2 42/100=3
(385 changes made)
```

We check the creation of **ell_cat** using the **tabulate** command below.

```
tabulate ell_cat

    ell_cat |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        136       34.00       34.00
          2 |        129       32.25       66.25
          3 |        135       33.75      100.00
------------+-----------------------------------
      Total |        400      100.00
```

We use **xi** with the **regress** command to perform this analysis, and this creates two dummy codes with category 1 (low number of English language learners) as the reference category.

```
xi : regress api00 i.ell_cat
i.ell_cat         _Iell_cat_1-3       (naturally coded; _Iell_cat_1
omitted)

      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
252.88
       Model |  4523139.17     2  2261569.59           Prob > F      =
0.0000
    Residual |  3550532.82   397  8943.40762           R-squared     =
0.5602
```

```
-------------+------------------------------           Adj R-squared =
0.5580
       Total |  8073672.00   399  20234.7669           Root MSE      =
94.57

----------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
 _Iell_cat_2 |   -141.319   11.62278   -12.16   0.000    -164.1689   -
118.4691
 _Iell_cat_3 |  -257.9758   11.48947   -22.45   0.000    -280.5636    -
235.388
       _cons |   780.2647   8.109276    96.22   0.000     764.3222
796.2072
----------------------------------------------------------------------
-------
```

The **_cons** represents the mean for the reference category, when **ell_cat** is coded 1. The coefficient for **_Iell_cat_2** is the difference in the mean api score between the **ell_cat**=2 group and the reference group, **ell_cat**=1, and this difference is significant. The schools with a middle amount of English language learners score 141 points lower on their api score as compared to the schools with low amounts of English language learners. The coefficient for **_Iell_cat_3** is the difference in the api scores for the **ell_cat**=3 group and the reference group, and this is significant as well. The schools with high amounts of English language learners score about 257 points lower than schools with low amounts of English language learners.

3. Do a regression analysis predicting **api00** from **yr_rnd** and the **ell** variable converted to a 0/1 variable. Then create an interaction term and run the analysis again. Interpret the results of these analyses.

Answer 3.
We use the **regress** command to perform this analysis below.

```
regress api00 yr_rnd ell_bin
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  2,   397) =
270.17
       Model |  4654146.48     2  2327073.24           Prob > F      =
0.0000
    Residual |  3419525.51   397  8613.41439           R-squared     =
0.5765
-------------+------------------------------           Adj R-squared =
0.5743
       Total |  8073672.00   399  20234.7669           Root MSE      =
92.808

----------------------------------------------------------------------
-------
```

```
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. 
Interval]
-------------+----------------------------------------------------------
-------
      yr_rnd |   -77.6647   11.93665    -6.51   0.000    -101.1316   -
54.19776
     ell_bin |   -182.082   10.04678   -18.12   0.000    -201.8335   -
162.3304
       _cons |   756.0712   6.598791   114.58   0.000     743.0982    
769.0441
------------------------------------------------------------------------
-------
```

These results indicate that year round schools (**yr_rnd**=1) score about 77 points lower on the api test than non-year round schools (**yr_rnd**=0). Also, schools with high numbers of English language learners score about 182 points lower on the api test than the schools with low numbers of English language learners. Both of these effects are significant.

Now we include an interaction term in the analysis.

```
generate yr_ell = yr_rnd*ell_bin
regress api00 yr_rnd ell_bin yr_ell
      Source |       SS       df       MS              Number of obs = 
400
-------------+------------------------------           F(  3,   396) = 
179.67
       Model |  4654224.91     3  1551408.30           Prob > F      = 
0.0000
    Residual |  3419447.09   396  8634.96739           R-squared     = 
0.5765
-------------+------------------------------           Adj R-squared = 
0.5733
       Total |  8073672.00   399  20234.7669           Root MSE      = 
92.925

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. 
Interval]
-------------+----------------------------------------------------------
-------
      yr_rnd |  -75.49121     25.748    -2.93   0.004    -126.1111   -
24.87135
     ell_bin |  -181.6966   10.84157   -16.76   0.000    -203.0109   -
160.3824
      yr_ell |  -2.770387   29.06936    -0.10   0.924    -59.91995    
54.37918
       _cons |   755.9198   6.795314   111.24   0.000     742.5604    
769.2792
------------------------------------------------------------------------
-------
```

The main effects of **yr_rnd** and **ell_bin** are still significant, but the interaction term **yr_ell** is not significant. This suggests that the effects we described in the analysis above are consistent across the levels of **yr_rnd** and **ell_bin**. In other words, we can say that the effect of **ell_bin** is much the same for the year round schools as for the non-year round schools.

We could also have run this analysis using the **anova** command, which can be much more convenient for models like these.

```
anova api00 yr_rnd ell_bin yr_rnd*ell_bin
                           Number of obs =     400     R-squared     =
0.5765
                           Root MSE      = 92.9245     Adj R-squared =
0.5733

                  Source |  Partial SS    df       MS           F
Prob > F
          ---------------+----------------------------------------------------
-------
                   Model |  4654224.91     3  1551408.30     179.67
0.0000
                         |
                  yr_rnd |  241566.044     1  241566.044      27.98
0.0000
                 ell_bin |  1370062.10     1  1370062.10     158.66
0.0000
          yr_rnd*ell_bin |  78.4279246     1  78.4279246       0.01
0.9241
                         |
                Residual |  3419447.09   396  8634.96739
          ---------------+----------------------------------------------------
-------
                   Total |  8073672.00   399  20234.7669
```

And we can use the **adjust** command to get the means for the cells. You can relate the coefficients from the regression model to the means below. For example, the **_cons** is the mean for the cell where all the variables are 0, and so forth.

```
adjust , by(yr_rnd ell_bin)

-----------------------------------------------------
     Dependent variable: api00     Command: anova
-----------------------------------------------------

----------------------------
year      |
round     |     ell_bin
school    |       0        1
----------+-----------------
       No |  755.92  574.223
      Yes | 680.429  495.962
----------------------------
     Key:  Linear Prediction
```

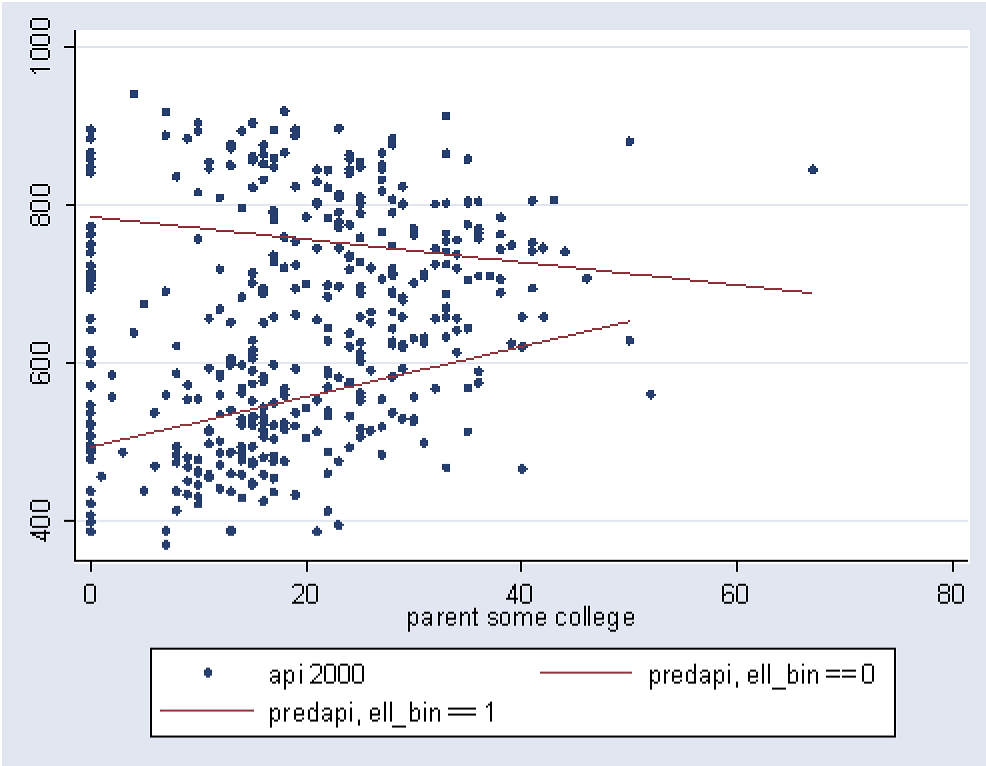
4. Do a regression analysis predicting **api00** from **ell** coded as 0/1 (from question 1) and **some_col**, and the interaction of these two variables. Interpret the results, including showing a graph of the results.

Answer 4.
Create an interaction and run the analysis

```
gen ell_col = ell_bin*some_col
regress api00 ell_bin some_col ell_col
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
167.96
       Model |  4520787.76     3  1506929.25           Prob > F      =
0.0000
    Residual |  3552884.24   396   8971.9299           R-squared     =
0.5599
-------------+------------------------------           Adj R-squared =
0.5566
       Total |  8073672.00   399  20234.7669           Root MSE      =
94.72

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
     ell_bin |  -291.6353   19.90341   -14.65   0.000    -330.7649   -
252.5057
    some_col |  -1.443942   .5595276    -2.58   0.010    -2.543958   -
.3439265
     ell_col |   4.622981   .9174408     5.04   0.000     2.819317
6.426644
       _cons |   784.5261   14.72533    53.28   0.000     755.5765
813.4757
------------------------------------------------------------------------
-------
```

Make a graph to help in the interpretation.

```
predict predapi
separate predapi, by(ell_bin)

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------
--------
predapi0        float  %9.0g                  predapi, ell_bin == 0
predapi1        float  %9.0g                  predapi, ell_bin == 1
graph twoway scatter api00 predapi0 predapi1 some_col, ///
  connect(i l l i) msymbol(o i i o) pstyle(p1 p2 p2 p1) sort
```



The graph helps us visually understand the interaction represented by **ell_col**. We can see that the regression lines between **some_col** and **api00** are not parallel -- specifically, the line for the schools with a low number of English language learners has a downward slope, and the line for the schools with a large number of English language learners has an upward slope. From the regression equation, we see that the slope of the line when **ell_bin** is 0 (low number of English language learners) is -1.44. This corresponds to the solid regression line we see in the above graph. The difference between the slopes for the schools with a high number of English language learners and the schools with a low number of English language learners is 4.62. In order to get the slopes for the schools with a high number of English language learners we would add 4.62 to -1.44 and that yields 3.18, so this is the slope for the line for the schools with the high number of English language learners. This corresponds to the dotted regression line that we see in the above graph.

5. Use the variable **ell** converted into 3 categories (from question 2) and predict api00 from **ell** in 3 categories, from **some_col** and the interaction. of these two variables. Interpret the results, including showing a graph.

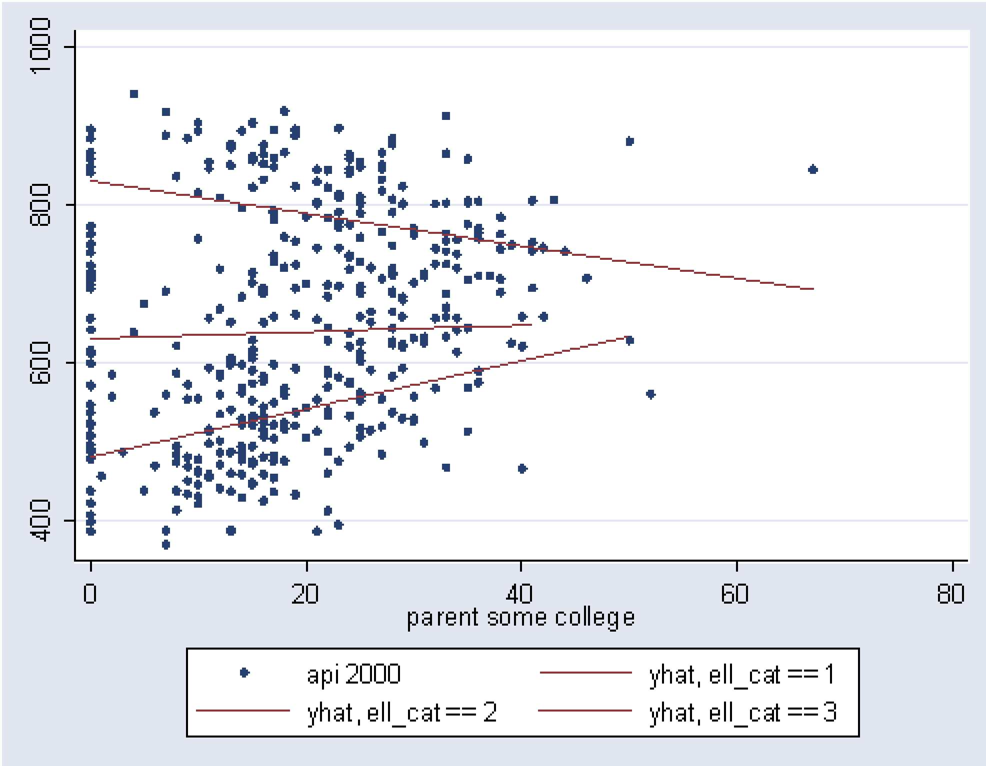
We use the **xi** command with **regress** to perform the analysis looking at the effect of **some_col** and **ell_cat** and the interaction.

```
xi : regress api00 i.ell_cat*some_col
i.ell_cat         _Iell_cat_1-3       (naturally coded; _Iell_cat_1
omitted)
i.ell_~t*some~l   _IellXsome__#       (coded as above)
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  5,   394) =
109.56
       Model |  4696120.14     5  939224.028           Prob > F      =
0.0000
    Residual |  3377551.86   394  8572.46664           R-squared     =
0.5817
-------------+------------------------------           Adj R-squared =
0.5763
       Total |  8073672.00   399  20234.7669           Root MSE      =
92.588

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
 _Iell_cat_2 |  -199.7005    25.2889    -7.90   0.000    -249.4186   -
149.9825
 _Iell_cat_3 |  -349.7611   23.60764   -14.82   0.000    -396.1738   -
303.3484
    some_col |  -2.056112   .6695881    -3.07   0.002    -3.372524   -
.7396998
_IellXsome~2 |    2.48773   1.003112     2.48   0.014     .5156074
4.459852
_IellXsome~3 |   5.112258   1.159782     4.41   0.000     2.832123
7.392393
       _cons |   829.4451   17.87578    46.40   0.000     794.3012
864.5889
------------------------------------------------------------------------
-------
```

To help interpretation, lets make a graph of the predicted values.

```
predict yhat
(option xb assumed; fitted values)
separate yhat, by(ell_cat)

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------
--------
yhat1           float  %9.0g                  yhat, ell_cat == 1
yhat2           float  %9.0g                  yhat, ell_cat == 2
yhat3           float  %9.0g                  yhat, ell_cat == 3
graph twoway scatter api00 yhat1 yhat2 yhat3 some_col, ///
  connect(i l l l i) msymbol(o i i i o) pstyle(p1 p2 p2 p2 p1) sort
```



We can use the information in the graph and in the regression equation to help interpret these results. First looking at the graph, we see that the slopes of the three regression lines are not parallel. For the schools with a low number of English language learners (when **ell_cat** is 1) the regression line has a downward slope, for the schools with a middle number of English language learners (when **ell_cat** is 2) the regression line is pretty flat, and for the schools with a high number of English language learners (when **ell_cat** is 3) the regression line has an upward tilt. we can use the regression model to compute the exact slopes of all three of these regression lines. Since group 1 is the reference category the slope for that regression line is the slope for **some_col,** which is -2.05.

The coefficient for **_IellXsome~2** (2.48) tells us how much we need to add to -2.05 to get the coefficient for the second group. when we add -2.05 to 2.48 we get .43, the slope for the second group. Because the coefficient **_IellXsome~2** is significant we can say that the coefficient for group 1 is significantly different from group 2.

The coefficient for **_IellXsome~3** (5.11) tells us how much we need to add to -2.05 to get the coefficient for the third group. when we add -2.05 to 5.11 we get 3.06, the slope for the third group. Because the coefficient **_IellXsome~3** is significant we can say that the coefficient for group 1 is significantly different from group 3.

Chapter 4 - Beyond OLS

Chapter Outline

In this chapter we will go into various commands that go beyond OLS. This chapter is a bit different from the others in that it covers a number of different concepts, some of which may be new to you. These extensions, beyond OLS, have much of the look and feel of OLS but will provide you with additional tools to work with linear models.

The topics will include robust regression methods, constrained linear regression, regression with censored and truncated data, regression with measurement error, and multiple equation models.

4.1 Robust Regression Methods

It seems to be a rare dataset that meets all of the assumptions underlying multiple regression. We know that failure to meet assumptions can lead to biased estimates of coefficients and especially biased estimates of the standard errors. This fact explains a lot of the activity in the development of robust regression methods.

The idea behind robust regression methods is to make adjustments in the estimates that take into account some of the flaws in the data itself. We are going to look at three approaches to robust regression: 1) regression with robust standard errors including the **cluster** option, 2) robust regression using iteratively reweighted least squares, and 3) quantile regression, more specifically, median regression.

Before we look at these approaches, let's look at a standard OLS regression using the elementary school academic performance index (elemapi2.dta) dataset.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2
```

We will look at a model that predicts the api 2000 scores using the average class size in K through 3 (**acs_k3**), average class size 4 through 6 (**acs_46**), the percent of fully credentialed teachers (**full**), and the size of the school (**enroll**). First let's look at the descriptive statistics for these variables. Note the missing values for **acs_k3** and **acs_k6**.

```
summarize api00 acs_k3 acs_46 full enroll

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
   api00 |     400    647.6225    142.249        369        940
  acs_k3 |     398     19.1608   1.368693         14         25
  acs_46 |     397    29.68514   3.840784         20         50
    full |     400       84.55   14.94979         37        100
  enroll |     400     483.465   226.4484        130       1570
```

Below we see the regression predicting **api00** from **acs_k3**, **acs_46 full** and **enroll**. We see that all of the variables are significant except for **acs_k3**.

```
regress api00 acs_k3 acs_46 full enroll

  Source |       SS       df       MS                  Number of obs =
395
---------+------------------------------               F(  4,   390) =
61.01
   Model |  3071909.06     4  767977.265               Prob > F      =
0.0000
Residual |  4909500.73   390  12588.4634               R-squared     =
0.3849
---------+------------------------------               Adj R-squared =
0.3786
   Total |  7981409.79   394  20257.3852               Root MSE      =
112.20

------------------------------------------------------------------------
-------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  acs_k3 |   6.954381   4.371097      1.591   0.112       -1.63948
15.54824
  acs_46 |   5.966015   1.531049      3.897   0.000       2.955873
8.976157
    full |   4.668221   .4142537     11.269   0.000       3.853771
5.482671
  enroll |  -.1059909   .0269539     -3.932   0.000      -.1589841   -
.0529977
   _cons |  -5.200407   84.95492     -0.061   0.951      -172.2273
161.8265
------------------------------------------------------------------------
-------
```

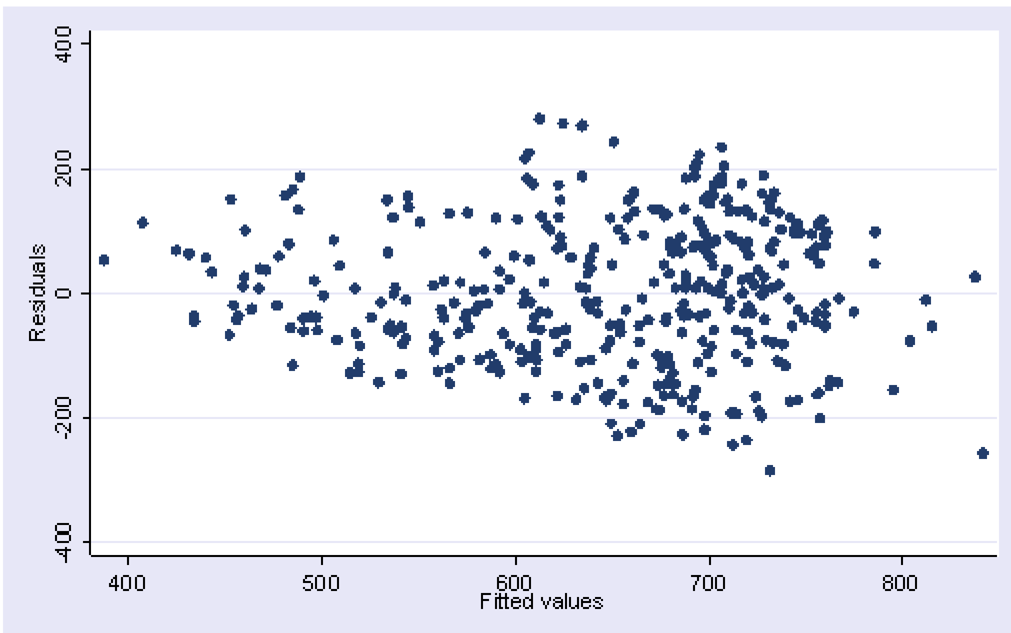
We can use the **test** command to test both of the class size variables, and we find the overall test of these two variables is significant.

test acs_k3 acs_46

```
 ( 1)  acs_k3 = 0.0
 ( 2)  acs_46 = 0.0

       F(  2,   390) =   11.08
            Prob > F =    0.0000
```

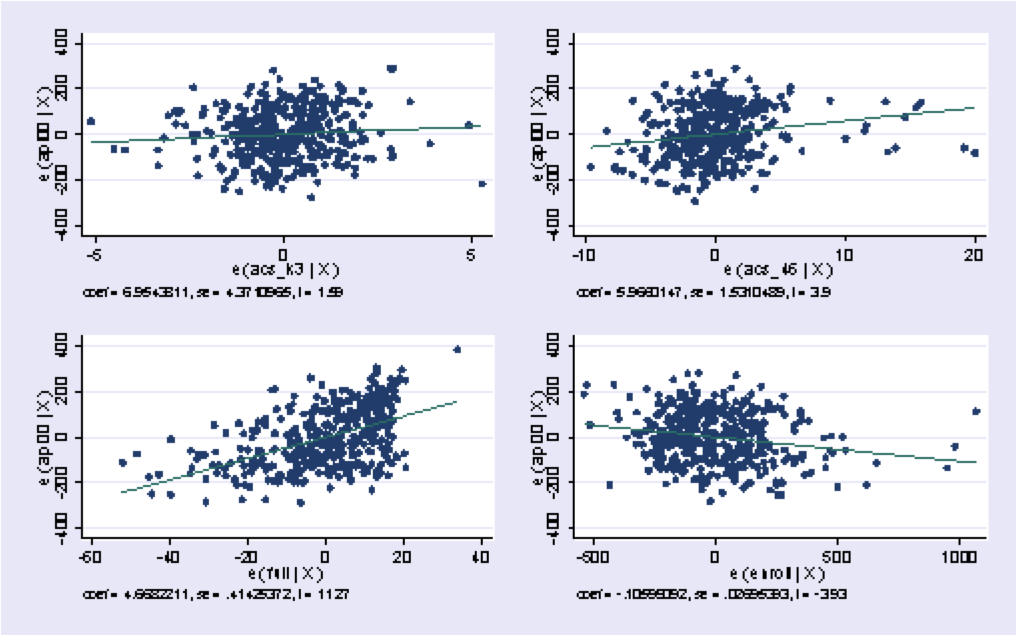
Here is the residual versus fitted plot for this regression. Notice that the pattern of the residuals is not exactly as we would hope. The spread of the residuals is somewhat wider toward the middle right of the graph than at the left, where the variability of the residuals is somewhat smaller, suggesting some heteroscedasticity.

rvfplot



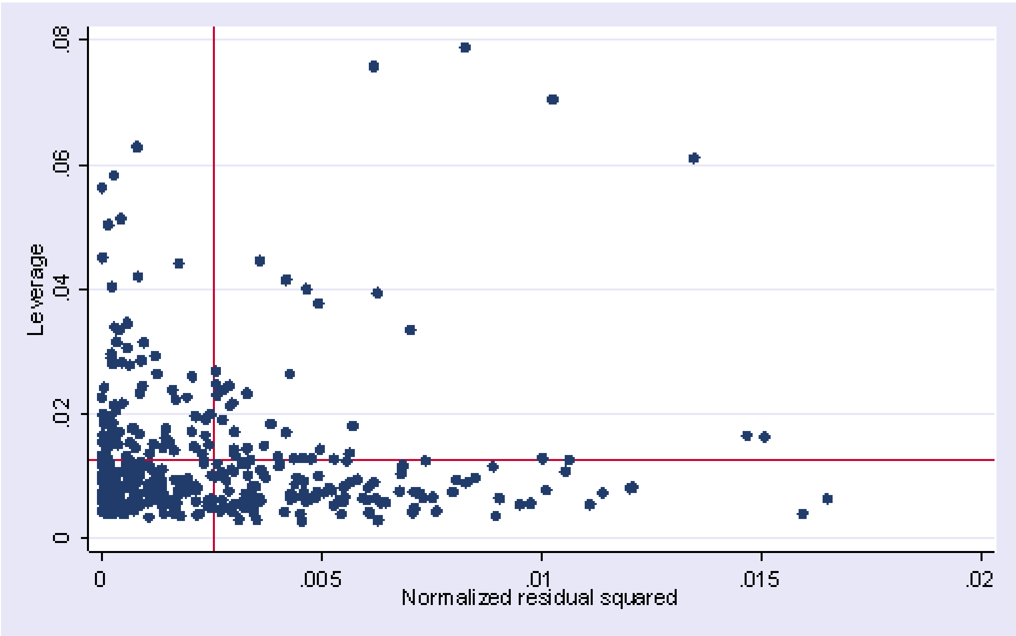
Below we show the **avplots**. Although the plots are small, you can see some points that are of concern. There is not a single extreme point (like we saw in chapter 2) but a handful of points that stick out. For example, in the top right graph you can see a handful of points that stick out from the rest. If this were just one or two points, we might look for mistakes or for outliers, but we would be more reluctant to consider such a large number of points as outliers.

avplots



Here is the **lvr2plot** for this regression. We see 4 points that are somewhat high in both their leverage and their residuals.

```
lvr2plot
```



None of these results are dramatic problems, but the **rvfplot** suggests that there might be some outliers and some possible heteroscedasticity; the **avplots** have some observations that look to have high leverage, and the **lvr2plot** shows some points in the upper right quadrant that could be influential. We might wish to use something other than OLS regression to estimate this model. In the next several sections we will look at some robust regression methods.

4.1.1 Regression with Robust Standard Errors

The Stata **regress** command includes a **robust** option for estimating the standard errors using the Huber-White sandwich estimators. Such robust standard errors can deal with a collection of minor concerns about failure to meet assumptions, such as minor problems about normality, heteroscedasticity, or some observations that exhibit large residuals, leverage or influence. For such minor problems, the robust option may effectively deal with these concerns.

With the **robust** option, the point estimates of the coefficients are exactly the same as in ordinary OLS, but the standard errors take into account issues concerning heterogeneity and lack of normality. Here is the same regression as above using the **robust** option. Note the changes in the standard errors and t-tests (but no change in the coefficients). In this particular example, using robust standard errors did not change any of the conclusions from the original OLS regression.

```
regress api00 acs_k3 acs_46 full enroll, robust

Regression with robust standard errors                 Number of obs =
395
                                                       F(  4,   390) =
84.67
                                                       Prob > F      =
0.0000
                                                       R-squared     =
0.3849
                                                       Root MSE      =
112.20

------------------------------------------------------------------------
-------
         |               Robust
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  acs_k3 |   6.954381   4.620599      1.505   0.133      -2.130019
16.03878
  acs_46 |   5.966015   1.573214      3.792   0.000       2.872973
9.059057
    full |   4.668221   .4146813     11.257   0.000       3.852931
5.483512
  enroll |  -.1059909   .0280154     -3.783   0.000      -.1610711   -
.0509108
   _cons |  -5.200407   86.66308     -0.060   0.952      -175.5857
165.1849
------------------------------------------------------------------------
-------
```

4.1.2 Using the Cluster Option

As described in Chapter 2, OLS regression assumes that the residuals are independent. The **elemapi2** dataset contains data on 400 schools that come from 37 school districts. It is very possible that the scores within each school district may not be independent, and this could lead to residuals that are not independent within districts. We can use the **cluster** option to indicate that the observations are clustered into districts (based on **dnum**) and that the observations may be correlated within districts, but would be independent between districts.

By the way, if we did not know the number of districts, we could quickly find out how many districts there are as shown below, by **quietly** tabulating **dnum** and then displaying the macro **r(r)** which gives the numbers of rows in the table, which is the number of school districts in our data.

```
quietly tabulate dnum
display r(r)
37
```

Now, we can run regress with the **cluster** option. We do not need to include the robust option since robust is implied with cluster. Note that the standard errors have changed substantially, much more so, than the change caused by the **robust** option by itself.

```
regress api00 acs_k3 acs_46 full enroll, cluster(dnum)

Regression with robust standard errors                Number of obs =
395
                                                      F(  4,    36) =
31.18
                                                      Prob > F      =
0.0000
                                                      R-squared     =
0.3849
Number of clusters (dnum) = 37                        Root MSE      =
112.20

-----------------------------------------------------------------------
-------
         |               Robust
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
  acs_k3 |   6.954381   6.901117      1.008   0.320      -7.041734
20.9505
  acs_46 |   5.966015   2.531075      2.357   0.024       .8327565
11.09927
    full |   4.668221   .7034641      6.636   0.000        3.24153
6.094913
  enroll |  -.1059909   .0429478     -2.468   0.018      -.1930931   -
.0188888
   _cons |  -5.200407   121.7856     -0.043   0.966       -252.193
241.7922
-----------------------------------------------------------------------
-------
```

As with the **robust** option, the estimate of the coefficients are the same as the OLS estimates, but the standard errors take into account that the observations within districts are non-independent. Even though the standard errors are larger in this analysis, the three variables that were significant in the OLS analysis are significant in this analysis as well. These standard errors are computed based on aggregate scores for the 37 districts, since these district level scores should be independent. If you have a very small number of clusters compared to your overall sample size it is possible that the standard errors could be quite larger than the OLS results. For example, if there were only 3 districts, the standard errors would be computed on the aggregate scores for just 3 districts.

4.1.3 Robust Regression

The Stata **rreg** command performs a robust regression using iteratively reweighted least squares, i.e., **rreg** assigns a weight to each observation with higher weights given to better behaved observations. In fact, extremely deviant cases, those with Cook's D greater than 1, can have their weights set to missing so that they are not included in the analysis at all.

We will use **rreg** with the generate option so that we can inspect the weights used to weight the observations. Note that in this analysis both the coefficients and the standard errors differ from the original OLS regression. Below we show the same analysis using robust regression using the **rreg** command.

```
rreg api00 acs_k3 acs_46 full enroll, gen(wt)

Robust regression estimates                               Number of obs =
395
                                                          F(  4,   390) =
56.51
                                                          Prob > F      =
0.0000

-----------------------------------------------------------------------
-------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
  acs_k3 |   6.110881   4.658131      1.312   0.190      -3.047308
15.26907
  acs_46 |   6.254708   1.631587      3.834   0.000       3.046901
9.462516
    full |   4.796072   .4414563     10.864   0.000        3.92814
5.664004
  enroll |  -.1092586   .0287239     -3.804   0.000      -.1657316   -
.0527855
   _cons |  -6.788183    90.5336     -0.075   0.940      -184.7832
171.2068
-----------------------------------------------------------------------
-------
```

If you compare the robust regression results (directly above) with the OLS results previously presented, you can see that the coefficients and standard errors are quite similar, and the t values and p values are also quite similar. Despite the minor problems that we found in the data when we performed the OLS analysis, the robust regression analysis yielded quite similar results suggesting that indeed these were minor problems. Had the results been substantially different, we would have wanted to further investigate the reasons why the OLS and robust regression results were different, and among the two results the robust regression results would probably be the more trustworthy.

Let's calculate and look at the predicted (fitted) values (**p**), the residuals (**r**), and the leverage (hat) values (**h**). Note that we are including **if e(sample)** in the commands because **rreg** can generate weights of missing and you wouldn't want to have predicted values and residuals for those observations.

```
predict p if e(sample)
(option xb assumed; fitted values)
(5 missing values generated)

predict r if e(sample), resid
(5 missing values generated)

predict h if e(sample), hat
(5 missing values generated)
```

Now, let's check on the various predicted values and the weighting. First, we will sort by **wt** then we will look at the first 15 observations. Notice that the smallest weights are near one-half but quickly get into the .7 range.

```
sort wt
list snum api00 p r h wt in 1/15

          snum   api00          p           r          h           wt
  1.       637     447   733.1567   -286.1568   .0037645   .55612093
  2.      5387     892   611.5344    280.4655   .0023925   .57126927
  3.      2267     897   621.4881    275.5119    .010207   .58433963
  4.        65     903   631.2718    271.7282   .0105486   .59425026
  5.      3759     585   842.4838   -257.4838   .0414728   .63063771
  6.      5926     469   715.2266   -246.2266   .0058346   .65892631
  7.      1978     894   650.7816    243.2184   .0058116    .6665881
  8.      3696     483   721.3105   -238.3105   .0052619   .67834344
  9.      5222     940    707.648     232.352   .0041016   .69303069
 10.       690     424   654.5795   -230.5795   .0094319   .69701005
 11.      3785     459   687.3311   -228.3311   .0081474   .70245717
 12.      2910     831   604.4401      226.56   .0536809   .70650365
 13.       699     437   660.2588   -223.2588   .0059152   .71449402
 14.      3070     479   698.1256   -219.1256   .0043322   .72399766
 15.      1812     917   698.9828    218.0172   .0099871   .72670695
```

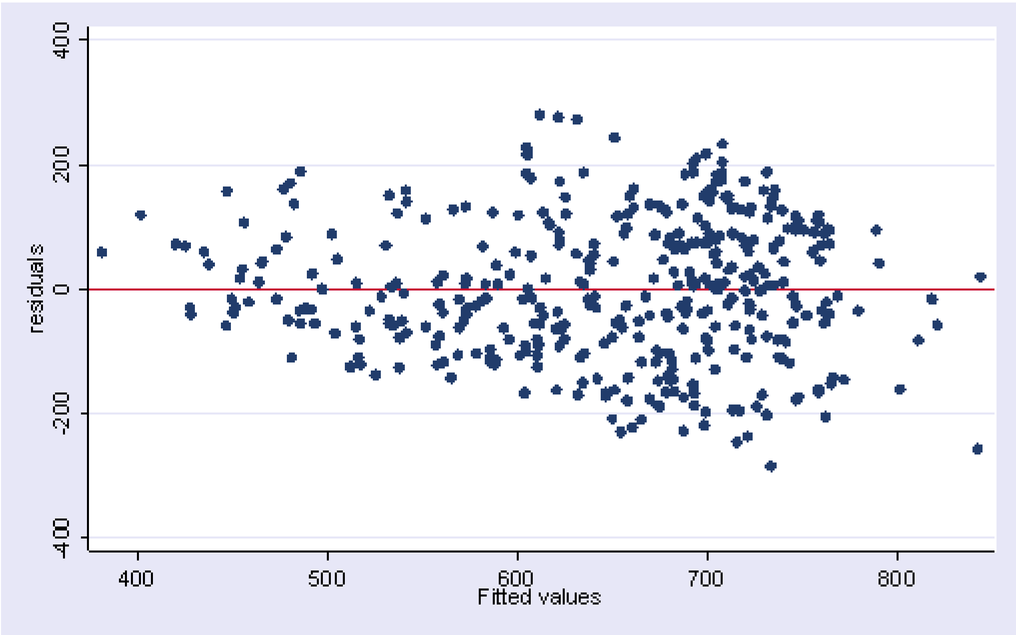
Now, let's look at the last 10 observations. The weights for observations 391 to 395 are all very close to one. The values for observations 396 to the end are missing due to the missing predictors. Note that the observations above that have the lowest weights are also those with the largest residuals (residuals over 200) and the observations below with the highest weights have very low residuals (all less than 3).

```
list snum api00 p r h wt in -10/l
```

```
          snum   api00          p          r          h          wt
391.      3024     727   729.0243  -2.024302   .0104834   .99997367
392.      3535     705    703.846   1.154008   .0048329   .99999207
393.      1885     605    605.427  -.4269809   .0144377   .99999843
394.      1678     497   496.8011   .1989256   .0243301   .99999956
395.      4486     706   705.8076    .192455   .0142448   .99999986
396.      4488     521          .          .          .           .
397.      3072     763          .          .          .           .
398.      3055     590          .          .          .           .
399.       116     513          .          .          .           .
400.      4534     445          .          .          .           .
```

After using **rreg**, it is possible to generate predicted values, residuals and leverage (hat), but most of the regression diagnostic commands are not available after **rreg**. We will have to create some of them for ourselves. Here, of course, is the graph of residuals versus fitted (predicted) with a line at zero. This plot looks much like the OLS plot, except that in the OLS all of the observations would be weighted equally, but as we saw above the observations with the greatest residuals are weighted less and hence have less influence on the results.

```
scatter r p, yline(0)
```



To get an **lvr2plot** we are going to have to go through several steps in order to get the normalized squared residuals and the means of both the residuals and the leverage (hat) values.

First, we generate the residual squared (**r2**) and then divide it by the sum of the squared residuals. We then compute the mean of this value and save it as a local macro called **rm** (which we will use for creating the leverage vs. residual plot).

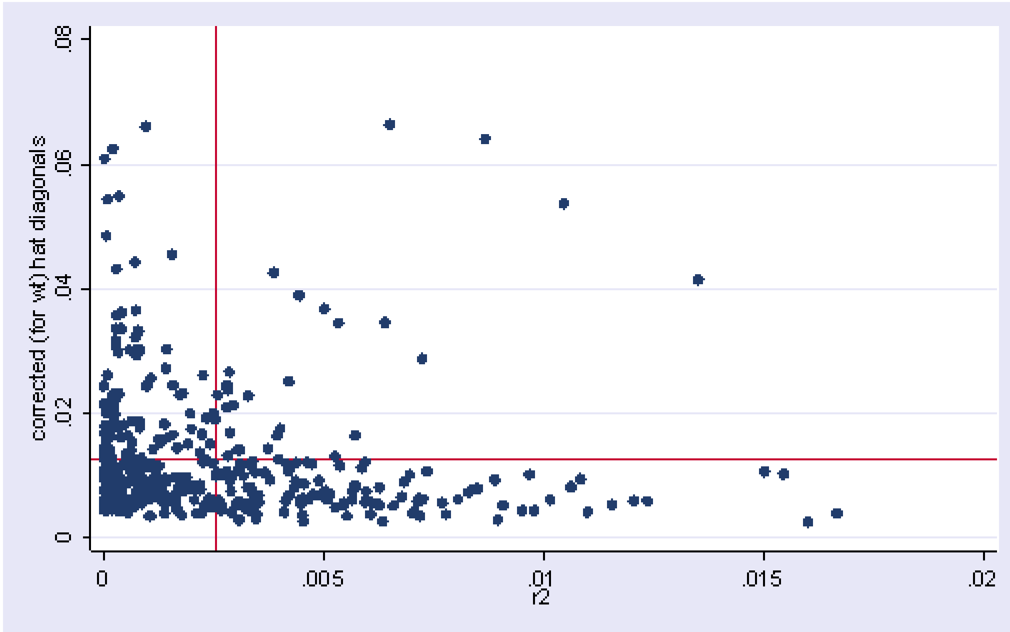
```
generate r2=r^2
(5 missing values generated)

sum r2

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      r2 |     395    12436.05   14677.98   .0370389    81885.7

replace r2 = r2/r(sum)
(395 real changes made)

summarize r2

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      r2 |     395    .0025316    .002988   7.54e-09   .0166697

local rm = r(mean)
```

Next we compute the mean of the leverage and save it as a local macro called **hm**.

```
summarize h

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
       h |     395    .0126422   .0108228   .0023925   .0664077

local hm = r(mean)
```

Now, we can plot the leverage against the residual squared as shown below. Comparing the plot below with the plot from the OLS regression, this plot is much better behaved.   There are no longer points in the upper right quadrant of the graph.

```
scatter h r2, yline(`hm') xline(`rm')
```



Let's close out this analysis by deleting our temporary variables.

```
drop wt p r h r2
```

4.1.4 Quantile Regression

Quantile regression, in general, and median regression, in particular, might be considered as an alternative to **rreg**. The Stata command **qreg** does quantile regression. **qreg** without any options will actually do a median regression in which the coefficients will be estimated by minimizing the absolute deviations from the median. Of course, as an estimate of central tendency, the median is a resistant measure that is not as greatly affected by outliers as is the mean. It is not clear that median regression is a resistant estimation procedure, in fact, there is some evidence that it can be affected by high leverage values.

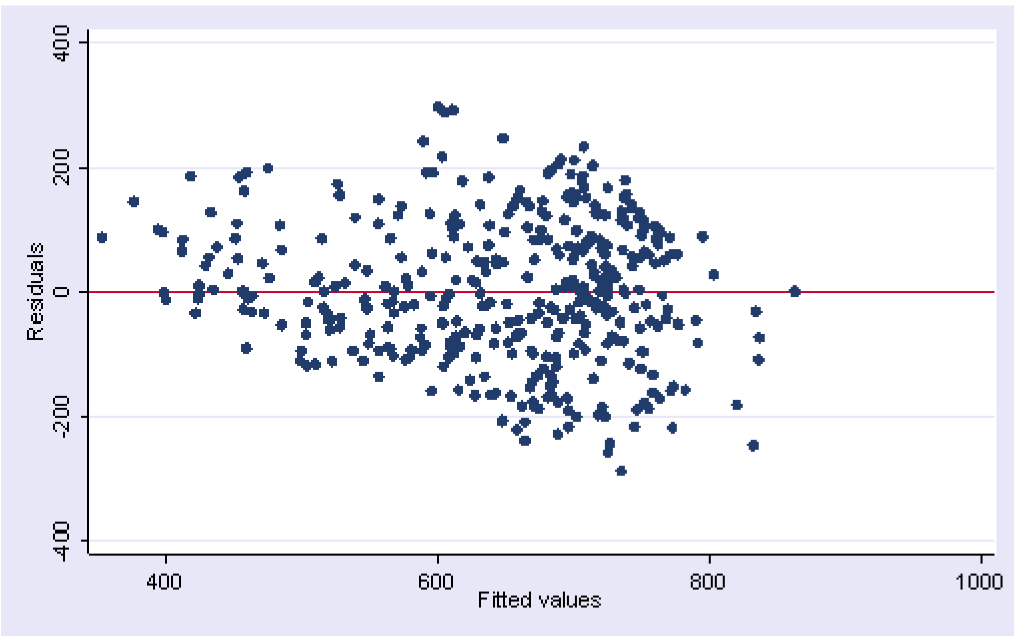
Here is what the quantile regression looks like using Stata's **qreg** command. The coefficient and standard error for **acs_k3** are considerably different when using **qreg** as compared to OLS using the **regress** command (the coefficients are 1.2 vs 6.9 and the standard errors are 6.4 vs 4.3). The coefficients and standard errors for the other variables are also different, but not as dramatically different. Nevertheless, the **qreg** results indicate that, like the OLS results, all of the variables except **acs_k3** are significant.

```
qreg api00 acs_k3 acs_46 full enroll

Median regression                                    Number of obs =
395
  Raw sum of deviations    48534 (about 643)
  Min sum of deviations 36268.11                     Pseudo R2     =
0.2527
```

```
--------------------------------------------------------------------------
-------
   api00 |      Coef.   Std. Err.       t     P>|t|       [95% Conf. 
Interval]
---------+----------------------------------------------------------------
-------
  acs_k3 |   1.269065   6.470588      0.196   0.845      -11.45253    
13.99066
  acs_46 |    7.22408   2.228949      3.241   0.001       2.841821    
11.60634
    full |   5.323841   .6157333      8.646   0.000       4.113269    
6.534413
  enroll |  -.1245734   .0397576     -3.133   0.002      -.2027395   -
.0464073
   _cons |   17.15049   125.4396      0.137   0.891      -229.4719    
263.7729
--------------------------------------------------------------------------
-------
```

The **qreg** command has even fewer diagnostic options than **rreg** does. About the only values we can obtain are the predicted values and the residuals.

```
predict p if e(sample)
(option xb assumed; fitted values)
(5 missing values generated)

predict r if e(sample), r
(5 missing values generated)

scatter r p, yline(0)
```



Stata has three additional commands that can do quantile regression.

iqreg estimates interquantile regressions, regressions of the difference in quantiles. The estimated variance-covariance matrix of the estimators is obtained via bootstrapping.

sqreg estimates simultaneous-quantile regression. It produces the same coefficients as qreg for each quantile. **sqreg** obtains a bootstrapped variance-covariance matrix of the estimators that includes between-quantiles blocks. Thus, one can test and construct confidence intervals comparing coefficients describing different quantiles.

bsqreg is the same as sqreg with one quantile. **sqreg** is, therefore, faster than bsqreg.

4.2 Constrained Linear Regression

Let's begin this section by looking at a regression model using the **hsb2** dataset. The **hsb2** file is a sample of 200 cases from the Highschool and Beyond Study (Rock, Hilton, Pollack, Ekstrom & Goertz, 1985). It includes the following variables: **id**, **female**, **race**, **ses**, **schtyp**, **program**, **read**, **write**, **math**, **science** and **socst**. The variables **read**, **write**, **math**, **science** and **socst** are the results of standardized tests on reading, writing, math, science and social studies (respectively), and the variable **female** is coded 1 if female, 0 if male.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2
```

Let's start by doing an OLS regression where we predict **socst** score from **read**, **write**, **math**, **science** and **female** (gender)

```
regress socst read write math science female

  Source |       SS       df       MS                  Number of obs =
200
---------+------------------------------               F(  5,   194) =
35.44
   Model |  10949.2575     5   2189.8515               Prob > F      =
0.0000
Residual |  11986.9375   194  61.7883375               R-squared     =
0.4774
---------+------------------------------               Adj R-squared =
0.4639
   Total |   22936.195   199  115.257261               Root MSE      =
7.8606

-----------------------------------------------------------------------
-------
   socst |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
    read |   .3784046   .0806267      4.693   0.000       .2193872
.537422
   write |   .3858743   .0889283      4.339   0.000       .2104839
.5612646
    math |   .1303258   .0893767      1.458   0.146       -.045949
.3066006
```

```
 science |  -.0333925   .0818741     -0.408   0.684      -.1948702    
.1280852
  female |  -.3532648   1.245372     -0.284   0.777      -2.809471    
2.102941
   _cons |   7.339342   3.650243      2.011   0.046       .1400864    
14.5386
------------------------------------------------------------------------
-------
```

Notice that the coefficients for **read** and **write** are very similar, which makes sense since they are both measures of language ability. Also, the coefficients for **math** and **science** are similar (in that they are both not significantly different from 0). Suppose that we have a theory that suggests that **read** and **write** should have equal coefficients, and that **math** and **science** should have equal coefficients as well. We can test the equality of the coefficients using the **test** command.

```
test read=write

 ( 1)  read - write = 0.0

       F(  1,   194) =    0.00
            Prob > F =    0.9558
```

We can also do this with the **testparm** command, which is especially useful if you were testing whether 3 or more coefficients were equal.

```
testparm read write, equal

 ( 1) - read + write = 0.0

       F(  1,   194) =    0.00
            Prob > F =    0.9558
```

Both of these results indicate that there is no significant difference in the coefficients for the reading and writing scores. Since it appears that the coefficients for **math** and **science** are also equal, let's test the equality of those as well (using the **testparm** command).

```
testparm math science, equal

 ( 1) - math + science = 0.0

       F(  1,   194) =    1.45
            Prob > F =    0.2299
```

Let's now perform both of these tests together, simultaneously testing that the coefficient for **read** equals **write** and **math** equals **science**. We do this using two **test** commands, the second using the **accum** option to accumulate the first test with the second test to test both of these hypotheses together.

```
test read=write

 ( 1)  read - write = 0.0
```

```
       F(  1,   194) =    0.00
            Prob > F =    0.9558
```

test math=science, accum

```
 ( 1)  read - write = 0.0
 ( 2)  math - science = 0.0

       F(  2,   194) =    0.73
            Prob > F =    0.4852
```

Note this second test has 2 df, since it is testing both of the hypotheses listed, and this test is not significant, suggesting these pairs of coefficients are not significantly different from each other. We can estimate regression models where we constrain coefficients to be equal to each other. For example, let's begin on a limited scale and constrain **read** to equal **write**. First, we will define a constraint and then we will run the **cnsreg** command.

```
constraint define 1 read = write
. cnsreg socst read write math science female, constraint(1)

Constrained linear regression                       Number of obs =
200
                                                    F(  4,   195) =
44.53
                                                    Prob > F      =
0.0000
                                                    Root MSE      =
7.8404
 ( 1)  read - write = 0.0
----------------------------------------------------------------------
-------
   socst |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+------------------------------------------------------------
-------
    read |   .3818488   .0513899      7.430   0.000       .2804975
.4832002
   write |   .3818488   .0513899      7.430   0.000       .2804975
.4832002
    math |   .1303036   .0891471      1.462   0.145      -.0455126
.3061197
 science |  -.0332762   .0816379     -0.408   0.684      -.1942827
.1277303
  female |  -.3296237   1.167364     -0.282   0.778      -2.631904
1.972657
   _cons |   7.354148   3.631175      2.025   0.044       .1927294
14.51557
----------------------------------------------------------------------
-------
```

Notice that the coefficients for **read** and **write** are identical, along with their standard errors, t-test, etc. Also note that the degrees of freedom for the F test is four, not five, as in the OLS model. This is because only one coefficient is estimated for **read** and **write**, estimated like a single variable equal to the sum of their values. In general, the Root MSE should increase in the constrained model, because estimation subject to linear restrictions does not improve fit relative to the unrestricted model (the coefficients that would minimize the SSE would be the coefficients from the unconstrained model). However, in this particular example (because the coefficients for read and write are already so similar) the decrease in model fit from having constrained **read** and **write** to equal each other is offset by the change in degrees of freedom .

Next, we will define a second constraint, setting **math** equal to **science**. We will also abbreviate the constraints option to **c**.

```
constraint define 2 math = science
. cnsreg socst read write math science female, c(1 2)

Constrained linear regression                          Number of obs =
200
                                                       F(  3,   196) =
58.75
                                                       Prob > F      =
0.0000
                                                       Root MSE      =
7.8496
 ( 1)  read - write = 0.0
 ( 2)  math - science = 0.0
------------------------------------------------------------------------
-------
   socst |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
    read |   .3860376   .0513322      7.520   0.000       .2848033
.4872719
   write |   .3860376   .0513322      7.520   0.000       .2848033
.4872719
    math |   .0428053   .0519238      0.824   0.411      -.0595958
.1452064
 science |   .0428053   .0519238      0.824   0.411      -.0595958
.1452064
  female |   -.200875   1.163831     -0.173   0.863      -2.496114
2.094364
   _cons |   7.505658   3.633225      2.066   0.040       .3404249
14.67089
------------------------------------------------------------------------
-------
```

Now the coefficients for **read** = **write** and **math** = **science** and the degrees of freedom for the model has dropped to three. Again, the Root MSE is slightly larger than in the prior model, but we should emphasize only very slightly larger. If indeed the population coefficients for **read** = **write** and **math** = **science**, then these combined (constrained) estimates may be more stable and generalize better to other samples. So although these estimates may lead to slightly higher standard error of prediction in this sample, they may generalize better to the population from which they came.

4.3 Regression with Censored or Truncated Data

Analyzing data that contain censored values or are truncated is common in many research disciplines. According to Hosmer and Lemeshow (1999), a censored value is one whose value is incomplete due to random factors for each subject. A truncated observation, on the other hand, is one which is incomplete due to a selection process in the design of the study.

We will begin by looking at analyzing data with censored values.

4.3.1 Regression with Censored Data

In this example we have a variable called **acadindx** which is a weighted combination of standardized test scores and academic grades. The maximum possible score on **acadindx** is 200 but it is clear that the 16 students who scored 200 are not exactly equal in their academic abilities. In other words, there is variability in academic ability that is not being accounted for when students score 200 on acadindx. The variable **acadindx** is said to be censored, in particular, it is right censored.

Let's look at the example. We will begin by looking at a description of the data, some descriptive statistics, and correlations among the variables.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/acadindx
(max possible on acadindx is 200)

describe

Contains data from acadindx.dta
  obs:           200                         max possible on acadindx
is 200
 vars:             5                         19 Jan 2001 20:14
 size:         4,800 (99.7% of memory free)
-----------------------------------------------------------------------
--------
   1. id        float  %9.0g
   2. female    float  %9.0g       fl
   3. reading   float  %9.0g
   4. writing   float  %9.0g
   5. acadindx  float  %9.0g                 academic index
-----------------------------------------------------------------------
--------

summarize

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
      id |     200       100.5   57.87918          1        200
  female |     200        .545   .4992205          0          1
 reading |     200       52.23   10.25294         28         76
 writing |     200      52.775   9.478586         31         67
acadindx |     200     172.185    16.8174        138        200

count if acadindx==200
   16
```

```
corr acadindx female reading writing
(obs=200)

         | acadindx   female  reading  writing
---------+------------------------------------
acadindx |   1.0000
  female |  -0.0821   1.0000
 reading |   0.7131  -0.0531   1.0000
 writing |   0.6626   0.2565   0.5968   1.0000
```

Now, let's run a standard OLS regression on the data and generate predicted scores in **p1**.

```
regress acadindx female reading writing

  Source |       SS       df       MS                  Number of obs =
200
---------+------------------------------               F(  3,   196) =
107.40
   Model |   34994.282     3  11664.7607               Prob > F      =
0.0000
Residual |   21287.873   196  108.611597               R-squared     =
0.6218
---------+------------------------------               Adj R-squared =
0.6160
   Total |   56282.155   199  282.824899               Root MSE      =
10.422

------------------------------------------------------------------------
-------
acadindx |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  female |  -5.832498    1.58821     -3.672   0.000      -8.964671   -
2.700324
 reading |   .7184174   .0931493      7.713   0.000       .5347138
.902121
 writing |   .7905706   .1040996      7.594   0.000       .5852715
.9958696
   _cons |   96.11841   4.489562     21.409   0.000       87.26436
104.9725
------------------------------------------------------------------------
-------

predict p1
(option xb assumed; fitted values)
```

The **tobit** command is one of the commands that can be used for regression with censored data. The syntax of the command is similar to regress with the addition of the **ul** option to indicate that the right censored value is 200. We will follow the **tobit** command by predicting **p2** containing the **tobit** predicted values.

```
tobit acadindx female reading writing, ul(200)
```

```
Tobit estimates                                   Number of obs   =
200
                                                  LR chi2(3)      =
190.39
                                                  Prob > chi2     =
0.0000
Log likelihood = -718.06362                       Pseudo R2       =
0.1171

------------------------------------------------------------------------
-------
acadindx |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  female |  -6.347316   1.692441     -3.750   0.000      -9.684943   -
3.009688
 reading |   .7776857   .0996928      7.801   0.000       .5810837
.9742877
 writing |   .8111221    .110211      7.360   0.000       .5937773
1.028467
   _cons |   92.73782   4.803441     19.307   0.000       83.26506
102.2106
---------+--------------------------------------------------------------
-------
     _se |   10.98973   .5817477              (Ancillary parameter)
------------------------------------------------------------------------
-------

Obs. summary:       184 uncensored observations
                     16 right-censored observations at acadindx>=200
```

predict p2

```
(option xb assumed; fitted values)
```

Summarizing the **p1** and **p2** scores shows that the **tobit** predicted values have a larger standard deviation and a greater range of values.

summarize acadindx p1 p2

```
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
acadindx |     200     172.185    16.8174        138        200
      p1 |     200     172.185   13.26087   142.3821   201.5311
      p2 |     200     172.704   14.00292   141.2211   203.8541
```

When we look at a listing of **p1** and **p2** for all students who scored the maximum of 200 on acadindx, we see that in every case the **tobit** predicted value is greater than the OLS predicted value. These predictions represent an estimate of what the variability would be if the values of **acadindx** could exceed 200.

list p1 p2 if acadindx==200

```
            p1         p2
```

```
 32.   179.175     179.62
 57.  192.6806   194.3291
 68.  201.5311   203.8541
 80.  191.8309    193.577
 82.  188.1537   189.5627
 88.  186.5725   187.9405
 95.  195.9971   198.1762
100.  186.9333   188.1076
132.  197.5782   199.7984
136.  189.4592   191.1436
143.  191.1846   192.8327
157.  191.6145   193.4767
161.  180.2511   181.0082
169.   182.275   183.3667
174.  191.6145   193.4767
200.  187.6616   189.4211
```

Here is the syntax diagram for tobit:

```
tobit depvar [indepvars] [weight] [if exp] [in range], ll[(#)] ul[(#)]
        [ level(#) offset(varname) maximize_options ]
```

You can declare both lower and upper censored values. The censored values are fixed in that the same lower and upper values apply to all observations.

There are two other commands in Stata that allow you more flexibility in doing regression with censored data.

cnreg estimates a model in which the censored values may vary from observation to observation.

intreg estimates a model where the response variable for each observation is either point data, interval data, left-censored data, or right-censored data.

4.3.2 Regression with Truncated Data

Truncated data occurs when some observations are not included in the analysis because of the value of the variable. We will illustrate analysis with truncation using the dataset, **acadindx**, that was used in the previous section. If **acadindx** is no longer loaded in memory you can get it with the following use command.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/acadindx
(max possible on acadindx is 200)
```

Let's imagine that in order to get into a special honors program, students need to score at least 160 on **acadindx**. So we will drop all observations in which the value of **acadindx** is less than 160.

```
drop if acadindx <= 160
(56 observations deleted)
```

Now, let's estimate the same model that we used in the section on censored data, only this time we will pretend that a 200 for **acadindx** is not censored.

```
regress acadindx female reading writing

      Source |       SS       df       MS              Number of obs =
144
-------------+------------------------------           F(  3,   140) =
33.01
       Model |  8074.79638     3  2691.59879           Prob > F      =
0.0000
    Residual |  11416.3633   140  81.5454524           R-squared     =
0.4143
-------------+------------------------------           Adj R-squared =
0.4017
       Total |  19491.1597   143  136.301816           Root MSE      =
9.0303

------------------------------------------------------------------------
-------
    acadindx |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
      female |  -5.238495   1.615632    -3.24   0.001    -8.432687   -
2.044303
     reading |   .4411066   .0963504     4.58   0.000     .2506166
.6315965
     writing |   .5873287   .1150828     5.10   0.000     .3598037
.8148537
       _cons |   125.6355   5.891559    21.32   0.000     113.9875
137.2834
------------------------------------------------------------------------
-------
```

It is clear that the estimates of the coefficients are distorted due to the fact that 56 observations are no longer in the dataset. This amounts to restriction of range on both the response variable and the predictor variables. For example, the coefficient for writing dropped from .79 to .59. What this means is that if our goal is to find the relation between **acadindx** and the predictor variables in the population, then the truncation of **acadindx** in our sample is going to lead to biased estimates. A better approach to analyzing these data is to use truncated regression. In Stata this can be accomplished using the **truncreg** command where the **ll** option is used to indicate the lower limit of **acadindx** scores used in the truncation.

```
truncreg acadindx female reading writing, ll(160)
(note: 0 obs. truncated)

Truncated regression
Limit:   lower =        160                             Number of obs =
144
         upper =       +inf                             Wald chi2(3)  =
77.87
Log likelihood = -510.00768                             Prob > chi2   =
0.0000
```

```
------------------------------------------------------------------------------
-------
    acadindx |      Coef.   Std. Err.      z    P>|z|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-------
eq1          |
      female |  -6.099602   1.925245    -3.17   0.002    -9.873012   -
2.326191
     reading |   .5181789   .1168288     4.44   0.000     .2891986
.7471592
     writing |   .7661636     .15262     5.02   0.000     .4670339
1.065293
       _cons |   110.2892   8.673849    12.72   0.000     93.28877
127.2896
-------------+----------------------------------------------------------------
-------
sigma        |
       _cons |   9.803572    .721646    13.59   0.000     8.389172
11.21797
------------------------------------------------------------------------------
-------
```

The coefficients from the **truncreg** command are closer to the OLS results, for example the coefficient for **writing** is .77 which is closer to the OLS results of .79. However, the results are still somewhat different on the other variables, for example the coefficient for reading is .52 in the **truncreg** as compared to .72 in the original OLS with the unrestricted data, and better than the OLS estimate of .47 with the restricted data. While **truncreg** may improve the estimates on a restricted data file as compared to OLS, it is certainly no substitute for analyzing the complete unrestricted data file.

4.4 Regression with Measurement Error

As you will most likely recall, one of the assumptions of regression is that the predictor variables are measured without error. The problem is that measurement error in predictor variables leads to under estimation of the regression coefficients. Stata's **eivreg** command takes measurement error into account when estimating the coefficients for the model.

Let's look at a regression using the hsb2 dataset.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2

regress write read female
```

```
  Source |       SS       df       MS                  Number of obs =
200
---------+------------------------------               F(  2,   197) =
77.21
   Model |  7856.32118     2  3928.16059               Prob > F      =
0.0000
Residual |  10022.5538   197  50.8759077               R-squared     =
0.4394
```

```
---------+------------------------------               Adj R-squared =
0.4337
   Total |   17878.875   199   89.843593               Root MSE      =
7.1327

------------------------------------------------------------------------
-------
   write |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
    read |   .5658869   .0493849     11.459   0.000        .468496
.6632778
  female |   5.486894   1.014261      5.410   0.000        3.48669
7.487098
   _cons |   20.22837   2.713756      7.454   0.000       14.87663
25.58011
------------------------------------------------------------------------
-------
```

The predictor read is a standardized test score. Every test has measurement error. We don't know the exact reliability of **read**, but using .9 for the reliability would probably not be far off. We will now estimate the same regression model with the Stata **eivreg** command, which stands for errors-in-variables regression.

```
eivreg write read female, r(read .9)

               assumed                           errors-in-variables
regression
variable     reliability
------------------------                             Number of obs =
200
    read       0.9000                                F(  2,   197) =
83.41
       *       1.0000                                Prob > F      =
0.0000
                                                     R-squared     =
0.4811
                                                     Root MSE      =
6.86268

------------------------------------------------------------------------
-------
   write |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
    read |   .6289607   .0528111     11.910   0.000        .524813
.7331085
  female |   5.555659   .9761838      5.691   0.000       3.630548
7.48077
   _cons |   16.89655   2.880972      5.865   0.000       11.21504
22.57805
```

Note that the F-ratio and the R^2 increased along with the regression coefficient for **read**. Additionally, there is an increase in the standard error for read.

Now, let's try a model with **read**, **math** and **socst** as predictors. First, we will run a standard OLS regression.

```
regress write read math socst female

  Source |       SS       df       MS                  Number of obs =
200
---------+------------------------------               F(  4,   195) =
64.37
   Model |  10173.7036     4  2543.42591               Prob > F      =
0.0000
Residual |  7705.17137   195  39.5136993               R-squared     =
0.5690
---------+------------------------------               Adj R-squared =
0.5602
   Total |   17878.875   199   89.843593               Root MSE      =
6.286

------------------------------------------------------------------------
-------
   write |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
    read |   .2065341   .0640006      3.227   0.001       .0803118
.3327563
    math |   .3322639   .0651838      5.097   0.000       .2037082
.4608195
   socst |   .2413236   .0547259      4.410   0.000        .133393
.3492542
  female |   5.006263   .8993625      5.566   0.000       3.232537
6.77999
   _cons |   9.120717   2.808367      3.248   0.001       3.582045
14.65939
------------------------------------------------------------------------
-------
```

Now, let's try to account for the measurement error by using the following reliabilities: **read** - .9, **math** - .9, **socst** - .8.

```
eivreg write read math socst female, r(read .9 math .9 socst .8)

               assumed                            errors-in-variables
regression
variable     reliability
------------------------                               Number of obs =
200
    read       0.9000                                  F(  4,   195) =
70.17
    math       0.9000                                  Prob > F      =
0.0000
```

```
   socst       0.8000                                  R-squared     =
0.6047
       *       1.0000                                  Root MSE      =
6.02062

------------------------------------------------------------------------
-------
   write |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
    read |   .1506668   .0936571      1.609   0.109      -.0340441
.3353776
    math |    .350551   .0850704      4.121   0.000       .1827747
.5183273
   socst |   .3327103   .0876869      3.794   0.000        .159774
.5056467
  female |   4.852501   .8730646      5.558   0.000        3.13064
6.574363
   _cons |    6.37062   2.868021      2.221   0.027       .7142973
12.02694
------------------------------------------------------------------------
-------
```

Note that the overall F and R^2 went up, but that the coefficient for read is no longer statistically significant.

4.5 Multiple Equation Regression Models

If a dataset has enough variables we may want to estimate more than one regression model. For example, we may want to predict y1 from x1 and also predict y2 from x2. Even though there are no variables in common these two models are not independent of one another because the data come from the same subjects. This is an example of one type of multiple equation regression known as seemingly unrelated regression. We can estimate the coefficients and obtain standard errors taking into account the correlated errors in the two models. An important feature of multiple equation models is that we can test predictors across equations.

Another example of multiple equation regression is if we wished to predict y1, y2 and y3 from x1 and x2. This is a three equation system, known as multivariate regression, with the same predictor variables for each model. Again, we have the capability of testing coefficients across the different equations.

Multiple equation models are a powerful extension to our data analysis tool kit.

4.5.1 Seemingly Unrelated Regression

Let's continue using the **hsb2** data file to illustrate the use of seemingly unrelated regression. You can load it into memory again if it has been cleared out.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2
(highschool and beyond (200 cases))
```

This time let's look at two regression models.

```
science = math female
write   = read female
```

It is the case that the errors (residuals) from these two models would be correlated. This would be true even if the predictor female were not found in both models. The errors would be correlated because all of the values of the variables are collected on the same set of observations. This is a situation tailor made for seemingly unrelated regression using the **sureg** command. Here is our first model using OLS.

```
regress science math female

<some output omitted>
------------------------------------------------------------------------------
 science |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    math |   .6631901   .0578724     11.460   0.000        .549061    .7773191
  female |  -2.168396   1.086043     -1.997   0.047      -4.310159    -.026633
   _cons |   18.11813   3.167133      5.721   0.000        11.8723    24.36397
------------------------------------------------------------------------------
```

And here is our second model using OLS.

```
regress write read female

<some output omitted>
------------------------------------------------------------------------------
   write |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    read |   .5658869   .0493849     11.459   0.000        .468496    .6632778
  female |   5.486894   1.014261      5.410   0.000        3.48669    7.487098
   _cons |   20.22837   2.713756      7.454   0.000       14.87663    25.58011
------------------------------------------------------------------------------
```

With the **sureg** command we can estimate both models simultaneously while accounting for the correlated errors at the same time, leading to efficient estimates of the coefficients and standard errors. By including the **corr** option with **sureg** we can also obtain an estimate of the correlation between the errors of the two models. Note that both the estimates of the coefficients and their standard errors are different from the OLS model estimates shown above. The bottom of the output provides a Breusch-Pagan test of whether the residuals from the two equations are independent (in this case, we would say the residuals were not independent, p=0.0407).

```
sureg (science math female) (write read female), corr

Seemingly unrelated regression
------------------------------------------------------------------
Equation      Obs  Parms        RMSE    "R-sq"       Chi2        P
------------------------------------------------------------------
science       200      2    7.595793    0.4085   125.4142   0.0000
write         200      2    7.085844    0.4383   144.2683   0.0000
------------------------------------------------------------------------
-------
         |      Coef.   Std. Err.       z     P>|z|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
science  |
    math |   .6251409   .0570948     10.949   0.000       .5132373
.7370446
  female |  -2.189344   1.077862     -2.031   0.042      -4.301914   -
.0767744
   _cons |   20.13265   3.125775      6.441   0.000       14.00624
26.25905
---------+--------------------------------------------------------------
-------
write    |
    read |   .5354838   .0487212     10.991   0.000       .4399919
.6309757
  female |   5.453748   1.006609      5.418   0.000        3.48083
7.426665
   _cons |   21.83439    2.67851      8.152   0.000        16.5846
27.08417
------------------------------------------------------------------------
-------

Correlation matrix of residuals:

         science    write
science   1.0000
  write   0.1447   1.0000

Breusch-Pagan test of independence: chi2(1) =     4.188, Pr = 0.0407
```

Now that we have estimated our models let's test the predictor variables. The test for **female** combines information from both models. The tests for **math** and **read** are actually equivalent to the z-tests above except that the results are displayed as chi-square tests.

```
test female

 ( 1)  [science]female = 0.0
 ( 2)  [write]female = 0.0

           chi2(  2) =   37.45
```

```
          Prob > chi2 =    0.0000
```

test math

```
 ( 1)  [science]math = 0.0

            chi2(  1) =  119.88
          Prob > chi2 =    0.0000
```

test read

```
 ( 1)  [write]read = 0.0

            chi2(  1) =  120.80
          Prob > chi2 =    0.0000
```

Now, let's estimate 3 models where we use the same predictors in each model as shown below.

```
read  = female prog1 prog3
write = female prog1 prog3
math  = female prog1 prog3
```

If you no longer have the dummy variables for **prog**, you can recreate them using the tabulate command.

tabulate prog, gen(prog)

Let's first estimate these three models using 3 OLS regressions.

regress read female prog1 prog3

```
<some output omitted>
------------------------------------------------------------------------
-------
    read |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
  female |  -1.208582   1.327672     -0.910   0.364      -3.826939
1.409774
   prog1 |   -6.42937   1.665893     -3.859   0.000      -9.714746   -
3.143993
   prog3 |  -9.976868   1.606428     -6.211   0.000      -13.14497   -
6.808765
   _cons |    56.8295   1.170562     48.549   0.000       54.52099
59.13802
------------------------------------------------------------------------
-------
```

regress write female prog1 prog3

```
<some output omitted>
------------------------------------------------------------------------
-------
```

```
   write |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
  female |   4.771211   1.181876      4.037   0.000       2.440385
7.102037
   prog1 |  -4.832929   1.482956     -3.259   0.001      -7.757528   -
1.908331
   prog3 |  -9.438071   1.430021     -6.600   0.000      -12.25827   -
6.617868
   _cons |   53.62162   1.042019     51.459   0.000       51.56661
55.67662
-----------------------------------------------------------------------
-------
```

```
regress math female prog1 prog3
```

```
<some output omitted>
-----------------------------------------------------------------------
-------
    math |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
  female |  -.6737673   1.176059     -0.573   0.567      -2.993122
1.645587
   prog1 |  -6.723945   1.475657     -4.557   0.000      -9.634149    -
3.81374
   prog3 |  -10.32168   1.422983     -7.254   0.000        -13.128   -
7.515352
   _cons |   57.10551    1.03689     55.074   0.000       55.06062
59.1504
-----------------------------------------------------------------------
-------
```

These regressions provide fine estimates of the coefficients and standard errors but these results assume the residuals of each analysis are completely independent of the others. Also, if we wish to test **female**, we would have to do it three times and would not be able to combine the information from all three tests into a single overall test.

Now let's use **sureg** to estimate the same models. Since all 3 models have the same predictors, we can use the syntax as shown below which says that **read**, **write** and **math** will each be predicted by **female**, **prog1** and **prog3**. Note that the coefficients are identical in the OLS results above and the **sureg** results below, however the standard errors are different, only slightly, due to the correlation among the residuals in the multiple equations.

```
sureg (read write math = female prog1 prog3), corr
```

```
Seemingly unrelated regression
------------------------------------------------------------------
Equation      Obs  Parms        RMSE    "R-sq"       Chi2        P
------------------------------------------------------------------
read          200      3    9.254765    0.1811   44.24114   0.0000
write         200      3    8.238468    0.2408   63.41908   0.0000
```

```
math          200      3    8.197921    0.2304   59.88479   0.0000
-----------------------------------------------------------------------
-------
         |      Coef.   Std. Err.       z     P>|z|       [95% Conf.
Interval]
---------+-------------------------------------------------------------
-------
read     |
  female |  -1.208582   1.314328     -0.920   0.358      -3.784618
1.367454
   prog1 |   -6.42937    1.64915     -3.899   0.000      -9.661645   -
3.197095
   prog3 |  -9.976868   1.590283     -6.274   0.000      -13.09377   -
6.859971
   _cons |    56.8295   1.158797     49.042   0.000        54.5583
59.1007
---------+-------------------------------------------------------------
-------
write    |
  female |   4.771211   1.169997      4.078   0.000       2.478058
7.064363
   prog1 |  -4.832929   1.468051     -3.292   0.001      -7.710257   -
1.955602
   prog3 |  -9.438071   1.415648     -6.667   0.000      -12.21269   -
6.663451
   _cons |   53.62162   1.031546     51.982   0.000       51.59982
55.64341
---------+-------------------------------------------------------------
-------
math     |
  female |  -.6737673   1.164239     -0.579   0.563      -2.955634
1.608099
   prog1 |  -6.723945   1.460826     -4.603   0.000      -9.587111   -
3.860778
   prog3 |  -10.32168   1.408681     -7.327   0.000      -13.08264   -
7.560711
   _cons |   57.10551   1.026469     55.633   0.000       55.09367
59.11735
-----------------------------------------------------------------------
-------

Correlation matrix of residuals:

         read   write    math
 read  1.0000
write  0.5519  1.0000
 math  0.5774  0.5577  1.0000

Breusch-Pagan test of independence: chi2(3) =   189.811, Pr = 0.0000
```

In addition to getting more appropriate standard errors, **sureg** allows us to test the effects of the predictors across the equations. We can test the hypothesis that the coefficient for **female** is 0 for all three outcome variables, as shown below.

```
test female
```

```
( 1)  [read]female = 0.0
( 2)  [write]female = 0.0
( 3)  [math]female = 0.0

          chi2(  3) =   35.59
        Prob > chi2 =    0.0000
```

We can also test the hypothesis that the coefficient for **female** is 0 for just **read** and **math**. Note that **[read]female** means the coefficient for **female** for the outcome variable **read**.

```
test [read]female [math]female

 ( 1)  [read]female = 0.0
 ( 2)  [math]female = 0.0

          chi2(  2) =    0.85
        Prob > chi2 =    0.6541
```

We can also test the hypothesis that the coefficients for **prog1** and **prog3** are 0 for all three outcome variables, as shown below.

```
test prog1 prog3

 ( 1)  [read]prog1 = 0.0
 ( 2)  [write]prog1 = 0.0
 ( 3)  [math]prog1 = 0.0
 ( 4)  [read]prog3 = 0.0
 ( 5)  [write]prog3 = 0.0
 ( 6)  [math]prog3 = 0.0

          chi2(  6) =   72.45
        Prob > chi2 =    0.0000
```

4.5.2 Multivariate Regression

Let's now use multivariate regression using the **mvreg** command to look at the same analysis that we saw in the **sureg** example above, estimating the following 3 models.

```
read  = female prog1 prog3
write = female prog1 prog3
math  = female prog1 prog3
```

If you don't have the **hsb2** data file in memory, you can use it below and then create the dummy variables for **prog1** - **prog3**.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2
tabulate prog, gen(prog)
<output omitted>
```

Below we use **mvreg** to predict **read**, **write** and **math** from **female**, **prog1** and **prog3**. Note that the top part of the output is similar to the **sureg** output in that it gives an overall summary of the model for each outcome variable, however the results are somewhat different and the **sureg** uses a **Chi-Square** test for the overall fit of the model, and **mvreg** uses an **F-test**. The lower part of the output appears similar to the **sureg** output; however, when you compare the standard errors you see that the results are not the same. These standard errors correspond to the OLS standard errors, so these results below do not take into account the correlations among the residuals (as do the **sureg** results).

```
mvreg read write math = female prog1 prog3

Equation      Obs  Parms        RMSE    "R-sq"          F         P
------------------------------------------------------------------
read          200      4    9.348725    0.1811   14.45211   0.0000
write         200      4     8.32211    0.2408    20.7169   0.0000
math          200      4    8.281151    0.2304   19.56237   0.0000

------------------------------------------------------------------------
-------
         |      Coef.   Std. Err.       t     P>|t|       [95% Conf.
Interval]
---------+--------------------------------------------------------------
-------
read     |
  female |  -1.208582   1.327672     -0.910   0.364      -3.826939
1.409774
   prog1 |   -6.42937   1.665893     -3.859   0.000      -9.714746   -
3.143993
   prog3 |  -9.976868   1.606428     -6.211   0.000      -13.14497   -
6.808765
   _cons |    56.8295   1.170562     48.549   0.000       54.52099
59.13802
---------+--------------------------------------------------------------
-------
write    |
  female |   4.771211   1.181876      4.037   0.000       2.440385
7.102037
   prog1 |  -4.832929   1.482956     -3.259   0.001      -7.757528   -
1.908331
   prog3 |  -9.438071   1.430021     -6.600   0.000      -12.25827   -
6.617868
   _cons |   53.62162   1.042019     51.459   0.000       51.56661
55.67662
---------+--------------------------------------------------------------
-------
math     |
  female |  -.6737673   1.176059     -0.573   0.567      -2.993122
1.645587
   prog1 |  -6.723945   1.475657     -4.557   0.000      -9.634149    -
3.81374
   prog3 |  -10.32168   1.422983     -7.254   0.000        -13.128   -
7.515352
   _cons |   57.10551    1.03689     55.074   0.000       55.06062
59.1504
------------------------------------------------------------------------
-------
```

Now, let's **test female**. Note, that **female** was statistically significant in only one of the three equations. Using the test command after **mvreg** allows us to test **female** across all three equations simultaneously. And, guess what? It is significant. This is consistent with what we found using **sureg** (except that **sureg** did this test using a **Chi-Square** test).

```
test female

 ( 1)  [read]female = 0.0
 ( 2)  [write]female = 0.0
 ( 3)  [math]female = 0.0

       F(  3,   196) =   11.63
            Prob > F =    0.0000
```

We can also test **prog1** and **prog3**, both separately and combined. Remember these are multivariate tests.

```
test prog1

 ( 1)  [read]prog1 = 0.0
 ( 2)  [write]prog1 = 0.0
 ( 3)  [math]prog1 = 0.0

       F(  3,   196) =    7.72
            Prob > F =    0.0001

test prog3

 ( 1)  [read]prog3 = 0.0
 ( 2)  [write]prog3 = 0.0
 ( 3)  [math]prog3 = 0.0

       F(  3,   196) =   21.47
            Prob > F =    0.0000

test prog1 prog3

 ( 1)  [read]prog1 = 0.0
 ( 2)  [write]prog1 = 0.0
 ( 3)  [math]prog1 = 0.0
 ( 4)  [read]prog3 = 0.0
 ( 5)  [write]prog3 = 0.0
 ( 6)  [math]prog3 = 0.0

       F(  6,   196) =   11.83
            Prob > F =    0.0000
```

Many researchers familiar with traditional multivariate analysis may not recognize the tests above. They don't see Wilks' Lambda, Pillai's Trace or the Hotelling-Lawley Trace statistics, statistics that they are familiar with. It is possible to obtain these statistics using the **mvtest** command written by David E. Moore of the University of Cincinnati. **mvtest** , which UCLA updated to work with Stata 6 and above, can be downloaded over the internet like this.

```
net from http://www.ats.ucla.edu/stat/stata/ado/analysis
net install mvtest
```

Now that we have downloaded it, we can use it like this.

```
mvtest female

                     MULTIVARIATE TESTS OF SIGNIFICANCE


Multivariate Test Criteria and Exact F Statistics for
the Hypothesis of no Overall "female" Effect(s)

                           S=1    M=.5    N=96

Test                          Value          F       Num DF     Den DF
Pr > F
Wilks' Lambda              0.84892448    11.5081          3   194.0000
0.0000
Pillai's Trace             0.15107552    11.5081          3   194.0000
0.0000
Hotelling-Lawley Trace     0.17796108    11.5081          3   194.0000
0.0000
```

```
mvtest prog1 prog3

                     MULTIVARIATE TESTS OF SIGNIFICANCE


Multivariate Test Criteria and Exact F Statistics for
the Hypothesis of no Overall "prog1 prog3" Effect(s)

                           S=2    M=0    N=96

Test                          Value          F       Num DF     Den DF
Pr > F
Wilks' Lambda              0.73294667    10.8676          6   388.0000
0.0000
Pillai's Trace             0.26859190    10.0834          6   390.0000
0.0000
Hotelling-Lawley Trace     0.36225660    11.6526          6   386.0000
0.0000
```

We will end with an **mvtest** including all of the predictor variables. This is an overall multivariate test of the model.

```
mvtest female prog1 prog3

                     MULTIVARIATE TESTS OF SIGNIFICANCE


Multivariate Test Criteria and Exact F Statistics for
the Hypothesis of no Overall "female prog1 prog3" Effect(s)
```

```
                              S=3    M=-.5    N=96

Test                              Value          F       Num DF     Den DF
Pr > F
Wilks' Lambda                0.62308940    11.2593          9   472.2956
0.0000
Pillai's Trace               0.41696769    10.5465          9   588.0000
0.0000
Hotelling-Lawley Trace       0.54062431    11.5734          9   578.0000
0.0000
```

The **sureg** and **mvreg** commands both allow you to test multi-equation models while taking into account the fact that the equations are not independent. The **sureg** command allows you to get estimates for each equation which adjust for the non-independence of the equations, and it allows you to estimate equations which don't necessarily have the same predictors. By contrast, **mvreg** is restricted to equations that have the same set of predictors, and the estimates it provides for the individual equations are the same as the OLS estimates. However, **mvreg** (especially when combined with **mvtest**) allows you to perform more traditional multivariate tests of predictors.

4.6 Summary

This chapter has covered a variety of topics that go beyond ordinary least squares regression, but there still remain a variety of topics we wish we could have covered, including the analysis of survey data, dealing with missing data, panel data analysis, and more. And, for the topics we did cover, we wish we could have gone into even more detail. One of our main goals for this chapter was to help you be aware of some of the techniques that are available in Stata for analyzing data that do not fit the assumptions of OLS regression and some of the remedies that are possible. If you are a member of the UCLA research community, and you have further questions, we invite you to use our consulting services to discuss issues specific to your data analysis.

4.7 Self Assessment

1. Use the **crime** data file that was used in chapter 2 (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/crime) and look at a regression model predicting **murder** from **pctmetro**, **poverty**, **pcths** and **single** using OLS and make a **avplots** and a **lvr2plot** following the regression. Are there any states that look worrisome? Repeat this analysis using regression with robust standard errors and show **avplots** for the analysis. Repeat the analysis using robust regression and make a manually created **lvr2plot**. Also run the results using **qreg**. Compare the results of the different analyses. Look at the weights from the robust regression and comment on the weights.

2. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) pretend that 550 is the lowest score that a school could achieve on **api00**, i.e., create a new variable with the **api00** score and recode it such that any score of 550 or below becomes 550. Use **meals**, **ell** and **emer** to predict api scores using 1) OLS to predict the original api score (before recoding) 2) OLS to predict the recoded score where 550 was the lowest value, and 3) using **tobit** to predict the recoded api score indicating the lowest value is 550. Compare the results of these analyses.

3. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) pretend that only schools with api scores of 550 or higher were included in the sample. Use **meals**, **ell** and **emer** to predict api scores using 1) OLS to predict api from the full set of observations, 2) OLS to predict api using just the observations with api scores of 550 or higher, and 3) using **truncreg** to predict api using just the observations where api is 550 or higher. Compare the results of these analyses.

4. Using the **hsb2** data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2) predict **read** from **science**, **socst**, **math** and **write**. Use the **testparm** and **test** commands to test the equality of the coefficients for **science**, **socst** and **math**. Use **cnsreg** to estimate a model where these three parameters are equal.

5. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) consider the following 2 regression equations.

```
api00 = meals ell emer
api99 = meals ell emer
```

Estimate the coefficients for these predictors in predicting **api00** and **api99** taking into account the non-independence of the schools. Test the overall contribution of each of the predictors in jointly predicting api scores in these two years. Test whether the contribution of **emer** is the same for **api00** and **api99**.

Click here for our answers to these self assessment questions.

4.8 For more information

- Stata Manuals
  - **[R] rreg**
  - **[R] qreg**
  - **[R] cnsreg**
  - **[R] tobit**
  - **[R] truncreg**
  - **[R] eivreg**
  - **[R] sureg**
  - **[R] mvreg**
  - **[U] 23 Estimation and post-estimation commands**
  - **[U] 29 Overview of model estimation in Stata**
- Web Links
  - How standard errors with cluster() can be smaller than those without
  - Advantages of the robust variance estimator
  - How to obtain robust standard errors for tobit
  - Pooling data in linear regression

Chapter 4 - Self Assessment

1. Use the **crime** data file that was used in chapter 2 (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/crime) and look at a regression model predicting **murder** from **pctmetro poverty pcths** and **single** using OLS and make a **avplots** and a **lvr2plot** following the regression. Are there any states that look worrisome? Repeat this analysis using regression with robust standard errors and show **avplots** for the analysis. Repeat the analysis using robust regression and make a manually created **lvr2plot**. Also run the results using **qreg**. Compare the results of the different analyses. Look at the weights from the robust regression and comment on the weights.

2. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) pretend that 550 is the lowest score that a school could achieve on **api00**, i.e. create a new variable with the **api00** score and recode it such that any score of 550 or below becomes 550. Use **meals ell** and **emer** to predict api scores using 1) OLS to predict the original api score (before recoding) 2) OLS to predict the recoded score where 550 was the lowest value, and 3) using **tobit** to predict the recoded api score indicating the lowest value is 550. Compare the results of these analyses.

3. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) pretend that only schools with api scores of 550 or higher were included in the sample. Use **meals ell** and **emer** to predict api scores using 1) OLS to predict api from the full set of observations, 2) OLS to predict api using just the observations with api scores of 550 or higher, and 3) using **truncreg** to predict api using just the observations where api is 550 or higher. Compare the results of these analyses.

4. Using the **hsb2** data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2) predict **read** from **science**, **socst**, **math** and **write**. Use the **testparm** and **test** commands to test the equality of the coefficients for **science**, **socst** and **math**. Use **cnsreg** to estimate a model where these three parameters are equal.

5. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) consider the following 2 regression equations.

```
api00 = meals ell emer
api99 = meals ell emer
```

Estimate the coefficients for these predictors in predicting **api00** and **api99** taking into account the non-independence of the schools. Test the overall contribution of each of the predictors in jointly predicting api scores in these two years. Test whether the contribution of **emer** is the same for **api00** and **api99**.

Chapter 4: Answers to Excersises

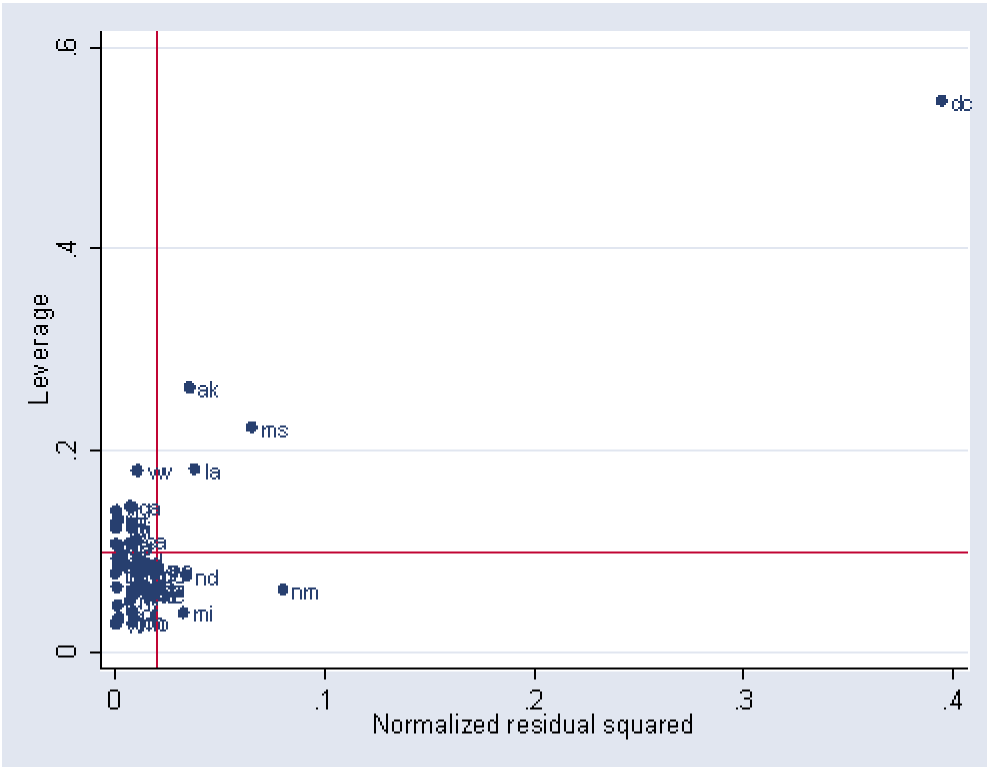
1. Use the **crime** data file that was used in chapter 2 (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/crime) and look at a regression model predicting **murder** from **pctmetro**, **poverty**, **pcths** and **single** using OLS and make a **avplots** and a **lvr2plot** following the regression. Are there any states that look worrisome? Repeat this analysis using regression with robust standard errors and show **avplots** for the analysis. Repeat the analysis using robust regression and make a manually created **lvr2plot**. Also run the results using **qreg**. Compare the results of the different analyses. Look at the weights from the robust regression and comment on the weights.

Answer 1.
First, consider the OLS regression predicting **murder** from **pctmetro**, **poverty**, **pcths** and **single**.

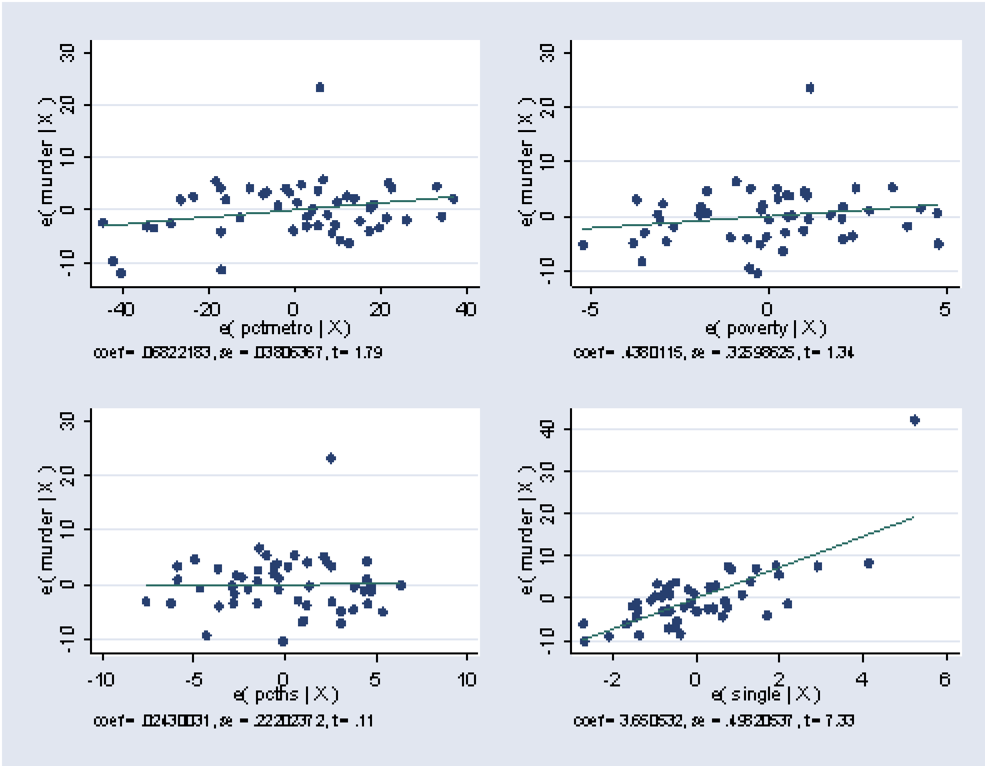
```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/crime , clear
(crime data from agresti & finlay - 1997)
regress murder pctmetro poverty pcths single
      Source |       SS       df       MS              Number of obs =
51
-------------+------------------------------           F(  4,    46) =
37.90
       Model |  4406.42207     4  1101.60552           Prob > F      =
0.0000
    Residual |  1336.89947    46  29.0630319           R-squared     =
0.7672
-------------+------------------------------           Adj R-squared =
0.7470
       Total |  5743.32154    50  114.866431           Root MSE      =
5.391

-----------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------
-------
    pctmetro |   .0682218   .0380637     1.79   0.080    -.0083964
.14484
     poverty |   .4380115   .3259862     1.34   0.186    -.2181648
1.094188
       pcths |   .0243003   .2220237     0.11   0.913    -.4226102
.4712109
      single |   3.650532   .4982054     7.33   0.000     2.647697
4.653367
       _cons |  -45.31188   19.39747    -2.34   0.024    -84.35697   -
6.266792
-----------------------------------------------------------------------
-------
```

These results suggest that **single** is the only predictor significantly related to number of murders in a state. Let's look at the **lvr2plot** for this analysis. Washington DC looks like it has both a very high leverage and a very high residual.

```
. lvr2plot, mlabel(state)
```




```
. avplots
```



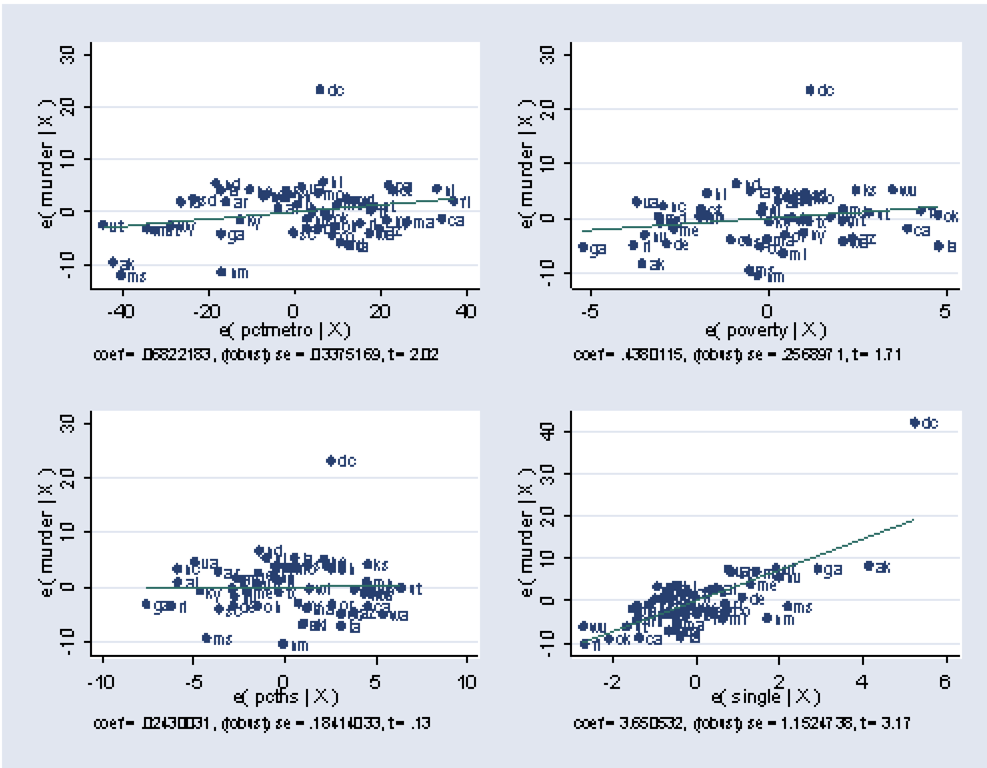
Let's consider the same analysis using robust standard errors. The results are largely the same, except that the p value for **pctmetro** fell from 0.08 to 0.049, which would then make it a significant predictor, however we would be somewhat skeptical of this particular result without further investigation.

```
regress murder pctmetro poverty pcths single, robust
Regression with robust standard errors                 Number of obs =
51
                                                       F(  4,    46) =
7.20
                                                       Prob > F      =
0.0001
                                                       R-squared     =
0.7672
                                                       Root MSE      =
5.391

------------------------------------------------------------------------
-------
             |               Robust
      murder |      Coef.   Std. Err.      t     P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    pctmetro |   .0682218   .0337517     2.02   0.049     .0002832
.1361604
     poverty |   .4380115   .2568971     1.71   0.095    -.0790955
.9551185
```

```
      pcths |   .0243003   .1841403     0.13   0.896    -.3463549
.3949556
     single |   3.650532   1.152474     3.17   0.003     1.330723
5.970341
      _cons |  -45.31188   25.39531    -1.78   0.081    -96.42999
5.806231
------------------------------------------------------------------------
-------
```

Stata allows us to compute the residual for this analysis but will not allow us to compute the leverage (hat) value. So instead of showing a **lvr2plot** let's look at the **avplots** for this analysis.

```
. avplots , mlabel(state)
```



As you can see, we still have an observation that sticks out from the rest, and this is Washington DC. This is especially pronounced for the lower right graph for **single** where DC would seem to have very strong leverage to influence the coefficient for single.

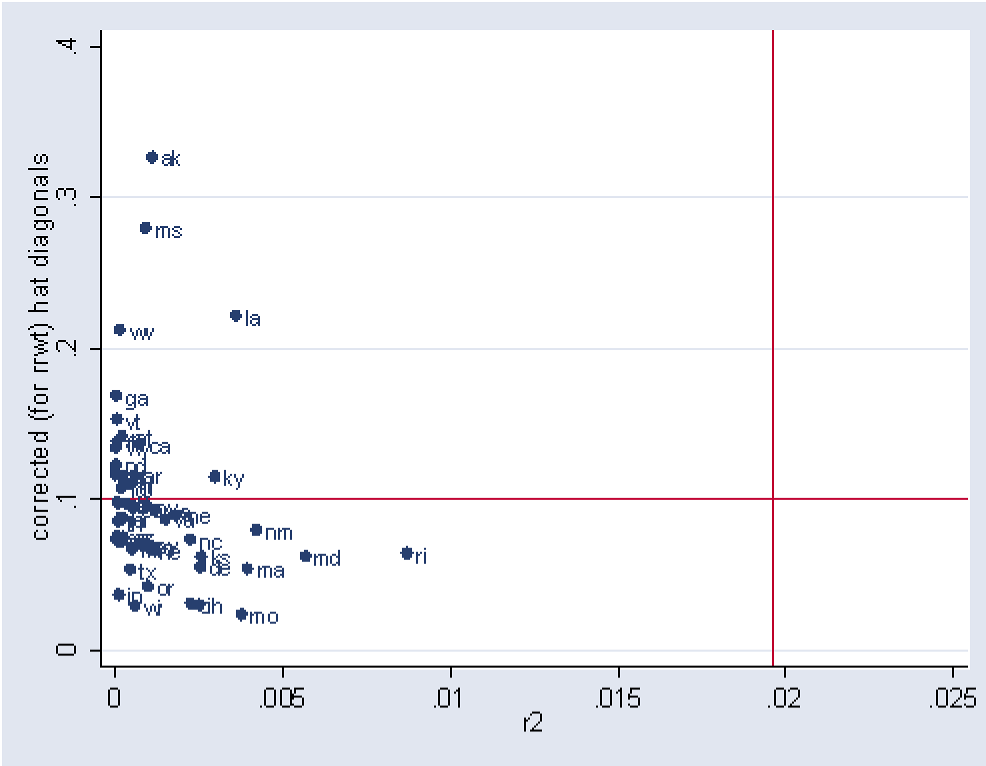
Now, let's look at the analysis using robust regression and save the weights, calling them **rrwt**.

```
rreg murder pctmetro poverty pcths single, genwt(rrwt)
   Huber iteration 1:  maximum difference in weights = .44857261
   Huber iteration 2:  maximum difference in weights = .0399983
Biweight iteration 3:  maximum difference in weights = .15321379
Biweight iteration 4:  maximum difference in weights = .00973214

Robust regression estimates                             Number of obs =
50
```

```
                                                       F(  4,    45) =
35.25
                                                       Prob > F      =
0.0000

------------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    pctmetro |   .0535439   .0146555     3.65   0.001     .0240262
.0830615
     poverty |    .182561   .1259505     1.45   0.154    -.0711163
.4362383
       pcths |  -.2245853   .0863452    -2.60   0.013    -.3984936   -
.0506771
      single |   1.392942   .2355845     5.91   0.000     .9184503
1.867434
       _cons |   2.888033   7.945302     0.36   0.718    -13.11463
18.89069
------------------------------------------------------------------------
-------
```

If you try the **avplots** command, this command is not available after **rreg** and the **lvr2plot** is not available either. But we can manually create the residual and hat values and create an **lvr2plot** of our own, see below.

```
predict r, r
predict h, hat
generate r2=r^2
sum r2
<output omitted>
replace r2 = r2/r(sum)
summarize r2
<output omitted>
local rm = r(mean)
summarize h
<output omitted>
local hm = r(mean)
graph twoway scatter h r2 if state ~= "dc", yline(`hm') xline(`rm')
mlabel(state) xlabel(0(.005).025)
```



As you see above, using the robust regression, none of the observations are jointly high in leverage and their residual values. Let's recap the **regress** results and the **rreg** results below and compare them.

```
regress murder pctmetro poverty pcths single
      Source |       SS       df       MS              Number of obs =
51
-------------+------------------------------           F(  4,    46) =
37.90
       Model |  4406.42207     4  1101.60552           Prob > F      =
0.0000
    Residual |  1336.89947    46  29.0630319           R-squared     =
0.7672
-------------+------------------------------           Adj R-squared =
0.7470
       Total |  5743.32154    50  114.866431           Root MSE      =
5.391

------------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    pctmetro |   .0682218   .0380637     1.79   0.080    -.0083964
.14484
     poverty |   .4380115   .3259862     1.34   0.186    -.2181648
1.094188
```

```
       pcths |   .0243003   .2220237     0.11   0.913    -.4226102
.4712109
      single |   3.650532   .4982054     7.33   0.000     2.647697
4.653367
       _cons |  -45.31188   19.39747    -2.34   0.024    -84.35697   -
6.266792
----------------------------------------------------------------------
-------
```

```
rreg murder pctmetro poverty pcths single
   Huber iteration 1:  maximum difference in weights = .44857261
   Huber iteration 2:  maximum difference in weights = .0399983
Biweight iteration 3:  maximum difference in weights = .15321379
Biweight iteration 4:  maximum difference in weights = .00973214

Robust regression estimates                            Number of obs =
50
                                                       F(  4,    45) =
35.25
                                                       Prob > F      =
0.0000

----------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+--------------------------------------------------------
-------
    pctmetro |   .0535439   .0146555     3.65   0.001     .0240262
.0830615
     poverty |    .182561   .1259505     1.45   0.154    -.0711163
.4362383
       pcths |  -.2245853   .0863452    -2.60   0.013    -.3984936   -
.0506771
      single |   1.392942   .2355845     5.91   0.000     .9184503
1.867434
       _cons |   2.888033   7.945302     0.36   0.718    -13.11463
18.89069
----------------------------------------------------------------------
-------
```

The results are consistent for **poverty** and for **single**, where **poverty** was not significant in both analyses and **single** was significant in both analyses. However, the results for **pctmetro** and **pcths** were both not significant in the OLS analysis and were significant in the robust regression anlaysis.

Let's look at the weights used in the robust regression to further understand why the results were so different. Note that the weight for **dc** is **.** meaning that it was eliminated from the analysis entirely (because it had such a high residual). Also, **ri** was weighted by less than half.

```
hilo rrwt state
10 lowest and highest observations on rrwt

      rrwt      state
46982663         ri
```

```
62949383         md
716977         nm
73472243         ma
74565543         mo
75750112         la
79708217         ky
82324958         ks
82552144         de
82728266         il

      rrwt      state
99592844         sd
99639177         pa
99799356         fl
99811845         vt
99838103         ga
99863411         nh
99981867         wy
99986937         nd
99991851         ok
dc
```

In our analyses in chapter 2 (involving different variables) we found **dc** to be a very serious outlier and decided that it should be excluded because it is not a state. If we investigated further into these variables we may reach the same conclusion and decide that **dc** should be excluded. If we did, we could try using OLS regression like this. These results are quite similar to the **rreg** results. The benefits of **rreg** is that it deals not only with the serious problems (like **dc** being a very bad outlier) but also minor problems as well.

```
regress murder pctmetro poverty pcths single if state != "dc"
      Source |       SS       df       MS              Number of obs =
50
-------------+------------------------------           F(  4,    45) =
39.88
       Model |  606.611746     4  151.652936           Prob > F      =
0.0000
    Residual |  171.137027    45  3.80304505           R-squared     =
0.7800
-------------+------------------------------           Adj R-squared =
0.7604
       Total |  777.748773    49  15.8724239           Root MSE      =
1.9501

------------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
    pctmetro |   .0534333    .013795     3.87   0.000     .0256488
.0812178
     poverty |   .2237151   .1185554     1.89   0.066    -.0150679
.462498
       pcths |  -.1938711   .0812756    -2.39   0.021    -.3575685   -
.0301737
```

```
      single |   1.388337   .2217525     6.26   0.000     .9417051
1.83497
       _cons |  -.0044014   7.478803    -0.00   1.000    -15.06748
15.05868
-----------------------------------------------------------------------
-------
```

Let's try running the results using **qreg** and compare them with **rreg**.

```
qreg murder pctmetro poverty pcths single
Iteration  1:  WLS sum of weighted deviations =  187.90652

Iteration  1: sum of abs. weighted deviations =  177.16784
Iteration  2: sum of abs. weighted deviations =  167.01302
Iteration  3: sum of abs. weighted deviations =  128.40282
Iteration  4: sum of abs. weighted deviations =  125.28249
Iteration  5: sum of abs. weighted deviations =    124.226
Iteration  6: sum of abs. weighted deviations =  122.93248
Iteration  7: sum of abs. weighted deviations =   122.6427
Iteration  8: sum of abs. weighted deviations =  122.40488
Iteration  9: sum of abs. weighted deviations =  122.03476
Iteration 10: sum of abs. weighted deviations =  122.03096

Median regression                                    Number of obs =
51
  Raw sum of deviations    235.3 (about 6.8000002)
  Min sum of deviations  122.031                     Pseudo R2     =
0.4814

-----------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------
-------
    pctmetro |   .0527879   .0226177     2.33   0.024     .0072608
.098315
     poverty |   .0908506   .1831176     0.50   0.622    -.2777461
.4594473
       pcths |  -.2686652   .1284197    -2.09   0.042    -.5271606   -
.0101697
      single |   1.796151   .2859057     6.28   0.000     1.220652
2.371649
       _cons |   3.524669   11.34322     0.31   0.757    -19.30806
26.35739
-----------------------------------------------------------------------
-------
rreg murder pctmetro poverty pcths single
   Huber iteration 1:  maximum difference in weights = .44857261
   Huber iteration 2:  maximum difference in weights = .0399983
Biweight iteration 3:  maximum difference in weights = .15321379
Biweight iteration 4:  maximum difference in weights = .00973214

Robust regression estimates                            Number of obs =
50
```

```
                                                       F(  4,    45) = 
35.25
                                                       Prob > F      = 
0.0000

------------------------------------------------------------------------
-------
      murder |      Coef.   Std. Err.      t    P>|t|     [95% Conf. 
Interval]
-------------+----------------------------------------------------------
-------
    pctmetro |   .0535439   .0146555     3.65   0.001     .0240262    
.0830615
     poverty |    .182561   .1259505     1.45   0.154    -.0711163    
.4362383
       pcths |  -.2245853   .0863452    -2.60   0.013    -.3984936   -
.0506771
      single |   1.392942   .2355845     5.91   0.000     .9184503    
1.867434
       _cons |   2.888033   7.945302     0.36   0.718    -13.11463    
18.89069
------------------------------------------------------------------------
-------
```

While the coefficients do not always match up, the variables that were significant in the **qreg** are also significant in the **rreg** and likewise for the non-significant variables. Even though these techniques use different strategies for resisting the influence of very deviant observations, they both arrive at the same conclusions regarding which variables are significantly related to **murder**, although they do not always agree in the strength of the relationship, i.e. the size of the coefficients.

2. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) pretend that 550 is the lowest score that a school could achieve on **api00**, i.e., create a new variable with the **api00** score and recode it such that any score of 550 or below becomes 550. Use **meals**, **ell** and **emer** to predict api scores using 1) OLS to predict the original api score (before recoding) 2) OLS to predict the recoded score where 550 was the lowest value, and 3) using **tobit** to predict the recoded api score indicating the lowest value is 550. Compare the results of these analyses.

Answer 2.
First, we will use the **elemapi2** data file and create the recoded version of the api score where the lowest value is 550. We will call this value **api00x**.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2 , clear
gen api00x = api00
replace api00x = 550 if api00 <= 550
(122 real changes made)
```

Analysis 1. Now, we will run an OLS regression on the un-recoded version of api.

```
regress api00 meals ell emer
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
673.00
       Model |  6749782.75     3  2249927.58           Prob > F      =
0.0000
    Residual |  1323889.25   396  3343.15467           R-squared     =
0.8360
-------------+------------------------------           Adj R-squared =
0.8348
       Total |  8073672.00   399  20234.7669           Root MSE      =
57.82

-----------------------------------------------------------------------------
-
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
       meals |  -3.159189   .1497371   -21.10   0.000    -3.453568   -
2.864809
         ell |  -.9098732   .1846442    -4.93   0.000    -1.272878   -
.5468678
        emer |  -1.573496    .293112    -5.37   0.000    -2.149746   -
.9972456
       _cons |   886.7033    6.25976   141.65   0.000     874.3967
899.0098
-----------------------------------------------------------------------------
-
```

Analysis 2. Now, we run an OLS regression on the recoded version of api.

regress api00x meals ell emer

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
682.88
       Model |  4567355.46     3  1522451.82           Prob > F      =
0.0000
    Residual |  882862.941   396  2229.45187           R-squared     =
0.8380
-------------+------------------------------           Adj R-squared =
0.8368
       Total |  5450218.40   399  13659.6952           Root MSE      =
47.217

-----------------------------------------------------------------------------
-
      api00x |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
       meals |  -3.010788   .1222786   -24.62   0.000    -3.251184   -
2.770392
         ell |  -.3034092   .1507844    -2.01   0.045    -.5998472   -
.0069713
```

```
        emer |  -.7484733   .2393616    -3.13   0.002    -1.219052    -
.277895
       _cons |     869.31   5.111854   170.06   0.000     859.2602
879.3597
-----------------------------------------------------------------------------
-
```

Analysis 3. And we use **tobit** to perform the analysis indicating that the lowest value possible was 550.

```
tobit api00x meals ell emer  , ll(550)
Tobit estimates                                   Number of obs   =
400
                                                  LR chi2(3)      =
660.74
                                                  Prob > chi2     =
0.0000
Log likelihood = -1581.8117                       Pseudo R2       =
0.1728

-----------------------------------------------------------------------
-------
      api00x |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------
-------
       meals |  -3.145065   .1595799   -19.71   0.000    -3.458792   -
2.831337
         ell |  -.8633529    .212474    -4.06   0.000    -1.281068   -
.4456381
        emer |  -1.470878   .3361215    -4.38   0.000    -2.131678   -
.8100772
       _cons |   885.2395   6.372871   138.91   0.000     872.7107
897.7683
-------------+---------------------------------------------------------
-------
         _se |   57.12718   2.473494           (Ancillary parameter)
-----------------------------------------------------------------------
-------

  Obs. summary:        122  left-censored observations at api00x <=550
278 uncensored observations
```

First, let's compare analysis 1 and 2. When the range in api was restricted in analysis 2, the size of the coefficients dropped due to the restriction in range of the api scores. For example, the coefficient for **ell** dropped from -.9 to -.3 and its significance level changed to 0.045 (nearly not significant from being quite significant). Let's see how well the **tobit** analysis compensated for the restriction in range by comparing analysis #1 and #3. The coefficients are quite similar in these two analyses. The standard errors are slightly larger in the **tobit** analysis leading the t values to be somewhat smaller. Nevertheless, the **tobit** estimates are much more on target than the second OLS analysis on the recoded data.

3. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) pretend that only schools with api scores of 550 or higher were included in the sample. Use **meals ell** and **emer** to predict api scores using 1) OLS to predict api from the full set of observations, 2) OLS to predict api using just the observations with api scores of 550 or higher, and 3) using **truncreg** to predict api using just the observations where api is 550 or higher. Compare the results of these analyses.

Answer 3.
First, we use the **elemapi2** data file and run the analysis on the complete data.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
```

Analysis 1 using all of the data.

```
regress api00 meals ell emer
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
673.00
       Model |  6749782.75     3  2249927.58           Prob > F      =
0.0000
    Residual |  1323889.25   396  3343.15467           R-squared     =
0.8360
-------------+------------------------------           Adj R-squared =
0.8348
       Total |  8073672.00   399  20234.7669           Root MSE      =
57.82

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t     P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
       meals |  -3.159189   .1497371   -21.10   0.000    -3.453568   -
2.864809
         ell |  -.9098732   .1846442    -4.93   0.000    -1.272878   -
.5468678
        emer |  -1.573496    .293112    -5.37   0.000    -2.149746   -
.9972456
       _cons |   886.7033    6.25976   141.65   0.000     874.3967
899.0098
------------------------------------------------------------------------
-------
```

Now let's keep just the schools with api scores of 550 or higher for the next 2 analyses.

```
keep if api00 >= 550
(122 observations deleted)
```

Analysis 2 using OLS on just the schools with api scores of 550 or higher.

```
regress api00 meals ell emer
      Source |       SS       df       MS              Number of obs =
278
-------------+------------------------------           F(  3,   274) =
292.55
       Model |  2268727.43     3  756242.478           Prob > F      =
0.0000
    Residual |  708297.044   274  2585.02571           R-squared     =
0.7621
-------------+------------------------------           Adj R-squared =
0.7595
       Total |  2977024.48   277  10747.3808           Root MSE      =
50.843

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
       meals |  -2.798288   .1600331   -17.49   0.000    -3.113339   -
2.483238
         ell |  -.3584496   .2315111    -1.55   0.123    -.8142161
.0973169
        emer |  -.9417814   .3547208    -2.65   0.008    -1.640106   -
.2434569
       _cons |    868.222   5.880858   147.64   0.000     856.6446
879.7994
------------------------------------------------------------------------
-------
```

Analysis 3 using **truncreg** on just the schools with api scores of 550 or higher.

```
truncreg api00 meals ell emer  , ll(550)
(note: 0 obs. truncated)

Fitting full model:

Iteration 0:   log likelihood = -1467.4296
Iteration 1:   log likelihood = -1460.6163
Iteration 2:   log likelihood = -1460.3638
Iteration 3:   log likelihood = -1460.3636
Iteration 4:   log likelihood = -1460.3636

Truncated regression
Limit:   lower =        550                             Number of obs =
278
         upper =       +inf                             Wald chi2(3)  =
634.48
Log likelihood = -1460.3636                             Prob > chi2   =
0.0000

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      z    P>|z|     [95% Conf.
Interval]
```

```
-------------+----------------------------------------------------------------
-------
eq1          |
       meals |   -2.90758   .1872438   -15.53   0.000    -3.274571   -
2.540589
         ell |  -.8212468   .2983573    -2.75   0.006    -1.406016   -
.2364771
        emer |  -1.446235   .4549632    -3.18   0.001    -2.337946   -
.5545233
       _cons |   879.4212   6.595712   133.33   0.000     866.4939
892.3486
-------------+----------------------------------------------------------------
-------
sigma        |
       _cons |   53.34897   2.545858    20.96   0.000     48.35918
58.33876
------------------------------------------------------------------------------
-------
```

Let's first compare the results of analysis 1 with analysis 2. When the schools with api scores of less than 550 are omitted, the coefficient for **ell** drops from -.9 to .35 and becomes no longer statistically significant. The coefficients for **meals** and **emer** remain significant although they both drop as well.

Now, let's compare analysis 3 using **truncreg** with the original OLS analysis of the complete data. In both of these analyses, all of the variables are significant and the coefficients are quite similar, although the standard errors are larger in the **truncreg**. The **truncreg** did a pretty good job of showing us what the coefficients were in the complete sample based just on the restricted sample.

4. Using the **hsb2** data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2) predict **read** from **science**, **socst**, **math** and **write**. Use the **testparm** and **test** commands to test the equality of the coefficients for **science**, **socst** and **math**. Use **cnsreg** to estimate a model where these three parameters are equal.

Answer 4.
We start by using the **hsb2** data file.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/hsb2 , clear
(highschool and beyond (200 cases))
```

We first run an ordinary regression predicting **read** from **science**, **socst**, **math** and **write**.

```
regress read science socst math  write
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  4,   195) =
69.74
       Model |  12312.7853     4  3078.19634           Prob > F      =
0.0000
    Residual |  8606.63466   195   44.136588           R-squared     =
0.5886
```

```
-------------+------------------------------           Adj R-squared =
0.5801
       Total |    20919.42   199  105.122714           Root MSE      =
6.6435

------------------------------------------------------------------------
-------
        read |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
     science |   .2736751    .064369     4.25   0.000     .1467263
.4006238
       socst |    .273267   .0574246     4.76   0.000      .160014
.38652
        math |   .3028976    .072581     4.17   0.000     .1597532
.446042
       write |   .1104172   .0713398     1.55   0.123    -.0302795
.2511139
       _cons |   1.946078   3.087346     0.63   0.529    -4.142797
8.034954
------------------------------------------------------------------------
-------
```

We use the **testparm** command to test that the coefficients for **science**, **socst** and **math** are equal.

```
testparm science socst math, equal
 ( 1) - science + socst = 0.0
 ( 2) - science + math = 0.0

       F(  2,   195) =    0.05
            Prob > F =    0.9554
```

We can also use the test command to test that the coefficients for **science**, **socst** and **math** are equal.

```
test science=socst
 ( 1)  science - socst = 0.0

       F(  1,   195) =    0.00
            Prob > F =    0.9964
test socst=math, accum
 ( 1)  science - socst = 0.0
 ( 2)  socst - math = 0.0

       F(  2,   195) =    0.05
            Prob > F =    0.9554
```

We now constrain these three coefficients to be equal.

```
constraint define 1 science = socst
constraint define 2 socst = math
```

And we use **cnsreg** to estimate the model with these constraints in place.

```
cnsreg read science socst math write, c(1 2)
```

```
Constrained linear regression                          Number of obs =
200
                                                       F(  2,   197) =
140.80
                                                       Prob > F      =
0.0000
                                                       Root MSE      =
6.6113
 ( 1)  science - socst = 0.0
 ( 2)  socst - math = 0.0
------------------------------------------------------------------------
-------
        read |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
     science |   .2828596   .0268291    10.54   0.000     .2299505
.3357687
       socst |   .2828596   .0268291    10.54   0.000     .2299505
.3357687
        math |   .2828596   .0268291    10.54   0.000     .2299505
.3357687
       write |   .1106022   .0708452     1.56   0.120      -.02911
.2503145
       _cons |   2.012299   3.061703     0.66   0.512    -4.025622
8.05022
------------------------------------------------------------------------
-------
```

5. Using the elemapi2 data file (use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2) consider the following 2 regression equations.

```
api00 = meals ell emer
api99 = meals ell emer
```

Estimate the coefficients for these predictors in predicting **api00** and **api99** taking into account the non-independence of the schools. Test the overall contribution of each of the predictors in jointly predicting api scores in these two years. Test whether the contribution of **emer** is the same for **api00** and **api99**.

Answer 5.
First, let's use the **elemapi2** data file.

```
use http://www.ats.ucla.edu/stat/stata/webbooks/reg/elemapi2, clear
```

Next, let's analysze these equations separately.

```
regress api00 meals ell emer
```

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
673.00
       Model |  6749782.75     3  2249927.58           Prob > F      =
0.0000
    Residual |  1323889.25   396  3343.15467           R-squared     =
0.8360
-------------+------------------------------           Adj R-squared =
0.8348
       Total |  8073672.00   399  20234.7669           Root MSE      =
57.82

------------------------------------------------------------------------
-------
       api00 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
       meals |  -3.159189   .1497371   -21.10   0.000    -3.453568   -
2.864809
         ell |  -.9098732   .1846442    -4.93   0.000    -1.272878   -
.5468678
        emer |  -1.573496    .293112    -5.37   0.000    -2.149746   -
.9972456
       _cons |   886.7033    6.25976   141.65   0.000     874.3967
899.0098
------------------------------------------------------------------------
-------
```

regress api99 meals ell emer

```
      Source |       SS       df       MS              Number of obs =
400
-------------+------------------------------           F(  3,   396) =
716.31
       Model |  7293890.24     3  2431296.75           Prob > F      =
0.0000
    Residual |  1344092.70   396  3394.17349           R-squared     =
0.8444
-------------+------------------------------           Adj R-squared =
0.8432
       Total |  8637982.94   399    21649.08           Root MSE      =
58.26

------------------------------------------------------------------------
-------
       api99 |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
       meals |  -3.412388   .1508754   -22.62   0.000    -3.709004   -
3.115771
         ell |   -.793822   .1860477    -4.27   0.000    -1.159587   -
.4280573
        emer |  -1.516305   .2953401    -5.13   0.000    -2.096936   -
.9356748
       _cons |    860.191   6.307343   136.38   0.000     847.7909
872.591
```

```
------------------------------------------------------------------------
-------
```

Now, let's analyze them using **sureg** that takes into account the non-independence of these equations.

```
sureg (api00 api99 = meals ell emer)
Seemingly unrelated regression
----------------------------------------------------------------------
Equation          Obs  Parms        RMSE    "R-sq"       chi2        P
----------------------------------------------------------------------
api00             400      3    57.53019    0.8360    2039.38   0.0000
api99             400      3    57.96751    0.8444   2170.651   0.0000
----------------------------------------------------------------------

------------------------------------------------------------------------
-------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf.
Interval]
-------------+----------------------------------------------------------
-------
api00        |
       meals |  -3.159189   .1489866   -21.20   0.000    -3.451197    -
2.86718
         ell |  -.9098732   .1837186    -4.95   0.000    -1.269955   -
.5497913
        emer |  -1.573496   .2916428    -5.40   0.000    -2.145105   -
1.001886
       _cons |   886.7033   6.228382   142.36   0.000     874.4959
898.9107
-------------+----------------------------------------------------------
-------
api99        |
       meals |  -3.412388   .1501191   -22.73   0.000    -3.706616    -
3.11816
         ell |   -.793822   .1851151    -4.29   0.000    -1.156641    -
.431003
        emer |  -1.516305   .2938597    -5.16   0.000     -2.09226   -
.9403509
       _cons |    860.191   6.275727   137.07   0.000     847.8908
872.4912
------------------------------------------------------------------------
-------
```

We can test the contribution of **meals ell** and **emer** as shown below.

```
test meals
 ( 1)  [api00]meals = 0.0
 ( 2)  [api99]meals = 0.0

           chi2(  2) =  518.30
         Prob > chi2 =    0.0000

test ell
 ( 1)  [api00]ell = 0.0
```

```
 ( 2)  [api99]ell = 0.0

           chi2(  2) =   24.80
         Prob > chi2 =    0.0000
test emer
 ( 1)  [api00]emer = 0.0
 ( 2)  [api99]emer = 0.0

           chi2(  2) =   29.48
         Prob > chi2 =    0.0000
```

We can test whether the coefficients for **emer** were the same in predicting **api00** and **api99** as shown below.

```
test [api00]emer = [api99]emer

 ( 1)  [api00]emer - [api99]emer = 0.0

           chi2(  1) =    0.21
         Prob > chi2 =    0.6456
```

We can also test the contribution of **meals ell** and **emer** using more traditional multivariate tests using the **mvreg** and **mvtest** commands as shown below.

```
mvreg api00 api99 = meals ell emer

Equation          Obs  Parms        RMSE    "R-sq"          F        P
----------------------------------------------------------------------
api00             400      4    57.82002    0.8360   672.9954   0.0000
api99             400      4    58.25954    0.8444   716.3148   0.0000

-----------------------------------------------------------------------
-------
             |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------
-------
api00        |
       meals |  -3.159189   .1497371   -21.10   0.000    -3.453568   -
2.864809
         ell |  -.9098732   .1846442    -4.93   0.000    -1.272878   -
.5468678
        emer |  -1.573496    .293112    -5.37   0.000    -2.149746   -
.9972456
       _cons |   886.7033    6.25976   141.65   0.000     874.3967
899.0098
-------------+---------------------------------------------------------
-------
api99        |
       meals |  -3.412388   .1508754   -22.62   0.000    -3.709004   -
3.115771
         ell |   -.793822   .1860477    -4.27   0.000    -1.159587   -
.4280573
        emer |  -1.516305   .2953401    -5.13   0.000    -2.096936   -
.9356748
```

```
        _cons |    860.191   6.307343   136.38   0.000     847.7909
872.591
-----------------------------------------------------------------------
-------
```

Below we show the multivariate tests for **meals ell** and for **emer**.

mvtest meals

```
                     MULTIVARIATE TESTS OF SIGNIFICANCE

Multivariate Test Criteria and Exact F Statistics for
the Hypothesis of no Overall "meals" Effect(s)

                           S=1    M=0    N=196.5

Test                          Value          F       Num DF      Den DF   Pr >
F
Wilks' Lambda              0.43558762   255.9105          2   395.0000
0.0000
Pillai's Trace             0.56441238   255.9105          2   395.0000
0.0000
Hotelling-Lawley Trace     1.29574936   255.9105          2   395.0000
0.0000
```

mvtest ell

```
                     MULTIVARIATE TESTS OF SIGNIFICANCE

Multivariate Test Criteria and Exact F Statistics for
the Hypothesis of no Overall "ell" Effect(s)

                           S=1    M=0    N=196.5

Test                          Value          F       Num DF      Den DF   Pr >
F
Wilks' Lambda              0.94161436    12.2462          2   395.0000
0.0000
Pillai's Trace             0.05838564    12.2462          2   395.0000
0.0000
Hotelling-Lawley Trace     0.06200590    12.2462          2   395.0000
0.0000
```

mvtest emer

```
                     MULTIVARIATE TESTS OF SIGNIFICANCE

Multivariate Test Criteria and Exact F Statistics for
the Hypothesis of no Overall "emer" Effect(s)

                           S=1    M=0    N=196.5

Test                          Value          F       Num DF      Den DF   Pr >
F
Wilks' Lambda              0.93136794    14.5537          2   395.0000
0.0000
```

Pillai's Trace	0.06863206	14.5537	2	395.0000	0.0000
Hotelling-Lawley Trace	0.07368952	14.5537	2	395.0000	0.0000

Chapter 5 - Additional coding systems for categorical variables in regression analysis

Chapter Outline

Please note: This page makes use of the program **xi3** which is no longer being maintained and has been from our archives. References to **xi3** will be left on this page because they illustrate specific principles of coding categorical variables.

5.0 Introduction

Categorical variables require special attention in regression analysis because, unlike dichotomous or continuous variables, they cannot by entered into the regression equation just as they are. For example, if you have a variable called **race** that is coded 1 = Hispanic, 2 = Asian 3 = Black 4 = White, then entering **race** in your regression will look at the linear effect of race, which is probably not what you intended. Instead, categorical variables like this need to be recoded into a series of variables which can then be entered into the regression model. There are a variety of coding systems that can be used when coding categorical variables. Ideally, you would choose a coding system that reflects the comparisons that you want to make. In Chapter 3 of the Regression with Stata Web Book we covered the use of categorical variables in regression analysis focusing on the use of dummy variables, but that is not the only coding scheme that you can use. For example, you may want to compare each level to the next higher level, in which case you would want to use "forward difference" coding, or you might want to compare each level to the mean of the subsequent levels of the variable, in which case you would want to use "Helmert" coding. By deliberately choosing a coding system, you can obtain comparisons that are most meaningful for testing your hypotheses. Regardless of the coding system you choose, the test of the overall effect of the categorical variable (i.e., the overall effect of **race**) will remain the same. Below is a table listing various types of contrasts and the comparison that they make.

Name of contrast	**Comparison made**
Simple Coding	Compares each level of a variable to the reference level
Forward Difference Coding	Adjacent levels of a variable (each level minus the next level)

Backward Difference Coding	Adjacent levels of a variable (each level minus the prior level)
Helmert Coding	Compare levels of a variable with the mean of the subsequent levels of the variable
Reverse Helmert Coding	Compares levels of a variable with the mean of the previous levels of the variable
Deviation Coding	Compares deviations from the grand mean
Orthogonal Polynomial Coding	Orthogonal polynomial contrasts
User-Defined Coding	User-defined contrast

There are a couple of notes to be made about the coding systems listed above. The first is that they represent planned comparisons and not post hoc comparisons. In other words, they are comparisons that you plan to do before you begin analyzing your data, not comparisons that you think of once you have seen the results of preliminary analyses. Also, some forms of coding make more sense with ordinal categorical variables than with nominal categorical variables. Below we will show examples using **race** as a categorical variable, which is a nominal variable. Because simple effect coding compares the mean of the dependent variable for each level of the categorical variable to the mean of the dependent variable at for the reference level, it makes sense with a nominal variable. However, it may not make as much sense to use a coding scheme that tests the linear effect of **race**. As we describe each type of coding system, we note those coding systems with which it does not make as much sense to use a nominal variable. Also, you may notice that we follow several rules when creating the contrast coding schemes. For more information about these rules, please see the section on User-Defined Coding.

This page will illustrate two ways that you can conduct analyses using these coding schemes: 1) using the **xi3** command (an extended version of the **xi** command) and 2) manually coding the variables and entering them using the **regress** command. When using **regress** to do contrasts, you first need to create k-1 new variables (where k is the number of levels of the categorical variable) and use these new variables as predictors in your regression model.

The Example Data File

The examples in this page will use dataset called hsb2.dta that you can download from within Stata like this.

```
use http://www.ats.ucla.edu/stat/stata/notes/hsb2
```

Within this data file, we will focus on the categorical variable **race**, which has four levels (1 = Hispanic, 2 = Asian, 3 = African American and 4 = white) and we will use **write** as our dependent variable. Although our example uses a variable with four levels, these coding systems work with variables that have more or fewer categories. No matter which coding system you select, you will always have one fewer recoded variables than levels of the original variable. In our example, our categorical variable has four levels so we will have three new variables (a variable corresponding to the final level of the categorical variables would be redundant and therefore unnecessary).

Before considering any analyses, let's look at the mean of the dependent variable, **write**, for each level of **race**. This will help in interpreting the output from later analyses.

```
tabulate race, summarize(write)

            |      Summary of writing score
       race |        Mean   Std. Dev.       Freq.
------------+------------------------------------
   hispanic |   46.458333   8.2724223          24
      asian |          58   7.8993671          11
  african-a |        48.2   9.3222992          20
      white |   54.055172   9.1725582         145
------------+------------------------------------
      Total |      52.775    9.478586         200
```

5.1 Simple Coding

The results of simple coding are very similar to dummy coding in that each level is compared to the reference level. In the example below, level 1 is the reference level and the first comparison compares level 2 to level 1, the second comparison compares level 3 to level 1, and the third comparison compares level 4 to level 1.

Method 1: Using xi3

When using **xi3**, we can refer to **g.race** to indicate that we wish to code race using simple coding comparing each group to a reference group, as shown in the example below.

```
xi3: regress write g.race
s.race            _Irace_1-4          (naturally coded; _Irace_1 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251
```

```
------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
    _Irace_2 |   11.54167   3.286129     3.51   0.001     5.060956
18.02238
    _Irace_3 |   1.741667   2.732488     0.64   0.525    -3.647186
7.130519
    _Irace_4 |   7.596839    1.98887     3.82   0.000     3.674507
11.51917
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

The coefficient for **_Irace_2** compares the mean of the dependent variable, **write**, for levels 2 and 1 yielding 58-46.458 = 11.54 and is statistically significant (p<.000). The coefficient for **_Irace_3** compares the mean of the dependent variable, **write**, for levels 3 and 1, yielding 48.2 - 46.46 = 1.74, and this is not statistically significant. Finally, the coefficient for **_Irace_4** compares the mean of the dependent variable, **write**, for levels 4 and 1, yielding 7.59, and that is statistically significant.

Method 2: Manual Coding

If we wished, we could manually code **race** instead of allowing **xi3** to do the coding for us. Below we see the coding that replicates the results we saw in the example above. In the coding below, level 1 is the reference level and **x1** compares level 2 to level 1, **x2** compares level 3 to level 1, and **x3** compares level 4 to level 1. For **x1** the coding is 3/4 for level 2, and -1/4 for all other levels. Likewise, for **x2** the coding is 3/4 for level 2, and -1/4 for all other levels, and for **x3** the coding is 3/4 for level 3, and -1/4 for all other levels. It is not intuitive that this regression coding scheme yields these comparisons; however, if you desire simple comparisons, you can follow this general rule to obtain these comparisons.

SIMPLE regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
1 (Hispanic)	-1/4	-1/4	-1/4
2 (Asian)	3/4	-1/4	-1/4
3 (African American)	-1/4	3/4	-1/4
4 (white)	-1/4	-1/4	3/4

Below we show the more general rule for creating this kind of coding scheme using regression coding, where k is the number of levels of the categorical variable (in this instance, k = 4).

SIMPLE regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
1 (Hispanic)	-1 / k	-1 / k	-1 / k
2 (Asian)	(k-1) / k	-1 / k	-1 / k
3 (African American)	-1 / k	(k-1) / k	-1 / k
4 (white)	-1 / k	-1 / k	(k-1) / k

Below we illustrate how to create **x1**, **x2** and **x3** and enter these new variables into the regression model using the **regression** command.

```
generate x1 = -1/4
replace x1 = 3/4 if race==2

generate x2 = -1/4
replace x2 = 3/4 if race==3

generate x3 = -1/4
replace x3 = 3/4 if race==4

regress write x1 x2 x3
```

As you can see, the results below match those when we used the **xi3** command above.

```
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
```

```
-------------+----------------------------------------------------------------
-
          x1 |   11.54167   3.286129     3.51   0.001     5.060956
18.02238
          x2 |   1.741667   2.732488     0.64   0.525    -3.647186
7.130519
          x3 |   7.596839    1.98887     3.82   0.000     3.674507
11.51917
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

5.2 Forward Difference Coding

In this coding system, the mean of the dependent variable for one level of the categorical variable is compared to the mean of the dependent variable for the next (adjacent) level. In our example below, the first comparison compares the mean of **write** for level 1 with the mean of **write** for level 2 of **race** (Hispanics minus Asians). The second comparison compares the mean of **write** for level 2 minus level 3, and the third comparison compares the mean of **write** for level 3 minus level 4. This type of coding may be useful with either a nominal or an ordinal variable.

Method 1: Using xi3

We can indicate that we want forward adjacent difference coding for race by specifying **a.race** as shown below.

```
xi3 : regress write a.race

f.race            _Irace_1-4          (naturally coded; _Irace_4 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
    _Irace_1 |  -11.54167   3.286129    -3.51   0.001    -18.02238   -
5.060956
```

```
    _Irace_2 |        9.8   3.387834     2.89   0.004     3.118714
16.48129
    _Irace_3 |  -5.855172    2.15276    -2.72   0.007    -10.10072   -
1.609626
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
----------------------------------------------------------------------------
-
```

With this coding system, adjacent levels of the categorical variable are compared. Hence, the mean of the dependent variable at level 1 is compared to the mean of the dependent variable at level 2: 46.4583 - 58 = -11.542, which is statistically significant. For the comparison between levels 2 and 3, the calculation of the contrast coefficient would be 58 - 48.2 = 9.8, which is also statistically significant. Finally, comparing levels 3 and 4, 48.2 - 54.0552 = -5.855, a statistically significant difference. One would conclude from this that each adjacent level of **race** is statistically significantly different.

Method 2: Manual Coding

For the first comparison, where the first and second levels are compared, **x1** is coded 3/4 for level 1 and the other levels are coded -1/4. For the second comparison where level 2 is compared with level 3, **x2** is coded 1/2 1/2 -1/2 -1/2, and for the third comparison where level 3 is compared with level 4, **x3** is coded 1/4 1/4 1/4 -3/4.

FORWARD DIFFERENCE regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
	Level 1 v. Level 2	Level 2 v. Level 3	Level 3 v. Level 4
1 (Hispanic)	3/4	1/2	1/4
2 (Asian)	-1/4	1/2	1/4
3 (African American)	-1/4	-1/2	1/4
4 (white)	-1/4	-1/2	-3/4

The general rule for this regression coding scheme is shown below, where k is the number of levels of the categorical variable (in this case k = 4).

FORWARD DIFFERENCE regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)

	Level 1 v. Level 2	Level 2 v. Level 3	Level 3 v. Level 4
1 (Hispanic)	(k-1)/k	(k-2)/k	(k-3)/k
2 (Asian)	-1/k	(k-2)/k	(k-3)/k
3 (African American)	-1/k	-2/k	(k-3)/k
4 (white)	-1/k	-2/k	-3/k

```
generate x1 = 3/4 if race==1
replace x1 = -1/4 if inlist(race,2,3,4)
generate x2 = 1/2 if inlist(race,1,2)
replace x2 = -1/2 if inlist(race,3,4)
generate x3 = 1/4 if inlist(race,1,2,3)
replace x3 = -3/4 if race==4
regress write x1 x2 x3
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
          x1 |  -11.54167   3.286129    -3.51   0.001    -18.02238   -
5.060956
          x2 |        9.8   3.387834     2.89   0.004     3.118714
16.48129
          x3 |  -5.855172    2.15276    -2.72   0.007    -10.10072   -
1.609626
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
-----------------------------------------------------------------------------
-
```

You can see the regression coefficient for **x1** is the mean of **write** for level 1 (Hispanic) minus the mean of **write** for level 2 (Asian). Likewise, the regression coefficient for **x2** is the mean of **write** for level 2 (Asian) minus the mean of **write** for level 3 (African American), and the regression coefficient for **x3** is the mean of **write** for level 3 (African American) minus the mean of **write** for level 4 (white).

5.3 Backward Difference Coding

In this coding system, the mean of the dependent variable for one level of the categorical variable is compared to the mean of the dependent variable for the prior adjacent level. In our example below, the first comparison compares the mean of **write** for level 2 with the mean of **write** for level 1 of **race** (Hispanics minus Asians). The second comparison compares the mean of **write** for level 3 minus level 2, and the third comparison compares the mean of **write** for level 4 minus level 3. This type of coding may be useful with either a nominal or an ordinal variable.

Method 1: Using xi3

We can indicate that we want backward difference coding for race by specifying **b.race** as shown below.

```
xi3 : regress write b.race
b.race            _Irace_1-4          (naturally coded; _Irace_1 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
    _Irace_2 |   11.54167   3.286129     3.51   0.001     5.060956
18.02238
    _Irace_3 |       -9.8   3.387834    -2.89   0.004    -16.48129   -
3.118714
    _Irace_4 |   5.855172    2.15276     2.72   0.007     1.609626
10.10072
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
-----------------------------------------------------------------------------
-
```

With this coding system, adjacent levels of the categorical variable are compared, with each level compared to the prior level. Hence, the mean of the dependent variable at level 2 is compared to the mean of the dependent variable at level 1: 58-46.4583 = 11.542, which is statistically significant. For the comparison between levels 3 and 2, we calculate 48.2 - 58 = -9.8, which is also statistically significant. Finally, comparing levels 4 and 3, 54.0552 - 48.2 = 5.855, a statistically significant difference. One would conclude from this that each adjacent level of **race** is statistically significantly different.

Method 2: Manual Coding

For the first comparison, where the first and second levels are compared, **x1** is coded 3/4 for level 1 while the other levels are coded -1/4. For the second comparison where level 2 is compared with level 3, **x2** is coded 1/2 1/2 -1/2 -1/2, and for the third comparison where level 3 is compared with level 4, **x3** is coded 1/4 1/4 1/4 -3/4.

BACKWARD DIFFERENCE regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
	Level 2 v. Level 1	Level 3 v. Level 2	Level 4 v. Level 3
1 (Hispanic)	- 3/4	-1/2	-1/4
2 (Asian)	1/4	-1/2	-1/4
3 (African American)	1/4	1/2	-1/4
4 (white)	1/4	1/2	3/4

The general rule for this regression coding scheme is shown below, where k is the number of levels of the categorical variable (in this case, k = 4).

BACKWARD DIFFERENCE regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
	Level 1 v. Level 2	Level 2 v. Level 3	Level 3 v. Level 4
1 (Hispanic)	-(k-1)/k	-(k-2)/k	-(k-3)/k
2 (Asian)	1/k	-(k-2)/k	-(k-3)/k
3 (African American)	1/k	2/k	-(k-3)/k
4 (white)	1/k	2/k	3/k

```
generate x1 = -3/4 if race==1
```

```
replace  x1 =  1/4 if inlist(race,2,3,4)

generate x2 = -1/2 if inlist(race,1,2)
replace  x2 =  1/2 if inlist(race,3,4)

generate x3 = -1/4 if inlist(race,1,2,3)
replace  x3 =  3/4 if race==4

regress write x1 x2 x3
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
          x1 |   11.54167   3.286129     3.51   0.001     5.060956
18.02238
          x2 |       -9.8   3.387834    -2.89   0.004    -16.48129   -
3.118714
          x3 |   5.855172    2.15276     2.72   0.007     1.609626
10.10072
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
-----------------------------------------------------------------------------
-
```

In the above example, the regression coefficient for **x1** is the mean of **write** for level 2 minus the mean of **write** for level 1 (58- 46.4583 = 11.542). Likewise, the regression coefficient for **x2** is the mean of **write** for level 3 minus the mean of **write** for level 2, and the regression coefficient for **x3** is the mean of **write** for level 4 minus the mean of **write** for level 3.

5.4 Helmert Coding

Helmert coding compares each level of a categorical variable to the mean of the subsequent levels. Hence, the first contrast compares the mean of the dependent variable for level 1 of **race** with the mean of all of the subsequent levels of **race** (levels 2, 3, and 4), the second contrast compares the mean of the dependent variable for level 2 of **race** with the mean of all of the subsequent levels of **race** (levels 3 and 4), and the third contrast compares the mean of the dependent variable for level 3 of **race** with the mean of all of the subsequent levels of **race** (level 4). While this type of coding system does not make much sense with a nominal variable like

race, it is useful in situations where the levels of the categorical variable are ordered say, from lowest to highest, or smallest to largest, etc.

Method 1: Using xi3

We can specify Helmert coding for **race** using **h.race** as shown below.

```
xi3 : regress write h.race
h.race            _Irace_1-4         (naturally coded; _Irace_4 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
    _Irace_1 |  -6.960057   2.175211    -3.20   0.002    -11.24988   -
2.670234
    _Irace_2 |   6.872414   2.926325     2.35   0.020     1.101287
12.64354
    _Irace_3 |  -5.855172    2.15276    -2.72   0.007    -10.10072   -
1.609626
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

The regression coefficient for the comparison between level 1 and the remaining levels is calculated by taking the mean of the dependent variable for level 1 and subtracting the mean of the dependent variable for levels 2, 3 and 4: 46.4583 - [(58 + 48.2 + 54.0552) / 3] = -6.960, which is statistically significant. This means that the mean of **write** for level 1 of **race** is statistically significantly different from the mean of **write** for levels 2 through 4. As noted above, this comparison probably is not meaningful because the variable **race** is nominal. This type of comparison would be more meaningful if the categorical variable was ordinal.

To calculate the contrast coefficient for the comparison between level 2 and the later levels, you subtract the mean of the dependent variable for levels 3 and 4 from the mean of the dependent variable for level 2: 58 - [(48.2 + 54.0552) / 2] = 6.872, which is statistically significant. The regression coefficient for the comparison between level 3 and level 4 is the difference between the mean of the dependent variable for the two levels: 48.2 - 54.0552 = -5.855, which is also statistically significant.

Method 2: Manual Coding

Below we see an example of Helmert regression coding. For the first comparison (comparing level 1 with levels 2, 3 and 4) the codes are 3/4 and -1/4 -1/4 -1/4. The second comparison compares level 2 with levels 3 and 4 and is coded 0 2/3 -1/3 -1/3. The third comparison compares level 3 to level 4 and is coded 0 0 1/2 -1/2.

HELMERT regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
	Level 1 v. Later	Level 2 v. Later	Level 3 v. Later
1 (Hispanic)	3/4	0	0
2 (Asian)	-1/4	2/3	0
3 (African American)	-1/4	-1/3	1/2
4 (white)	-1/4	-1/3	-1/2

Below we illustrate how to create **x1**, **x2** and **x3** and enter these new variables into the regression model using the **regression** command.

```
generate x1 = -3/4 if race==1
replace  x1 =  1/4 if inlist(race,2,3,4)

generate x2 =    0 if race==1
replace  x2 =  2/3 if race==2
replace  x2 = -1/3 if inlist(race,3,4)

generate x3 =    0 if inlist(race,1,2)
replace  x3 =  1/2 if race==3
replace  x3 = -1/2 if race==4

regress write x1 x2 x3
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
```

```
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
          x1 |   6.960057   2.175211     3.20   0.002     2.670234
11.24988
          x2 |   6.872414   2.926325     2.35   0.020     1.101287
12.64354
          x3 |  -5.855172    2.15276    -2.72   0.007    -10.10072   -
1.609626
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

As you see above, regression coefficient for **x1** is the mean of **write** for level 1 (Hispanic) versus all subsequent levels (levels 2, 3 and 4). Likewise, the regression coefficient for **x2** is the mean of **write** for level 2 minus the mean of **write** for levels 3 and 4. Finally, the regression coefficient for **x3** is the mean of **write** for level 3 minus the mean of **write** for level 4.

5.5 Reverse Helmert Coding

Reverse Helmert coding (also know as difference coding) is just the opposite of Helmert coding: instead of comparing each level of categorical variable to the mean of the subsequent level(s), each is compared to the mean of the previous level(s). In our example, the first contrast codes the comparison of the mean of the dependent variable for level 2 of **race** to the mean of the dependent variable for level 1 of **race**. The second comparison compares the mean of the dependent variable level 3 of **race** with both levels 1 and 2 of **race**, and the third comparison compares the mean of the dependent variable for level 4 of **race** with levels 1, 2 and 3. Clearly, this coding system does not make much sense with our example of **race** because it is a nominal variable. However, this system is useful when the levels of the categorical variable are ordered in a meaningful way. For example, if we had a categorical variable in which work-related stress was coded as low, medium or high, then comparing the means of the previous levels of the variable would make more sense.

Method 1: Using xi3

We can specify Helmert coding for **race** using **r.race** as shown below.

```
xi3 : regress write r.race

r.race            _Irace_1-4          (naturally coded; _Irace_1 omitted)
```

```
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
    _Irace_2 |   11.54167   3.286129     3.51   0.001     5.060956
18.02238
    _Irace_3 |  -4.029167   2.602363    -1.55   0.123    -9.161394
1.103061
    _Irace_4 |   3.169061   1.487987     2.13   0.034     .2345401
6.103582
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

The regression coefficient for the first comparison shown in this output was calculated by subtracting the mean of the dependent variable for level 2 of the categorical variable from the mean of the dependent variable for level 1: 58 - 46.4583 = 11.542. This result is statistically significant. The regression coefficient for the second comparison (between level 3 and the previous levels) was calculated by subtracting the mean of the dependent variable for levels 1 and 2 from that of level 3: 48.2 - [(46.4583 + 58) / 2] = -4.029. This result is not statistically significant, meaning that there is not a reliable difference between the mean of **write** for level 3 of **race** compared to the mean of **write** for levels 1 and 2 (Hispanics and Asians). As noted above, this type of coding system does not make much sense for a nominal variable such as **race**. For the comparison of level 4 and the previous levels, you take the mean of the dependent variable for the those levels and subtract it from the mean of the dependent variable for level 4: 54.0552 - [(46.4583 + 58 + 48.2) / 3] = 3.169. This result is statistically significant.

Method 2: Manual Coding

The regression coding for reverse Helmert coding is shown below. For the first comparison, where the first and second level are compared, **x1** is coded -1/2 and 1/2 and 0 otherwise. For the second comparison, the values of **x2** are coded -1/3 -1/3 2/3 and 0. Finally, for the third comparison, the values of **x3** are coded -1/4 -1/4 -/14 and 3/4.

REVERSE HELMERT regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
1 (Hispanic)	-1/2	-1/3	-1/4
2 (Asian)	1/2	-1/3	-1/4
3 (African American)	0	2/3	-1/4
4 (white)	0	0	3/4

Below we illustrate how to create **x1**, **x2** and **x3** and enter these new variables into the regression model using the **regress** command.

```
generate x1 = -1/2 if race==1
replace  x1 =  1/2 if race==2
replace  x1 =    0 if inlist(race,3,4)

generate x2 = -1/3 if inlist(race,1,2)
replace  x2 =  2/3 if race==3
replace  x2 =    0 if race==4

generate x3 = -1/4 if inlist(race,1,2,3)
replace  x3 =  3/4 if race==4

regress write x1 x2 x3
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
          x1 |   11.54167   3.286129     3.51   0.001     5.060956
18.02238
          x2 |  -4.029167   2.602363    -1.55   0.123    -9.161394
1.103061
          x3 |   3.169061   1.487987     2.13   0.034     .2345401
6.103582
```

```
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
-----------------------------------------------------------------------------
-
```

In the above example, the regression coefficient for **x1** is the mean of **write** for level 1 (Hispanic) minus the mean of **write** for level 2 (Asian). Likewise, the regression coefficient for **x2** is the mean of **write** for levels 1 and 2 combined minus the mean of **write** for level 3. Finally, the regression coefficient for **x3** is the mean of **write** for levels 1, 2 and 3 combined minus the mean of **write** for level 4.

5.6 Deviation Coding

This coding system compares the mean of the dependent variable for a given level to the mean of the dependent variable for the all levels of the variable. In our example below, the first comparison compares level 2 (Asians) to all levels of **race**, the second compares level 3 (African Americans) to all levels of **race**, and the third comparison compares level 4 (White) to all levels of race.

Method 1: Using xi3

We indicate we would like **race** to be coded using deviation effect coding using **e.race** as shown below.

```
. xi3 : regress write e.race
d.race            _Irace_1-4          (naturally coded; _Irace_1 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
    _Irace_2 |   6.321624   2.160314     2.93   0.004     2.061179
10.58207
    _Irace_3 |  -3.478376   1.732305    -2.01   0.046    -6.894726    -
.062027
    _Irace_4 |   2.376796   1.115991     2.13   0.034     .1759051
4.577687
```

```
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
-----------------------------------------------------------------------------
-
```

The regression coefficient for **_Irace_2** is the mean for level 2 minus the grand mean. However, this grand mean is not the overall mean of the dependent variable that you would get from the **summarize** command. Rather, it is the mean of means of the dependent variable at each level of the categorical variable: (46.4583 + 58 + 48.2 + 54.0552) / 4 = 51.678375. This regression coefficient is then 58 - 51.678375 = 6.32. Likewise, the coefficient for **_Irace_3** is the mean for level 3 of race minus the overall mean, i.e., 48.2 - 51.678 = -3.47, and **_Irace_4** is the mean for level 4 of race minus the overall mean, 54.055 - 51.678 = 2.37.

Method 2: Manual Coding

As you see in the example below, the regression coding is accomplished by assigning 1 to level 2 for the first comparison (because level 2 is the level to be compared to all), level 1 to level 3 for the second comparison (because level 3 is to be compared to all), and 1 to level 4 for the third comparison (because level 4 is to be compared to all). Note that a -1 is assigned to level 1 for all three comparisons (because it is the level that is never compared to the other levels) and all other values are assigned a 0. This regression coding scheme yields the comparisons described above.

DEVIATION regression coding

Level of race	New variable 1 (x1)	New variable 2 (x2)	New variable 3 (x3)
	Level 2 v. Mean	Level 3 v. Mean	Level 4 v. Mean
1 (Hispanic)	-1	-1	-1
2 (Asian)	1	0	0
3 (African American)	0	1	0
4 (white)	0	0	1

Below we illustrate how to create **x1**, **x2** and **x3** and enter these new variables into the regression model using the **regress** command.

```
generate x1 = -1 if race==1
replace  x1 =  1 if race==2
replace  x1 =  0 if inlist(race,3,4)

generate x2 = -1 if race==1
replace  x2 =  1 if race==3
```

```
replace  x2 =  0 if inlist(race,2,4)

generate x3 = -1 if race==1
replace  x3 =  1 if race==4
replace  x3 =  0 if inlist(race,2,3)

regress write x1 x2 x3
     Source |       SS       df       MS              Number of obs =     200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+---------------------------------------------------------------
-
          x1 |   6.321624   2.160314     2.93   0.004     2.061179
10.58207
          x2 |  -3.478376   1.732305    -2.01   0.046    -6.894726    -
.062027
          x3 |   2.376796   1.115991     2.13   0.034     .1759051
4.577687
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
-----------------------------------------------------------------------------
-
```

The regression coefficients for this analysis match those in the example above and have the same interpretation.

5.7 Orthogonal Polynomial Coding

Orthogonal polynomial coding is a form of trend analysis in that it is looking for the linear, quadratic and cubic trends in the categorical variable. This type of coding system should be used only with an ordinal variable in which the levels are equally spaced. Examples of such a variable might be income or education. The table below shows the contrast coefficients for the linear, quadratic and cubic trends for the four levels. These could be obtained from most statistics books on linear models.

POLYNOMIAL

Level of race	Linear (x1)	Quadratic (x2)	Cubic (x3)

1 (Hispanic)	-.671	.5	-.224
2 (Asian)	-.224	-.5	.671
3 (African American)	.224	-.5	-.671
4 (white)	.671	.5	.224

Method 1: Using xi3

We indicate we would like race to be coded using orthogonal polynomials by using o.race as shown below.

```
. xi3 : regress write o.race
o.race            _Irace_1-4          (naturally coded; _Irace_4 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0000
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
    _Irace_1 |   2.080058   .6381718     3.26   0.001     .8214929
3.338622
    _Irace_2 |  -.2159021   .6381718    -0.34   0.735    -1.474467
1.042663
    _Irace_3 |   2.279811   .6381718     3.57   0.000     1.021246
3.538375
       _cons |     52.775   .6381718    82.70   0.000     51.51644
54.03356
------------------------------------------------------------------------------
-
```

The three coded variables, **_Irace_1**, **_Irace_2** and **_Irace_3**, represent the linear, quadratic and cubic trends respectively. Of course, the term 'trend' doesn't make sense if the variable is nominal, like **race**. But if we pretend that **race** is ordinal than there would be a significant linear and cubic trend. It is also easy to test for nonlinear trend.

```
. test _Irace_2 _Irace_3

 ( 1)  _Irace_2 = 0.0
 ( 2)  _Irace_3 = 0.0

       F(  2,   196) =    6.44
            Prob > F =    0.0020
```

The test for nonlinear trend is statistically significant. This example worked okay to show how to use **xi3** but we need an ordered example that can be interpreted.

Example 2

We will create our own categorical variable, **readcat**, from the continuous variable **read**.

```
. gen readcat = read
recode readcat 1/43=1 44/49=2 50/59=3 60/100=4

tab readcat

    readcat |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         39       19.50       19.50
          2 |         44       22.00       41.50
          3 |         61       30.50       72.00
          4 |         56       28.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

Now we can run the regression with **xi3**.

```
. xi3: regress write o.readcat
o.readcat         _Ireadcat_1-4       (naturally coded; _Ireadcat_4 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
29.64
       Model |  5579.22989     3   1859.7433           Prob > F      =
0.0000
    Residual |  12299.6451   196  62.7532914           R-squared     =
0.3121
-------------+------------------------------           Adj R-squared =
0.3015
       Total |   17878.875   199   89.843593           Root MSE      =
7.9217

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
```

```
 _Ireadcat_1 |    5.27249   .5601486     9.41   0.000     4.167798
6.377182
 _Ireadcat_2 |   .3097532   .5601486     0.55   0.581     -.794939
1.414445
 _Ireadcat_3 |  -.0324612   .5601486    -0.06   0.954    -1.137153
1.072231
       _cons |     52.775   .5601486    94.22   0.000     51.67031
53.87969
-----------------------------------------------------------------------------
-
```

We see from the significant **_Ireadcat_1** that the linear trend is significant while neither quadratic nor cubic trends (_Ireadcat_2 & _Ireadcat_3) are significant. The test for nonlinear trend is also nonsignificant.

```
. test _Ireadcat_2 _Ireadcat_3

 ( 1)  _Ireadcat_2 = 0.0
 ( 2)  _Ireadcat_3 = 0.0

       F(  2,   196) =    0.15
            Prob > F =    0.8569
```

Method 2: Manual Coding

```
For the moment we are skipping manual coding.
```

5.8 User Defined Coding

You can use the **xi3** command to create your own regression coding system. For our example, we will make the following three comparisons:

1) level 1 to level 3
2) level 2 to levels 1 and 4
3) levels 1 and 2 to levels 3 and 4.

In order to compare level 1 to level 3, we use the contrast coefficients 1 0 -1 0. To compare level 2 to levels 1 and 4 we use the contrast coefficients -1/2 1 0 -1/2 Finally, to compare levels 1 and 2 with levels 3 and 4 we use the coefficients 1/2 1/2 -1/2 -1/2. Before proceeding to the Stata code necessary to conduct these analyses, let's take a moment to more fully explain the logic behind the selection of these contrast coefficients.

For the first contrast, we are comparing level 1 to level 3, and the contrast coefficients are 1 0 -1 0. This means that the levels associated with the contrast coefficients with opposite signs are being compared. In fact, the mean of the dependent variable is multiplied by the contrast coefficient. Hence, levels 2 and 4 are not involved in the comparison: they are multiplied by zero and "dropped out." You will also notice that the contrast coefficients sum to zero. This is necessary. If the contrast coefficients do not sum to zero, the contrast is not estimable and Stata will issue an error message. Which level of the categorical variable is assigned a positive or negative value is not terribly important: 1 0 -1 0 is the same as -1 0 1 0 in that both of these codings compare the first and the third levels of the variable. However, the sign of the regression coefficient would change.

Now let's look at the contrast coefficients for the second and third comparisons. You will notice that in both cases we use fractions that sum to one (or minus one). They do not have to sum to one (or minus one). You may wonder why we would use fractions like -1/2 1 0 -1/2 instead of whole numbers such as -1 2 0 -1. While -1/2 1 0 -1/2 and -1 2 0 -1 both compare level 2 with levels 1 and 4 and both will give you the same t-value and p-value for the regression coefficient, the regression coefficients themselves would be different, as would their interpretation. The coefficient for the -1/2 1 0 -1/2 contrast is the mean of level 2 minus the mean of the means for levels 1 and 4: 58 - (46.4583 + 54.0552)/2 = 7.74325. (Alternatively, you can multiply the contrasts by the mean of the dependent variable for each level of the categorical variable: -1/2*46.4583 + 1*58.00 + 0*48.20 + -1/2*54.0552 = 7.74325. Clearly these are equivalent ways of thinking about how the contrast coefficient is calculated.) By comparison, the coefficient for the -1 2 0 -1 contrast is two times the mean for level 2 minus the means of the dependent variable for levels 1 and 4: 2*58 - (46.4583 + 54.0552) = 15.4865, which is the same as -1*46.4583 + 2*58 + 0*48.20 - 1*54.0552 = 15.4865. Note that the regression coefficient using the contrast coefficients -1 2 0 -1 is twice the regression coefficient obtained when -1/2 1 0 -1/2 is used.

Method 1: Using xi3

We use the **char** command to indicate the contrast coefficients to be used for **race** as shown below. In order to compare level 1 to level 3, we use the contrast coefficients 1 0 -1 0. To compare level 2 to levels 1 and 4 we use the contrast coefficients -1/2 1 0 -1/2 Finally, to compare levels 1 and 2 with levels 3 and 4, we use the coefficients 1/2 1/2 -1/2 -1/2. These coefficients are used in the **char race[user]** command below. This indicates that for **race** that the user defined contrast is defined as having three contrasts (because **race** has four levels) as (1 0 -1 0 \ -.5 1 0 -.5 \ .5 .5 -.5 -.5).

```
char race[user] (1 0 -1 0 \ -.5 1 0 -.5 \ .5 .5 -.5 -.5)
```

```
xi3 : regress write u.race
u.race            _Irace_1-4          (naturally coded; _Irace_4 omitted)

      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

-----------------------------------------------------------------------------
-
```

```
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
    _Irace_1 |  -1.741667   2.732488    -0.64   0.525    -7.130519
3.647186
    _Irace_2 |   7.743247   2.897186     2.67   0.008     2.029588
13.45691
    _Irace_3 |    1.10158   1.964244     0.56   0.576    -2.772186
4.975347
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

The coefficient for **_Irace_1** corresponds to the first contrast comparing level 1 to level 3 of **race**. The coefficient is the mean of level 1 of **write** minus the mean for level 3 of **write**, and the significance of this is .525, i.e., not significant. The coefficient for **_Irace_2** is 7.743, which is the mean of level 2 minus the mean of level 1 and level 4, and this difference is significant, p = 0.008. The final regression coefficient is 1.1 which is the mean of levels 1 and 2 minus the mean of levels 3 and 4, and this contrast is not statistically significant, p = .576.

Method 2: Manual Coding

As in the prior examples, we will make the following three comparisons:

1) level 1 to level 3,
2) level 2 to levels 1 and 4 and
3) levels 1 and 2 to levels 3 and 4.

The **xi3** command converts the contrast coding into regression coding for us. However, we could do this process manually as well.

For methods 1 and 2 it was quite easy to translate the comparisons we wanted to make into contrast codings, but it is not as easy to translate the comparisons we want into a regression coding scheme. If we know the contrast coding system, then we can convert that into a regression coding system using the Stata program shown below. As you can see, we place the three contrast codings we want into the matrix **c** and then perform a set of matrix operations on **c,** yielding the matrix **x**. We then display **x** using the **print** command.

```
matrix input c = (1 0 -1 0 \ -.5 1 0 -.5 \ .5 .5 -.5 -.5)
matrix x = c'*inv(c*c')
matrix list x
```

```
x[4,3]
      r1    r2    r3
c1   -.5    -1   1.5
c2    .5     1   -.5
c3  -1.5    -1   1.5
c4   1.5     1  -2.5
```

This converted the contrast coding into the regression coding that we need for running this analysis with the **regress** command. Below, we use the **generate** and **replace** commands to create **x1**, **x2** and **x3** according to the coding shown above and then enter them into the regression analysis.

```
generate x1 =  -.5 if race == 1
replace  x1 =   .5 if race == 2
replace  x1 = -1.5 if race == 3
replace  x1 =  1.5 if race == 4
generate x2 =  -1 if race == 1
replace  x2 =   1 if race == 2
replace  x2 =  -1 if race == 3
replace  x2 =   1 if race == 4
generate x3 =  1.5 if race == 1
replace  x3 =  -.5 if race == 2
replace  x3 =  1.5 if race == 3
replace  x3 = -2.5 if race == 4
regress write x1 x2 x3
      Source |       SS       df       MS              Number of obs =
200
-------------+------------------------------           F(  3,   196) =
7.83
       Model |  1914.15805     3  638.052682           Prob > F      =
0.0001
    Residual |   15964.717   196  81.4526375           R-squared     =
0.1071
-------------+------------------------------           Adj R-squared =
0.0934
       Total |   17878.875   199   89.843593           Root MSE      =
9.0251

------------------------------------------------------------------------------
-
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
Interval]
-------------+----------------------------------------------------------------
-
          x1 |  -1.741667   2.732488    -0.64   0.525    -7.130519
3.647186
          x2 |   7.743247   2.897186     2.67   0.008     2.029588
13.45691
          x3 |    1.10158   1.964244     0.56   0.576    -2.772186
4.975347
       _cons |   51.67838    .982122    52.62   0.000     49.74149
53.61526
------------------------------------------------------------------------------
-
```

As you can see, the results of this analysis matches those produced using **xi3**.

5.9 Summary

This page has described a number of different coding systems that you could use for categorical data, and two different strategies you could use for performing the analyses. You can choose a coding system that yields comparisons that make the most sense for testing your hypotheses. Between the two strategies (**xi3** and manual coding), you can see that **xi3** automates the process of creating the coding, but this gives up a certain amount of control. If you like, you can use manual coding which gives you more control over creating the coding of the variables, but may be more laborious and tedious. In general we would recommend using the easiest method that accomplishes your goals.

5.10 Additional Information

Here are some additional resources.

- Stata Textbook Examples from Design and Analysis: Chapter 6
- Stata Textbook Examples from Design and Analysis: Chapter 7
- Stata Textbook Examples: Applied Regression Analysis, Chapter 8
- One-Way ANOVA Contrast Code Problems From Charles Judd and Gary McClelland
- Two-way contrast code solutions