JEM174 Stata for Economists
Institute of Economic Studies
Charles University in Prague

Radek Janhuba
janhuba@gmail.com
Spring 2018

Contents

5 Econometric Methods and Diagnostics

This part of lecture notes will cover regressions and topics associated with them.

5.1 Basics of Linear Regression

We will illustrate the concept of regression in Stata on the `auto` dataset:

```
. sysuse auto, clear
(1978 Automobile Data)
```

5.1.1 Simple Linear Regression

In order to fit a regression, we will write the command `reg`[1], followed by the name of the dependent variable and a list of explanatory variables:

```
. reg mpg weight foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     69.75
       Model |   1619.2877         2  809.643849   Prob > F        =    0.0000
    Residual |  824.171761        71   11.608053   R-squared       =    0.6627
-------------+----------------------------------   Adj R-squared   =    0.6532
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4071

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0006371   -10.34   0.000    -.0078583   -.0053175
     foreign |  -1.650029   1.075994    -1.53   0.130      -3.7955    .4954422
       _cons |    41.6797   2.165547    19.25   0.000     37.36172    45.99768
------------------------------------------------------------------------------
```

Let's now go over the output:
Top left: Analysis of variance (ANOVA)
In this part of the output, we will find

- sums of squares
- degrees of freedom
- mean squares (the first column divided by the second).

Top right: Overall regression statistics.
These are important for comparison of multiple models that have been fit on the same data. They are (from the top)

- Number of observations actually used in the regression
- The F statistic testing that all regressors in the model (excluding the constant) are jointly zero
- P-value of the F statistic above
- R-squared from the regression
- Adjusted R-squared from the regression
- Square root of the mean squared error

Bottom: Regression coefficients
In here we have coefficients of all explanatory variables with the constant being reported at the last place. Each coefficient includes the following:

[1]Note that while the full name of the command is `regress`, it is very often abbreviated to simply `reg`.

- coefficient
- standard error
- t statistic
- p value
- 95% confidence interval

5.1.2 Changing the Confidence Intervals

You can either use an option `level(x)` in the regression itself,

```
.  reg mpg weight foreign, level(90)

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     69.75
       Model |   1619.2877         2  809.643849   Prob > F        =    0.0000
    Residual |  824.171761        71   11.608053   R-squared       =    0.6627
-------------+----------------------------------   Adj R-squared   =    0.6532
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4071

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [90% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0006371   -10.34   0.000    -.0076497   -.0055261
     foreign |  -1.650029   1.075994    -1.53   0.130    -3.443281    .1432223
       _cons |    41.6797   2.165547    19.25   0.000      38.0706     45.2888
------------------------------------------------------------------------------
```

or, if you want to change it after the regression, you can simply type

```
.  reg, level(99)

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     69.75
       Model |   1619.2877         2  809.643849   Prob > F        =    0.0000
    Residual |  824.171761        71   11.608053   R-squared       =    0.6627
-------------+----------------------------------   Adj R-squared   =    0.6532
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4071

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [99% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0006371   -10.34   0.000    -.0082742   -.0049015
     foreign |  -1.650029   1.075994    -1.53   0.130    -4.498039     1.19798
       _cons |    41.6797   2.165547    19.25   0.000     35.94779    47.41161
------------------------------------------------------------------------------
```

5.1.3 Summary Statistics on Regression Sample

If you want to see the summary statistics of the sample from the regression, you can type

```
. estat summarize

Estimation sample regress                 Number of obs =         74

------------------------------------------------------------------------------
    Variable |         Mean      Std. Dev.         Min          Max
-------------+----------------------------------------------------------------
         mpg |      21.2973      5.785503          12           41
      weight |     3019.459      777.1936        1760         4840
     foreign |     .2972973      .4601885           0            1
------------------------------------------------------------------------------
```

Alternatively, you can use the following to find statistics from more (or even all) variables:

```
. sum if e(sample) == 1

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        make |          0
       price |         74    6165.257    2949.496       3291      15906
         mpg |         74     21.2973    5.785503         12         41
       rep78 |         69    3.405797    .9899323          1          5
    headroom |         74    2.993243    .8459948        1.5          5
-------------+---------------------------------------------------------
       trunk |         74    13.75676    4.277404          5         23
      weight |         74    3019.459    777.1936       1760       4840
      length |         74    187.9324    22.26634        142        233
        turn |         74    39.64865    4.399354         31         51
displacement |         74    197.2973    91.83722         79        425
-------------+---------------------------------------------------------
  gear_ratio |         74    3.014865    .4562871       2.19       3.89
     foreign |         74    .2972973    .4601885          0          1
```

Note that the actual regression sample is not necessarily the same as the sample you specified in the regression command (possibly with `if` and `in`). The reason is that Stata automatically throws out observations with missing values in any of the variables included in the model.

5.1.4 Information Criteria

When comparing similar models on the same set of data, we can use adjusted R-squared, mean square error, or information criteria. The general rule of thumb is that you would like to select the model with the lowest information criterion (note that this very often means that you select the "highest negative" value). You can pull the ICs up by typing

```
. estat ic

Akaike's information criterion and Bayesian information criterion
```

```
------------------------------------------------------------------------------
       Model |        Obs  ll(null)  ll(model)      df          AIC        BIC
-------------+----------------------------------------------------------------
           . |         74 -234.3943  -194.1831       3    394.3661   401.2783
------------------------------------------------------------------------------
               Note: N=Obs used in calculating BIC; see [R] BIC note.
```

Note: the command `estat` is a general command designed to calculate various post-estimation statistics. The list of subcommands available depends on which command was used in the estimation. See `help estat` for general information and help *command*`_postestimation` for a list of statistics available after a specific command.

5.2 Recovering Results After a Regression

The function `e(sample)` above is a special function that is equal to 1 for those observations that were included in the sample and 0 otherwise. In fact, all Stata's estimation commands store a range of results and other estimation regarding information to e-class results. These are similar to r-class results stored after general commands. You can find their list and contents by typing

```
. ereturn list

scalars:
                  e(rank) =  3
                  e(ll_0) =  -234.3943376482347
                    e(ll) =  -194.1830643938065
                  e(r2_a) =  .6532015851691599
                   e(rss) =  824.1717612920727
                   e(mss) =  1619.287698167387
                  e(rmse) =  3.407059285651584
                    e(r2) =  .6627029116028815
                     e(F) =  69.74846262000308
                  e(df_r) =  71
                  e(df_m) =  2
                     e(N) =  74

macros:
              e(cmdline) : "regress mpg weight foreign, level(90)"
                e(title) : "Linear regression"
            e(marginsok) : "XB default"
                  e(vce) : "ols"
                e(depvar) : "mpg"
                  e(cmd) : "regress"
           e(properties) : "b V"
              e(predict) : "regres_p"
            e(estat_cmd) : "regress_estat"

matrices:
                    e(b) :  1 x 3
                    e(V) :  3 x 3
```

```
functions:
             e(sample)
```

5.2.1 Scalars

Scalar variables can hold any value.[2] Essentially, you can assign any value here that is identical for all observations and then use them for calculations. You can e.g.:

- Calculate with them

```
.  display e(N)/2
37
```

- Assign them into a named scalar for later use

```
.  scalar ll_first = e(ll)

. display ll_first
-194.18306
```

- use them while creating new variables

```
.  gen r2 = e(r2)

. gen r2manual = e(mss) / (e(rss)+e(mss)) if e(sample)==1
```

5.2.2 Macros

For now, you can think about macros as objects storing some text - we will cover them in more detail in the last part of the course.

5.2.3 Matrices

As you know from other classes, matrices are arrays of values of some height and width. By default, Stata stores a matrix of regression coefficients into the matrix `e(b)`[3] and does the same with the variance-covariance into the matrix `e(V)`. You can list these matrices using

```
.  matrix list e(b)

e(b)[1,3]
        weight     foreign       _cons
y1  -.00658789  -1.6500291   41.679702

. matrix list e(V)

symmetric e(V)[3,3]
```

[2]Scalars can actually hold a string as well, however, we do not need to go into the details.
[3]Note that in this case, it actually is a vector, as it only has one row.

```
             weight     foreign       _cons
 weight   4.059e-07
foreign    .0004064   1.1577633
  _cons  -.00134646  -1.5713159   4.6895946
```

Generally, if you want to access any part of a matrix, you can access it by typing *matname*`[r,c]` where r is the row number and column is the column number.

However, for some technical reason, one cannot access elements in this way in case of these `ereturn` matrices. Therefore, in case we want to access it directly, we need to first store the matrix into the memory:

```
.  matrix B = e(b)

. matrix list B

B[1,3]
        weight     foreign       _cons
y1  -.00658789  -1.6500291   41.679702

. display B[1,1] // coefficient on weight
-.00658789

. display B[1,2] // constant
-1.6500291
```

Luckily, regression also stores coefficient values and their standard errors into matrices *_b* and *_se*, so you do not actually have to store them in order to access them. Moreover, you can use the name of the variable in question to access the particular coefficient:

```
.  display _b[weight] // gets coefficient of weight
-.00658789

. display _se[weight] // gets standard error of the above
.00063711

. display _b[_cons]/_se[_cons] // gets t-statistic of the constant
19.24673
```

5.3 Specifying Variables Using Factor Notation

5.3.1 Categorical Variables

Suppose now we want to look at whether the repair record of the car has effect as well on the mileage it can travel.

Suppose more that we treat the repair record as a categorical rather than a continuous variable, in sense that increase from 1 to 2 is qualitatively different than e.g. an increase from 4 to 5. Therefore, we need to add it as a categorical rather than continuous variable.

Of course, we could do everything by manually generating the variables we need (e.g. using the command `tab rep78, gen(rep)`) and then running the regression with these new variables. Instead, we will use one of Stata's great functionalities, *factor notation*. This allows to run the regression without actually needing to create the particular dummies:

```
. reg mpg weight foreign i.rep78

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(6, 62)        =     23.55
       Model |  1626.43408         6  271.072346   Prob > F        =    0.0000
    Residual |  713.768822        62  11.5124004   R-squared       =    0.6950
-------------+----------------------------------   Adj R-squared   =    0.6655
       Total |   2340.2029        68  34.4147485   Root MSE        =     3.393

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0062911    .000687    -9.16   0.000    -.0076644   -.0049177
     foreign |  -2.923268   1.342996    -2.18   0.033    -5.607878    -.238658
             |
       rep78 |
          2  |  -.2786445   2.688056    -0.10   0.918    -5.651992    5.094703
          3  |  -.0224197   2.489182    -0.01   0.993    -4.998224    4.953384
          4  |   .6813578   2.599968     0.26   0.794    -4.515903    5.878619
          5  |   3.865535   2.770732     1.40   0.168     -1.67308    9.404149
             |
       _cons |   40.50227   3.208188    12.62   0.000      34.0892    46.91535
------------------------------------------------------------------------------
```

5.3.2 Interaction Terms

The previous section showed us how to add categorical dummies into the regression. Similarly, this section will explain how to add interactions between the variables without the actual need to create these variables.

Suppose we suspect that effect of weight on mpg may be different for domestic and foreign cars. In such case, we want to include interaction parameter for these two variables. Using the factor notation, the regression is specified as

```
. reg mpg weight foreign c.weight#foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(3, 70)        =     51.99
       Model |  1686.54824         3  562.182746   Prob > F        =    0.0000
    Residual |  756.911221        70  10.8130174   R-squared       =    0.6902
-------------+----------------------------------   Adj R-squared   =    0.6770
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.2883

------------------------------------------------------------------------------
             mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-----------------+----------------------------------------------------------------
          weight |  -.0059751   .0006622    -9.02   0.000    -.0072958   -.0046544
         foreign |   9.271333   4.500409     2.06   0.043     .2955505    18.24711
                 |
foreign#c.weight |
         Foreign |  -.0044509   .0017846    -2.49   0.015    -.0080101   -.0008916
                 |
```

```
           _cons |   39.64696   2.243364    17.67   0.000     35.17272    44.12121
------------------------------------------------------------------------------
```

where the `c.` marks that *weight* is a continuous variable (do not forget that as in case you do, Stata will treat it as a categorical variable and will create a dummy for each of its values) and a single hashtag identifies the interaction of the two variables.

We can also write

```
. reg mpg c.weight##foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(3, 70)        =     51.99
       Model |  1686.54824         3  562.182746   Prob > F        =    0.0000
    Residual |  756.911221        70  10.8130174   R-squared       =    0.6902
-------------+----------------------------------   Adj R-squared   =    0.6770
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.2883

------------------------------------------------------------------------------
             mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-----------------+------------------------------------------------------------------
          weight |  -.0059751   .0006622    -9.02   0.000    -.0072958   -.0046544
                 |
         foreign |
         Foreign |   9.271333   4.500409     2.06   0.043     .2955505    18.24711
                 |
foreign#c.weight |
         Foreign |  -.0044509   .0017846    -2.49   0.015    -.0080101   -.0008916
                 |
           _cons |   39.64696   2.243364    17.67   0.000     35.17272    44.12121
------------------------------------------------------------------------------
```

which is the shortest way we can do it - the double hashtag specifies that the two variables are to be represented on their own as well as in the interaction term.

In the case of categorical variables, we can simply use their names and they will be added as dummies:

```
. reg mpg weight foreign##rep78 // adds levels as well as interactions
note: 1.foreign#1b.rep78 identifies no observations in the sample
note: 1.foreign#2.rep78 identifies no observations in the sample
note: 1.foreign#5.rep78 omitted because of collinearity

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(8, 60)        =     17.55
       Model |  1639.58652         8  204.948314   Prob > F        =    0.0000
    Residual |  700.616383        60  11.6769397   R-squared       =    0.7006
-------------+----------------------------------   Adj R-squared   =    0.6607
       Total |   2340.2029        68  34.4147485   Root MSE        =    3.4172

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

```
-------------+----------------------------------------------------------------
      weight |  -.0063232   .0007639    -8.28   0.000    -.0078512   -.0047951
             |
     foreign |
    Foreign  |  -2.863401   2.692694    -1.06   0.292    -8.249592     2.52279
             |
       rep78 |
          2  |  -.2704992   2.708443    -0.10   0.921    -5.688191    5.147193
          3  |   .1639244   2.517795     0.07   0.948    -4.872416    5.200265
          4  |   .1774528    2.69164     0.07   0.948    -5.206629    5.561535
          5  |   3.791603   3.526377     1.08   0.287    -3.262202    10.84541
             |
foreign#rep78 |
  Foreign#1  |          0  (empty)
  Foreign#2  |          0  (empty)
  Foreign#3  |   -1.85943   3.676071    -0.51   0.615    -9.212668    5.493808
  Foreign#4  |    .933177   3.399182     0.27   0.785    -5.866199    7.732553
  Foreign#5  |          0  (omitted)
             |
       _cons |   40.60178   3.383268    12.00   0.000     33.83424    47.36933
-------------------------------------------------------------------------------
```

Note that the range of coefficients we can include is pretty much limited only by imagination. We can e.g. write

```
. reg mpg c.weight##c.price##rep78
note: 5.rep78#c.price omitted because of collinearity
note: 5.rep78#c.weight#c.price omitted because of collinearity

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(17, 51)       =     15.98
       Model |   1970.2938        17  115.899635   Prob > F        =    0.0000
    Residual |  369.909097        51  7.25311955   R-squared       =    0.8419
-------------+----------------------------------   Adj R-squared   =    0.7892
       Total |   2340.2029        68  34.4147485   Root MSE        =    2.6932

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0208668   .0074048    -2.82   0.007    -.0357326    -.006001
       price |  -.0059557   .0033807    -1.76   0.084    -.0127428    .0008314
             |
c.weight#c.price |   2.44e-06   1.12e-06     2.18   0.034     1.95e-07    4.69e-06
             |
       rep78 |
          2  |  -46.94406   50.25463    -0.93   0.355    -147.8344    53.94633
          3  |  -37.48727   23.30371    -1.61   0.114    -84.27142    9.296874
          4  |  -26.86595   24.07101    -1.12   0.270    -75.19052    21.45861
          5  |   35.97223    17.9413     2.00   0.050    -.0464304    71.99089
             |
rep78#c.weight |
```

```
                    2  |   .0167883   .0135092     1.24   0.220    -.0103326    .0439092
                    3  |   .0147675    .007638     1.93   0.059    -.0005664    .0301015
                    4  |   .0098239   .0080854     1.22   0.230    -.0064082    .0260559
                    5  |  -.0166062   .0064917    -2.56   0.014    -.0296388   -.0035736
                       |
         rep78#c.price |
                    2  |   .0095385   .0121951     0.78   0.438    -.0149442    .0340212
                    3  |   .0047084   .0036019     1.31   0.197    -.0025228    .0119396
                    4  |   .0034131    .003709     0.92   0.362     -.004033    .0108591
                    5  |          0  (omitted)
                       |
rep78#c.weight#c.price |
                    2  |  -3.38e-06   3.18e-06    -1.06   0.293    -9.77e-06    3.01e-06
                    3  |  -2.14e-06   1.16e-06    -1.84   0.071    -4.46e-06    1.90e-07
                    4  |  -1.45e-06   1.24e-06    -1.18   0.245    -3.94e-06    1.03e-06
                    5  |          0  (omitted)
                       |
                 _cons |   77.98403    22.3105     3.50   0.001     33.19382    122.7742
------------------------------------------------------------------------------------------
```

which will include all single variables, all twoway interactions as well as a threeway interaction term.

5.4 Testing of Parameters

Let's now return to the following form of the regression:

```
. reg mpg weight foreign i.rep78

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(6, 62)        =     23.55
       Model |  1626.43408         6  271.072346   Prob > F        =    0.0000
    Residual |  713.768822        62  11.5124004   R-squared       =    0.6950
-------------+----------------------------------   Adj R-squared   =    0.6655
       Total |   2340.2029        68  34.4147485   Root MSE        =     3.393

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0062911    .000687    -9.16   0.000    -.0076644   -.0049177
     foreign |  -2.923268   1.342996    -2.18   0.033    -5.607878    -.238658
             |
       rep78 |
          2  |  -.2786445   2.688056    -0.10   0.918    -5.651992    5.094703
          3  |  -.0224197   2.489182    -0.01   0.993    -4.998224    4.953384
          4  |   .6813578   2.599968     0.26   0.794    -4.515903    5.878619
          5  |   3.865535   2.770732     1.40   0.168     -1.67308    9.404149
             |
       _cons |   40.50227   3.208188    12.62   0.000      34.0892    46.91535
------------------------------------------------------------------------------
```

5.4.1 Simple Testing for Values

Unsurprisingly, simple tests of parameter values are performed using the command `test`. The command can e.g.

a) test for coefficients equal to zero by simply typing their names:[4]

```
. test weight

 ( 1)  weight = 0

       F(  1,    62) =   83.84
            Prob > F =    0.0000
```

b) test for coefficients equal to any value by specifying the value:

```
. test foreign = -3

 ( 1)  foreign = -3

       F(  1,    62) =    0.00
            Prob > F =    0.9546
```

c) test for coefficients equal to each other

```
. test 4.rep78 = 5.rep78

 ( 1)  4.rep78 - 5.rep78 = 0

       F(  1,    62) =    5.51
            Prob > F =    0.0221
```

d) test for multiple coefficients being jointly zero:

```
. test 2.rep78 3.rep78 4.rep78 5.rep78

 ( 1)  2.rep78 = 0
 ( 2)  3.rep78 = 0
 ( 3)  4.rep78 = 0
 ( 4)  5.rep78 = 0

       F(  4,    62) =    1.96
            Prob > F =    0.1120
```

e) jointly test multiple conditions by separating them in parenthesis:

[4]Note that this test leads to equal p-values as coefficients in the table, with the distinction that one is from a t-test and the other one from the F-test.

```
. test (weight) (foreign = -3)

 ( 1)  weight = 0
 ( 2)  foreign = -3

       F(  2,    62) =   58.42
            Prob > F =    0.0000
```

If we want to test each condition separately as well as jointly, we can specify the option `mtest`, optionally with the correction for number of tests (the correction of the method is actually optional - as the help states, writing `mtest` without an argument means we do not want to adjust p-values).

```
. test (weight) (foreign = -3), mtest

 ( 1)  weight = 0
 ( 2)  foreign = -3

------------------------------------
       |    F(df,62)     df       p
-------+----------------------------
  (1)  |       83.84      1     0.0000 #
  (2)  |        0.00      1     0.9546 #
-------+----------------------------
  all  |       58.42      2     0.0000
------------------------------------
                 # unadjusted p-values
```

Note that the command

```
. test i.rep78
i:  operator invalid
r(198);
```

leads to an error. This is because the command `test` does not understand this type of notation. Luckily, the command `testparm` saves the situation:[5]

```
. testparm i.rep78

 ( 1)  2.rep78 = 0
 ( 2)  3.rep78 = 0
 ( 3)  4.rep78 = 0
 ( 4)  5.rep78 = 0

       F(  4,    62) =    1.96
            Prob > F =    0.1120
```

[5]The reason for this behavior actually is that `test` takes *coeflist* as an argument, while `testparm` uses *varlist*.

Note: The difference between the two comments is pretty much only in the fact that `test` cannot understand wildcards. The most intuitive way how to remember it is the following: `test` can only perform tests on specific list of variables, while `testparm` can test any specification that can be written in the `reg` command.

Before we continue to combinations of parameters, note that if you specify the regression as

```
.reg mpg c.weight##foreign // output omitted
```

you will need to add "`1.`" into the coefficient name:

```
.  test foreign = -3
regressor foreign not found
r(111);

. test 1.foreign = -3

 ( 1)  1.foreign = -3

       F(  1,    70) =    7.43
            Prob > F =    0.0081
```

The reason is the factor notation. In cases you are not sure what the coefficient name is, you can list the coefficient matrix and the names of particular coefficients will show up above their values:

```
.  matrix list e(b)

e(b)[1,6]
                           0b.            1.  0b.foreign#   1.foreign#
         weight      foreign      foreign    co.weight     c.weight        _cons
y1   -.00597508            0    9.2713327            0   -.00445087    39.646965
```

5.4.2 Simple Linear Combinations

Let's now get back to the same regression specified above:

```
.reg mpg weight foreign i.rep78 // output omitted - see above
```

Commands from the previous section allow you to test jointly for values of some coefficients, however, do not actually allow you to test for combinations of these parameters. Linear combinations of such parameters can be calculated by the command `lincom`. The syntax is

```
lincom equation
```

where "equation" stands for a combination of parameters that should evaluate to zero. Suppose e.g. that we want to test for the effect of *500*weight* being opposite to the effect of *foreign*. In principle, this can be done by typing

```
. test 500*weight = - foreign

 ( 1)  500*weight + foreign = 0

       F(  1,    62) =   15.30
            Prob > F =    0.0002
```

however, we do not see the size and confidence interval of this effect. It is therefore better to write

```
. lincom 500*weight + foreign

 ( 1)  500*weight + foreign = 0

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |  -6.068796   1.551724    -3.91   0.000    -9.170649   -2.966943
------------------------------------------------------------------------------
```

which will show us the information.

5.4.3 General Parameter Combinations

The command `lincom` is very powerful for linear restrictions, however, unfortunately it cannot calculate non-linear restrictions. These are calculated using the command `nlcom`, which is its generalization. In principle, one should use `lincom` to calculate values and confidence intervals for single linear restrictions, and use `nlcom` in two cases:

a) to test for nonlinear combinations:

```
. nlcom _b[weight] / _b[foreign]

       _nl_1:  _b[weight] / _b[foreign]

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |   .0021521   .0008876     2.42   0.015     .0004124    .0038918
------------------------------------------------------------------------------
```

b) to test for combinations of (also linear) restrictions:

```
. nlcom (500*_b[weight] + _b[foreign]) (_b[2.rep78] + _b[4.rep78])

       _nl_1:  500*_b[weight] + _b[foreign]
       _nl_2:  _b[2.rep78] + _b[4.rep78]

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
```

```
-------------+----------------------------------------------------------------
       _nl_1 |  -6.068796   1.551724    -3.91   0.000     -9.11012   -3.027472
       _nl_2 |   .4027133   5.056298     0.08   0.937    -9.507449    10.31288
-------------------------------------------------------------------------------
```

Note the difference in syntax of `lincom` and `nlcom` - in the first case, you can use *X* for a variable, in the latter, you need to use `_b[X]` for a coefficient, otherwise Stata returns a (very informative) error:

```
. nlcom (500*weight + foreign) (2.rep78 + 4.rep78)

expression (500*weight + foreign) contains reference to X rather than _b[X]
r(198);
```

5.5 Predictions and Forecasting

Let's now come back to the simplest imaginable regression:

```
. reg mpg weight

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =    134.62
       Model |   1591.9902         1   1591.9902   Prob > F        =    0.0000
    Residual |  851.469256        72  11.8259619   R-squared       =    0.6515
-------------+----------------------------------   Adj R-squared   =    0.6467
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4389

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0060087   .0005179   -11.60   0.000    -.0070411   -.0049763
       _cons |   39.44028   1.614003    24.44   0.000     36.22283    42.65774
------------------------------------------------------------------------------
```

5.5.1 Obtaining Predicted Values

Very often we want to obtain the fitted values from the regression. Command predict is very useful in that. We can get these as

```
. predict pmpg, xb
```

where *pmpg* is a name of the new variable we want to create and `xb` tells Stata that we want predicted values to be stored in it. Note that you do not actually need to specify the `xb` option in case of a linear regression as (in this case) `xb` actually is the prediction:

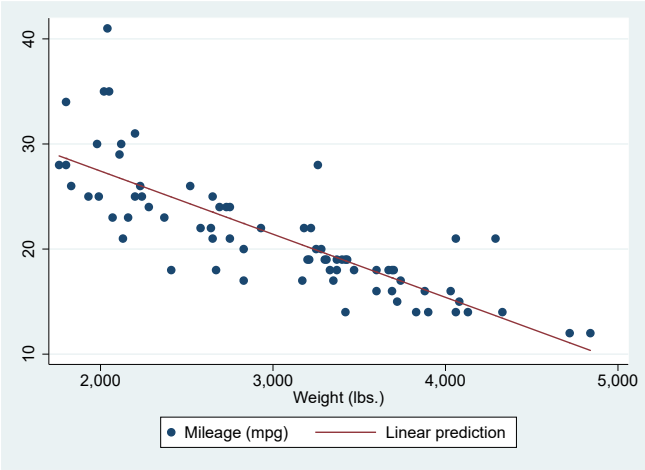
```
. predict pmpg2
(option xb assumed; fitted values)

. list make mpg pmpg pmpg2 in 1/3 // pmpg and pmpg2 are identical
```

```
     +----------------------------------------+
     | make          mpg       pmpg      pmpg2 |
     |----------------------------------------|
  1. | AMC Concord    22   21.83483   21.83483 |
  2. | AMC Pacer      17   19.31118   19.31118 |
  3. | AMC Spirit     22   23.57735   23.57735 |
     +----------------------------------------+
```

5.5.2 Drawing Predicted Lines: Simple Regression

Let's now draw a scatter plot overlayed with the regression line in the most basic regression. The most straightforward way to do it is to overlay the two graphs:

```
. twoway (scatter mpg weight) (line pmpg weight, sort(weight))
```

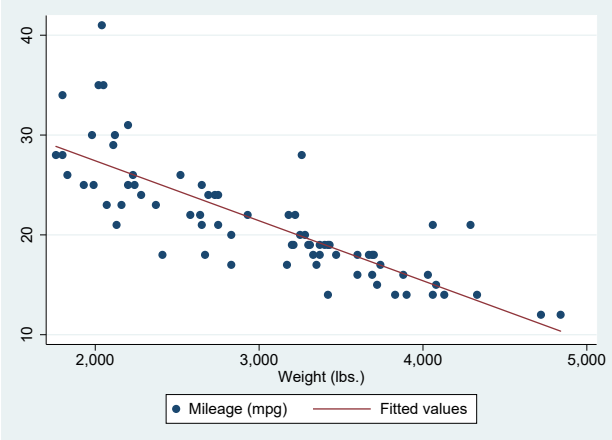


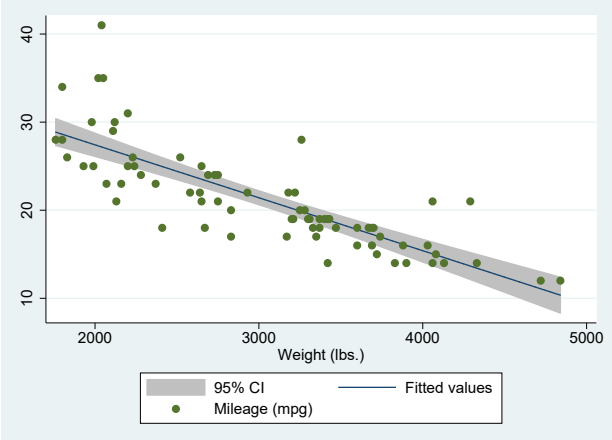
Alternatively, we can use

```
. twoway (scatter mpg weight) (lfit mpg weight) // left
```

or even

```
. twoway (lfitci mpg weight) (scatter mpg weight) // right
```





What we used in the previous two commands are new types of twoway graphs that we did not cover in the graphics part. These graphs take regression lines of simple regression of `yvar` on `xvar` and plot the regression line, optionally along with its confidence intervals. Their usage has some advantages and some disadvantages:
Advantages:

1. Very illustrative
2. Less typing compared to the manual method
3. It's not actually necessary to perform the regression

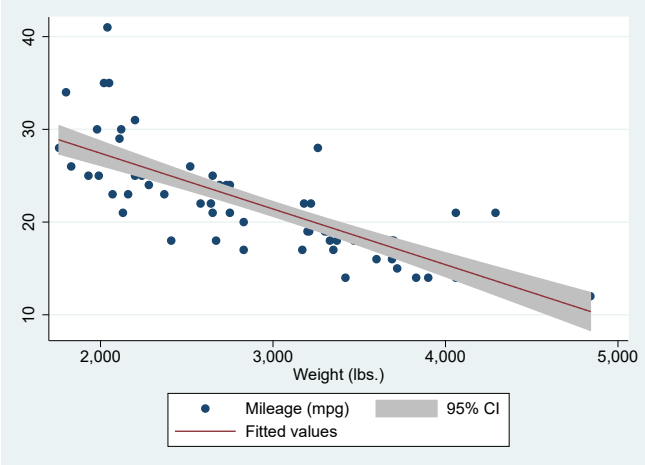
Disadvantages:

1. Only take into account single explanatory variable (see below)
2. It's not actually necessary to perform the regression

Notes:

- the "disadvantage 2" above is meant in a sense that one may actually draw this plot and then try to interpret the graph without actually seeing the estimates. This is very dangerous!
- there is also a twoway graph `qfit` and `qfitci` which shows regression lines from `yvar` on `xvar` and `xvar`^{2}, otherwise it's identical.
- If you use the lfitci command, put it in front of the scatter. If you code it after the scatter, the confidence intervals will overwrite the dots that are inside:

```
. twoway (scatter mpg weight) (lfitci mpg weight)
```



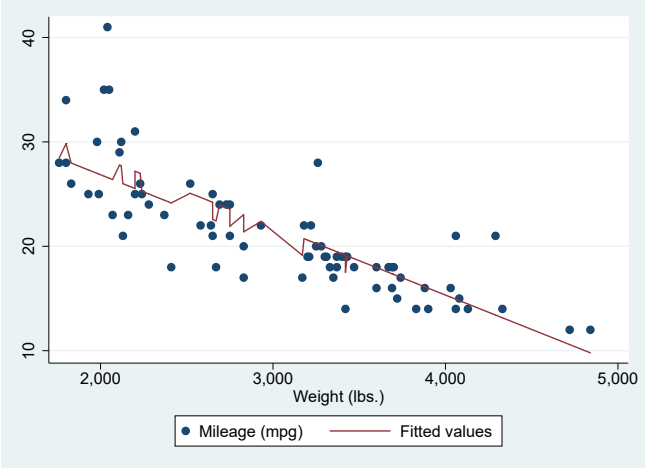
5.5.3 Drawing Predicted Lines: Multiple Regression

Suppose now we want to fit a regression including the information about car's origin:

```
. quietly reg mpg weight foreign // output omitted
```

If we save the prediction and plot it in the similar way as in the previous section, the resulting figure will not be a straight line:

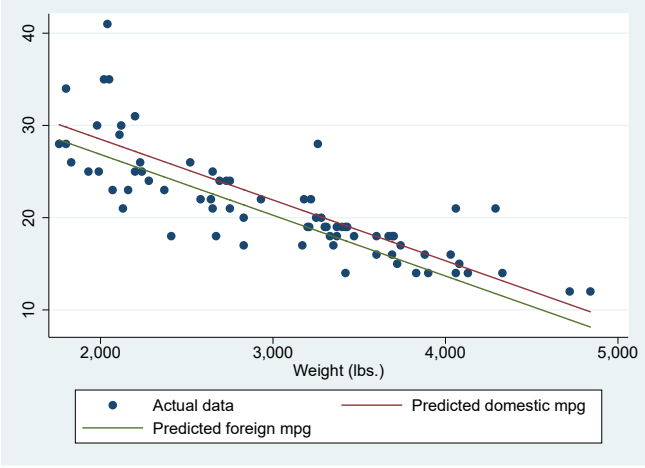
```
.  predict pmpg_for
(option xb assumed; fitted values)

. twoway (scatter mpg weight) (line pmpg_for weight, sort(weight))
```



The reason for this is that the prediction calculated this way incorporates the information about whether a particular car is domestic or foreign, while the predicted lines should be drawn *ceteris paribus*.

In order to draw a regression line against some predictor in a multiple regression, you can use the following way:

```
.  qui reg mpg weight foreign // output omitted

. gen pred_dom = _b[_cons] + _b[weight]*weight // domestic prediction

. gen pred_for = _b[_cons] + _b[weight]*weight + _b[foreign] // foreign prediction

. twoway (scatter mpg weight) (line pred_dom weight, sort(weight)) ///
       (line pred_for weight, sort(weight)), legend(order(1 "Actual data" ///
           2 "Predicted domestic mpg" 3 "Predicted foreign mpg"))
```

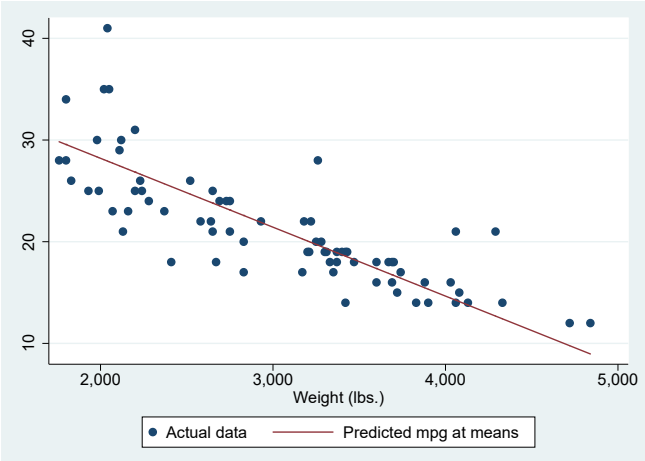


Suppose now that you want to draw a single prediction line only, evaluated at means of other explanatory variables. In order to illustrate the concept, let's now add *price* into the regression:

```
.  reg mpg weight price foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(3, 70)        =     45.93
       Model |  1620.30716         3  540.102388   Prob > F        =    0.0000
    Residual |  823.152295        70  11.7593185   R-squared       =    0.6631
-------------+----------------------------------   Adj R-squared   =    0.6487
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4292

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0067758   .0009048    -7.49   0.000    -.0085805   -.0049712
       price |   .0000566   .0001922     0.29   0.769    -.0003268       .00044
     foreign |  -1.855891   1.289063    -1.44   0.154    -4.426846    .7150641
       _cons |   41.95948   2.377726    17.65   0.000     37.21725     46.7017
------------------------------------------------------------------------------
```

In order to create the correct prediction, we first have to evaluate the sample means and store them into scalars. Afterwards we will generate the predicted variable and plot it:

```
.  qui sum price if e(sample) // quietly summarize price

. scalar m_price = r(mean) // store mean into scalar

. qui sum foreign if e(sample) // do the same for foreign

. scalar m_foreign = r(mean) // store mean into scalar

. gen pred_means = _b[_cons] + _b[weight]*weight + _b[price]*m_price + ///
                                _b[foreign]*m_foreign

. twoway (scatter mpg weight) (line pred_means weight, sort(weight)), ///
                legend(order(1 "Actual data" 2 "Predicted mpg at means"))
```



Note that this particular approach to the solution has a substantial caveat in sense that one needs to code all the means manually. We will cover an automatic solution in the last part of the course when we know how to use macros and loops.

5.5.4 Using Predicted Values for Out-of-Sample Predictions/Forecasting

Let's now illustrate another useful concept that fitted values can be used for. Suppose we fit a regression in the form

```
.  reg mpg weight foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     69.75
       Model |   1619.2877         2  809.643849   Prob > F        =    0.0000
    Residual |  824.171761        71   11.608053   R-squared       =    0.6627
-------------+----------------------------------   Adj R-squared   =    0.6532
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4071

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0006371   -10.34   0.000    -.0078583   -.0053175
     foreign |  -1.650029   1.075994    -1.53   0.130      -3.7955    .4954422
       _cons |    41.6797   2.165547    19.25   0.000     37.36172    45.99768
------------------------------------------------------------------------------
```

We now have an estimated model in memory and we can use its power to calculate out-of-sample predictions. Let's now load a web based example dataset containing some information on new cars:

```
.  webuse newautos, clear
(New Automobile Models)

. list

     +-----------------------------------+
     |          make   weight    foreign |
     |-----------------------------------|
  1. | Pont. Sunbird     2690   Domestic |
  2. |     Volvo 260     3170    Foreign |
     +-----------------------------------+
```

we can see that there are two new cars, on which we only know how much they weight and whether they are domestic or foreign, but nothing about mileage. What is the best prediction of their mileage based on what we know so far? Arguably, it should be the fitted value based on the regression we ran. We can get this prediction as

```
.  predict pmpg, xb

. list

     +-------------------------------------------+
```

```
     |          make   weight    foreign       pmpg |
     |-----------------------------------------------|
  1. | Pont. Sunbird     2690   Domestic   23.95829 |
  2. |     Volvo 260     3170    Foreign   19.14607 |
     +-----------------------------------------------+
```

We can see that we now have an idea about how much these new autos should go per a gallon of gas. However, it is extremely unlikely that this would be a correct value (after all, statistically speaking, the probability of a point estimate is always zero).

Therefore, we should rather be interested in the confidence interval of the prediction. In order to calculate it, we first need to obtain the standard error of the prediction as

```
. predict pmpg_se, stdf
```

The option `stdf` tells Stata that you want to save the "standard error of the forecast" into the new variable.

Finally, we know that approximately 95% of the data usually lies within 2 standard deviations from the expected value. Therefore, if we want to obtain confidence intervals, we can write:

```
. gen pmpg_ci_low = pmpg-2*pmpg_se

. gen pmpg_ci_high = pmpg+2*pmpg_se

. list
```

```
     +-------------------------------------------------------------------------------+
     |          make   weight    foreign       pmpg    pmpg_se   pmpg_c~w   pmpg_c~h |
     |-------------------------------------------------------------------------------|
  1. | Pont. Sunbird     2690   Domestic   23.95829   3.462791   17.03271   30.88387 |
  2. |     Volvo 260     3170    Foreign   19.14607   3.525875   12.09432   26.19782 |
     +-------------------------------------------------------------------------------+
```

We now have 95% confidence intervals for the two cars, which is a much more reliable measure.

Before we continue, notice that Stata offers one more option for prediction, `stdp`. This stands for a standard error of *prediction* rather than *forecast*. The difference between the two is that `stdf` adds additional uncertainty from predicting a new value of the dependent variable, so it produces a bigger standard error.[6]

```
. predict pmpg_se_p, stdp

. list make weight foreign pmpg pmpg_se pmpg_se_p
```

```
     +-----------------------------------------------------------------------+
     |          make   weight    foreign       pmpg    pmpg_se   pmpg_s~p |
     |-----------------------------------------------------------------------|
  1. | Pont. Sunbird     2690   Domestic   23.95829   3.462791   .6187622 |
  2. |     Volvo 260     3170    Foreign   19.14607   3.525875   .9076015 |
     +-----------------------------------------------------------------------+
```

[6] The rule of thumb which to use goes as follows:
In sample prediction -> have y -> stdp
Out of sample forecast -> do not have y -> stdf

5.6 Classical Regression Assumptions

The following part of lecture notes draws on Baum (2006). Let's load the dataset *hprice2a* associated with this book.

```
. use hprice2a, clear
(Housing price data for Boston-area communities)
```

The dataset contains information on 506 communities in Boston area. In this regression, we will be looking at the relationship of the following variables:

```
. describe price nox dist rooms stratio

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
price           float   %9.0g                 median housing price, $
nox             float   %9.0g                 nitrous oxide, parts per 100m
dist            float   %9.0g                 weighted dist. to 5 employ centers
rooms           float   %9.0g                 avg number of rooms per house
stratio         float   %9.0g                 average student-teacher ratio
```

Specifically, we will be looking at the effects of the four variables on logarithmic price. Air pollution and distance to community centers will also be specified in logarithm.

```
. reg lprice lnox ldist rooms stratio

      Source |       SS           df       MS      Number of obs   =       506
-------------+----------------------------------   F(4, 501)       =    175.86
       Model |  49.3987735         4  12.3496934   Prob > F        =    0.0000
    Residual |  35.1834974       501  .070226542   R-squared       =    0.5840
-------------+----------------------------------   Adj R-squared   =    0.5807
       Total |  84.5822709       505  .167489645   Root MSE        =      .265

------------------------------------------------------------------------------
      lprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        lnox |    -.95354   .1167418    -8.17   0.000    -1.182904   -.7241762
       ldist |  -.1343401   .0431032    -3.12   0.002    -.2190255   -.0496548
       rooms |   .2545271   .0185303    13.74   0.000     .2181203    .2909338
     stratio |  -.0524512   .0058971    -8.89   0.000    -.0640373   -.0408651
       _cons |   11.08387   .3181115    34.84   0.000     10.45887    11.70886
------------------------------------------------------------------------------
```

We can see that all the coefficients are statistically significant and all are in expected signs. Let's now try to identify whether classical regression assumptions are satisfied.

5.6.1 Multicollinearity

The easiest way to detect multicollinearity is to look at variance inflation factors, where the rule of thumb is that in case the largest VIF is more than 10, there is an issue of collinearity. In such a case,

it may be a good idea to explore the correlation between the explanatory variables and drop some coefficients from the regression.

```
. estat vif

    Variable |       VIF       1/VIF
-------------+----------------------
        lnox |      3.98    0.251533
       ldist |      3.89    0.257162
       rooms |      1.22    0.820417
     stratio |      1.17    0.852488
-------------+----------------------
    Mean VIF |      2.56
```

We can see that in this case there are no VIFs even close to the value of 10 so there is not an issue of multicollinearity.

5.6.2 Omitted Variables

In order to test for whether there are some omitted variables in the model, we can use Ramsey RESET specification test:

```
. estat ovtest

Ramsey RESET test using powers of the fitted values of lprice
       Ho:  model has no omitted variables
                 F(3, 498) =      9.69
                  Prob > F =      0.0000

. estat ovtest, rhs

Ramsey RESET test using powers of the independent variables
       Ho:  model has no omitted variables
                F(12, 489) =     11.79
                  Prob > F =      0.0000
```

This test adds second, third and fourth powers of either the dependent variable or all of the right hand side variables (depending whether the option rhs is selected) into the regression and then tests the null hypothesis that there are no omitted variables from the model.

In our case, we can see that the RESET test rejects the null hypothesis under all imaginable levels of confidence. Therefore, we would probably conclude that there is something missing. In practice, we would now most probably start looking at other possible variables and/or add higher powers into the model, however, as this is only a demonstration of the RESET test, we will not proceed further.

5.6.3 Heteroskedasticity

To informally analyze the data graphically, you can use the command

```
. rvfplot
```

to check for residuals versus fitted values (note that in this case, the test suggests that there definitely is an issue of heteroskedasticity). You can also use the command

```
. rvpplot ldist
```

to check the values against individual predictors (note that this graph also definitely suggests a presence of heteroskedasticity).

In order to test for heteroskedasticity formally, you can use the command

```
. estat hettest

Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
         Ho: Constant variance
         Variables: fitted values of lprice

         chi2(1)      =   127.21
         Prob > chi2  =   0.0000

. estat hettest, rhs

Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
         Ho: Constant variance
         Variables: lnox ldist rooms stratio

         chi2(4)      =   236.55
         Prob > chi2  =   0.0000
```

Alternatively, you can use the information matrix test which also tests for normality using a test for skewness and kurtosis:

```
. estat imtest

Cameron & Trivedi's decomposition of IM-test

---------------------------------------------------
              Source |       chi2     df      p
---------------------+-----------------------------
  Heteroskedasticity |     143.98     14    0.0000
            Skewness |      16.99      4    0.0019
            Kurtosis |      11.30      1    0.0008
---------------------+-----------------------------
               Total |     172.26     19    0.0000
---------------------------------------------------
```

If we conclude that there is heteroskedasticity in the data, we can specify the robust standard errors:

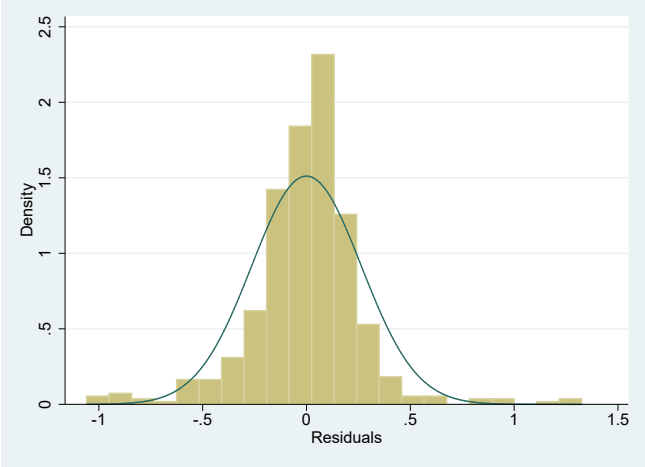
```
. reg lprice lnox ldist rooms stratio, vce(robust)

Linear regression                               Number of obs     =        506
```

```
                                                F(4, 501)         =     146.27
                                                Prob > F          =     0.0000
                                                R-squared         =     0.5840
                                                Root MSE          =       .265

------------------------------------------------------------------------------
             |               Robust
      lprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        lnox |    -.95354   .1268006    -7.52   0.000    -1.202667   -.7044135
       ldist |  -.1343401   .0535287    -2.51   0.012    -.2395086   -.0291717
       rooms |   .2545271   .0247204    10.30   0.000     .2059586    .3030956
     stratio |  -.0524512   .0046082   -11.38   0.000     -.061505   -.0433974
       _cons |   11.08387   .3772952    29.38   0.000     10.34259    11.82514
------------------------------------------------------------------------------
```

5.6.4 Analysis of Residuals

In case you want to analyze residuals, you can save them into a variable using the `residual` option of `predict`.

```
.  predict res, residual

. sum res, det // shows the detailed descriptives (skewness and kurtosis)

                         Residuals
-------------------------------------------------------------
      Percentiles      Smallest
 1%    -.8646391      -1.058902
 5%    -.4448558      -1.010229
10%    -.2795184      -.9948394       Obs                 506
25%    -.1245533        -.92048       Sum of Wgt.         506

50%     .0212827                      Mean           2.19e-10
                       Largest       Std. Dev.      .2639513
75%     .1288318        .989933
90%     .2434836       1.128679       Variance       .0696703
95%     .3390855       1.235122       Skewness       .0511794
99%     .8526777       1.325311       Kurtosis       7.771061

. hist res, normal // look informally whether they could be normal
(bin=22, start=-1.0589021, width=.10837332)
```



In order to test the normality formally, you can either use the information matrix test shown above or you can use the Shapiro-Wilk test:

```
. swilk res

                    Shapiro-Wilk W test for normal data

    Variable |        Obs       W           V         z       Prob>z
-------------+--------------------------------------------------------
         res |        506    0.91733     28.111     8.024     0.00000
```

5.7 Presenting Regression Results

5.7.1 Storing and Recalling estimates

The command `estimates` along with its many subcommands helps you to present your regression estimates in a more efficient manner:

```
. estimates store Robust // store the heteroskedasticity-robust from above

. qui reg lprice lnox ldist rooms stratio // rerun the non-robust quietly

. estimates store nonRobust // store the non-robust
```

We can now table these estimates:

```
. estimates table nonRobust Robust // shows only coefficients

----------------------------------------
    Variable | nonRobust      Robust
-------------+--------------------------
        lnox | -.95354002   -.95354002
       ldist | -.13434015   -.13434015
       rooms |  .25452706    .25452706
     stratio | -.05245119   -.05245119
       _cons |  11.083865    11.083865
----------------------------------------
```

```
. estimates table nonRobust Robust, star // stars to show significance

-----------------------------------------
    Variable |   nonRobust        Robust
-------------+---------------------------
        lnox | -.95354002***   -.95354002***
       ldist | -.13434015**    -.13434015*
       rooms |  .25452706***    .25452706***
     stratio | -.05245119***   -.05245119***
       _cons |  11.083865***    11.083865***
-----------------------------------------
      legend: * p<0.05; ** p<0.01; *** p<0.001

. estimates table nonRobust Robust, stat(r2_a rmse) b se p // show three stats

----------------------------------------
    Variable | nonRobust      Robust
-------------+--------------------------
        lnox | -.95354002   -.95354002
             |  .11674181    .12680064
             |     0.0000       0.0000
       ldist | -.13434015   -.13434015
             |  .04310321    .05352872
             |     0.0019       0.0124
       rooms |  .25452706    .25452706
             |  .01853034    .02472044
             |     0.0000       0.0000
     stratio | -.05245119   -.05245119
             |   .0058971     .0046082
             |     0.0000       0.0000
       _cons |  11.083865    11.083865
             |  .31811155    .37729521
             |     0.0000       0.0000
-------------+--------------------------
        r2_a |  .58071114    .58071114
        rmse |  .26500291    .26500291
----------------------------------------
                         legend: b/se/p
```

Note that there is an excellent additional package `estout` that helps tremendously with presenting regression estimates. If you will be using Stata extensively, it is almost a must - its functionalities include wide ranges of presenting output as well as automatic exports to various formats including HTML or LaTeX. Type `ssc install estout` to install the package into Stata.

5.7.2 Using `margins`

The command `margins` is an extremely useful and complex command which can be used in postestimation after a wide range of models. It can be used to summarize model predictions and/or to calculate marginal effects, elasticities, etc.

Let's now load a web-based dataset specifically designed to show possibilities of `margins`:

```
. webuse margex, clear
(Artificial data for margins)

. list y outcome sex group age distance treatment in 1/5

     +-------------------------------------------------------------+
     |     y   outcome      sex   group   age   distance   treatm~t |
     |-------------------------------------------------------------|
  1. | 102.6         1   female       1    55      11.75          1 |
  2. |  77.2         0   female       1    60      30.75          1 |
  3. |  80.5         0   female       1    27      14.25          1 |
  4. |  82.5         1   female       1    60      29.25          1 |
  5. |  71.6         1   female       1    52      19.00          1 |
     +-------------------------------------------------------------+
```

Suppose we are interested in the effect of gender and group on y:

```
. reg y sex##group

      Source |       SS           df       MS      Number of obs   =     3,000
-------------+----------------------------------   F(5, 2994)      =     92.91
       Model |  186898.199         5  37379.6398   Prob > F        =    0.0000
    Residual |  1204534.81     2,994  402.316235   R-squared       =    0.1343
-------------+----------------------------------   Adj R-squared   =    0.1329
       Total |  1391433.01     2,999  463.965657   Root MSE        =    20.058

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
      female |   21.62507   1.509999    14.32   0.000     18.66433    24.58581
             |
       group |
          2  |   11.37444   1.573314     7.23   0.000     8.289552    14.45932
          3  |    21.6188   1.588487    13.61   0.000     18.50416    24.73344
             |
   sex#group |
   female#2  |  -4.851582   1.942744    -2.50   0.013     -8.66083   -1.042333
   female#3  |  -6.084875   3.004638    -2.03   0.043    -11.97624   -.1935115
             |
       _cons |    50.6107   1.367932    37.00   0.000     47.92852    53.29288
------------------------------------------------------------------------------
```

In order to calculate predictions separated by gender, we can type

```
. margins sex

Predictive margins                              Number of obs     =      3,000
Model VCE    : OLS
```

```
Expression   : Linear prediction, predict()

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
        male |   59.77145   .6454388    92.61   0.000      58.5059      61.037
      female |   78.20318   .7105463   110.06   0.000     76.80997    79.59639
------------------------------------------------------------------------------
```

We can also do it for multiple groups using the option `at`:

```
. margins sex, at(group=1 group=2 group=3)

Adjusted predictions                            Number of obs     =      3,000
Model VCE    : OLS

Expression   : Linear prediction, predict()

1._at        : group           =           1

2._at        : group           =           2

3._at        : group           =           3

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     _at#sex |
     1#male  |    50.6107   1.367932    37.00   0.000     47.92852    53.29288
   1#female  |   72.23577     .63942   112.97   0.000     70.98203    73.48952
     2#male  |   61.98514   .7772248    79.75   0.000     60.46119    63.50908
   2#female  |   78.75863   .9434406    83.48   0.000     76.90877    80.60849
     3#male  |    72.2295   .8074975    89.45   0.000     70.64619     73.8128
   3#female  |    87.7697   2.468947    35.55   0.000     82.92869     92.6107
------------------------------------------------------------------------------
```

Notice that if we run a logit model rather than a linear regression, `margins` will report probabilities of *outcome* rather than predictive means:

```
. logit outcome sex##group

Iteration 0:   log likelihood = -1366.0718
Iteration 1:   log likelihood = -1194.7466
Iteration 2:   log likelihood = -1170.6512
Iteration 3:   log likelihood = -1169.7303
Iteration 4:   log likelihood = -1169.7292
Iteration 5:   log likelihood = -1169.7292
```

```
Logistic regression                                     Number of obs     =      3,000
                                                        LR chi2(5)        =     392.69
                                                        Prob > chi2       =     0.0000
Log likelihood = -1169.7292                             Pseudo R2         =     0.1437

------------------------------------------------------------------------------
     outcome |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
      female |  -.0836901   .1605098    -0.52   0.602    -.3982835    .2309033
             |
       group |
           2 |  -2.215395   .2280855    -9.71   0.000    -2.662435   -1.768356
           3 |  -3.055726   .3067831    -9.96   0.000     -3.65701   -2.454443
             |
   sex#group |
   female#2  |   1.484466   .2679525     5.54   0.000     .9592889    2.009643
   female#3  |   .3808141   .7838848     0.49   0.627    -1.155572      1.9172
             |
       _cons |  -.7071334   .1450136    -4.88   0.000    -.9913549   -.4229119
------------------------------------------------------------------------------

. margins sex

Predictive margins                                      Number of obs     =      3,000
Model VCE    : OIM

Expression   : Pr(outcome), predict()

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
        male |   .1561738   .0132774    11.76   0.000     .1301506     .182197
      female |   .1983749   .0101546    19.54   0.000     .1784723    .2182776
------------------------------------------------------------------------------
```

The reason is that, by default, `margins` tabulates the default option of `predict`. With the option `predict()` it can, however, produce tables of other statistics `predict` can calculate (see examples 11 and 12 in the `margins` pdf help file for an example).

Besides calculating predictions, `margins` can also be used to calculate various forms of marginal effects. Suppose now we are interested in the effect of *age* and *distance* on *y*, moreover, we want to include *age* in the quadratic form as well:

```
.  reg y c.age##c.age distance

      Source |       SS           df       MS      Number of obs   =     3,000
-------------+----------------------------------   F(3, 2996)      =     37.08
       Model |  49816.4378         3  16605.4793   Prob > F        =    0.0000
```

```
    Residual |  1341616.57     2,996  447.802593   R-squared       =    0.0358
-------------+----------------------------------   Adj R-squared   =    0.0348
       Total |  1391433.01     2,999  463.965657   Root MSE        =    21.161

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   .7005506    .259336     2.70   0.007     .1920559    1.209045
             |
 c.age#c.age |  -.0126749   .0032256    -3.93   0.000    -.0189995   -.0063503
             |
    distance |  -.0067168   .0021303    -3.15   0.002    -.0108937   -.0025398
       _cons |    64.0103   4.899377    13.06   0.000     54.40381    73.61678
------------------------------------------------------------------------------
```

We can obtain the marginal effects at age 30 to 50 separated by 5 years by typing

```
.  margins, dydx(age) at(age=(30(5)50))

Average marginal effects                        Number of obs     =      3,000
Model VCE    : OLS

Expression   : Linear prediction, predict()
dy/dx w.r.t. : age

1._at        : age             =          30

2._at        : age             =          35

3._at        : age             =          40

4._at        : age             =          45

5._at        : age             =          50

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
age          |
         _at |
          1  |  -.0599439   .0719004    -0.83   0.405    -.2009231    .0810352
          2  |   -.186693   .0458827    -4.07   0.000    -.2766579   -.0967282
          3  |  -.3134421   .0334915    -9.36   0.000    -.3791108   -.2477734
          4  |  -.4401912   .0471066    -9.34   0.000    -.5325558   -.3478265
          5  |  -.5669403   .0734662    -7.72   0.000    -.7109896   -.4228909
------------------------------------------------------------------------------
```

Notes:

- The list `30(5)50` means "from 30 to 50 in steps of 5". See `help numlist` for a general treatment on numerical lists

- The first effect (the effect of age for a 30-year old) is exactly equal to

$$\frac{\left[\hat{\beta}_{age} + \hat{\beta}_{age^2}\left(31^2 - 30^2\right)\right] + \left[\hat{\beta}_{age} + \hat{\beta}_{age^2}\left(30^2 - 29^2\right)\right]}{2}$$

```
. display (       _b[age] + _b[c.age#c.age]*(31^2-30^2) + ///
                        _b[age] + _b[c.age#c.age]*(30^2-29^2) ) / 2

-.05994393
```

- It is absolutely crucial to specify all interactions (including higher powers of regressors) by factor notation if you want `margins` to produce correct results - if you do not do so, Stata will not understand that these variables are linked together and will produce incorrect results:

```
. gen age2 = age^2

.reg y age age2 distance // output omitted - same as above
. margins, dydx(age) at(age=(30(5)50)) // results are incorrect

Average marginal effects                        Number of obs     =      3,000
Model VCE    : OLS

Expression   : Linear prediction, predict()
dy/dx w.r.t. : age

1._at        : age             =          30

2._at        : age             =          35

3._at        : age             =          40

4._at        : age             =          45

5._at        : age             =          50

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
age          |
         _at |
          1  |   .7005506    .259336     2.70   0.007     .1920559    1.209045
          2  |   .7005506    .259336     2.70   0.007     .1920559    1.209045
          3  |   .7005506    .259336     2.70   0.007     .1920559    1.209045
          4  |   .7005506    .259336     2.70   0.007     .1920559    1.209045
          5  |   .7005506    .259336     2.70   0.007     .1920559    1.209045
------------------------------------------------------------------------------
```

5.8 Specifics of Time Series and Panel Data Methods

Note that the purpose of this section is not to cover details of these topics, but rather point out some specifics while approaching them in Stata rather than some other software.

5.8.1 Telling Stata about time series data

Let's first open the *sp500* example dataset:

```
. sysuse sp500, clear
(S&P 500)
```

In order for Stata to understand that we have time series data, we need to use the command `tsset`:

```
. tsset date // we specify that "date" is our time variable.
        time variable:  date, 02jan2001 to 31dec2001, but with gaps
                delta:  1 day
```

1 day.

Applying the `tsset` will allow us to run time series models. Moreover, once we have the data in Stata set as time series, we can use time operators in specifying *varlist*s. For example, if we want to recreate the *change* variable, we have two options:

```
. gen change2 = close - L.close // difference of close and its lag
(56 missing values generated)

. gen change3 = D.close // Difference of close
(56 missing values generated)

. list date close change* in 1/6 // notice the issue in 5th observation

     +-----------------------------------------------------------+
     |      date     close      change     change2     change3 |
     |-----------------------------------------------------------|
  1. | 02jan2001   1283.27           .           .           . |
  2. | 03jan2001   1347.56    64.29004    64.29004    64.29004 |
  3. | 04jan2001   1333.34   -14.22009   -14.22009   -14.22009 |
  4. | 05jan2001   1298.35   -34.98999   -34.98999   -34.98999 |
  5. | 08jan2001   1295.86    -2.48999           .           . |
     |-----------------------------------------------------------|
  6. | 09jan2001    1300.8    4.940063    4.940063    4.940063 |
     +-----------------------------------------------------------+
```

As we can see from the data, the change was calculated as missing if we have the data `tsset` with 1 day as the delta. This is because Stata will consider the weekend as a gap in the data and will therefore not know what the actual value was.

In order to go around this issue, we can create a different variable that will not contain any gaps:

```
. sort date

. gen t = _n // this will create observation number
```

```
. tsset t // notice now we there are no gaps
        time variable:  t, 1 to 248
                delta:  1 unit
```

Let's now recreate *change* again (note that I chose to use `replace` here as the variables exist from above)

```
.  replace change2 = close - L.close
(55 real changes made)

. replace change3 = D.close
(55 real changes made)

. list date close change* in 1/6 // no issue anymore

     +-------------------------------------------------------+
     |      date     close      change     change2     change3 |
     |-------------------------------------------------------|
  1. | 02jan2001   1283.27           .           .           . |
  2. | 03jan2001   1347.56    64.29004    64.29004    64.29004 |
  3. | 04jan2001   1333.34   -14.22009   -14.22009   -14.22009 |
  4. | 05jan2001   1298.35   -34.98999   -34.98999   -34.98999 |
  5. | 08jan2001   1295.86    -2.48999    -2.48999    -2.48999 |
     |-------------------------------------------------------|
  6. | 09jan2001    1300.8    4.940063    4.940063    4.940063 |
     +-------------------------------------------------------+
```

Notes:

- The above combination of keeping two time variables (one with actual and the other with a *sequential* date) allows working with business calendar data while still keeping Stata's time series functionalities - one simply applies `tsset` with the sequential date for calculations and then reapplies it with the actual date for e.g. graphics. See Baum (2007) for a more general treatment of the topic.
- TS operators (in this case `L.` and `D.`) may be combined with numbers in order to get higher order differences. Notice however that while `L2.` stands for x_{t-2}, `D2.` does **not** stand for $x - x_{t-2}$, as it represents $(x - x_{t-1}) - (x_{t-1} - x_{t-2}) = x - 2x_{t-1} + x_{t-2}$. The correct notation for the former form is `S2.`
- TS operators can also be combined with *numlist*s in order to obtain their combination. For example, `L(1/3).x` is the same as `L1.x L2.x L3.x`
- See `help tsvarlist` for more detailed description.

5.8.2 Two Examples of Time Series Methods

The time series operators explained above allow us to specify wide ranging lists of variables without actually generating them. Suppose e.g. we suspect the time series to follow an AR(5) model. We can write

```
.  reg close L(1/5).close

      Source |       SS           df       MS      Number of obs   =       243
-------------+----------------------------------   F(5, 237)       =   1484.70
       Model |  1731994.48         5  346398.897   Prob > F        =    0.0000
    Residual |  55295.0327       237  233.312374   R-squared       =    0.9691
-------------+----------------------------------   Adj R-squared   =    0.9684
       Total |  1787289.52       242  7385.49386   Root MSE        =    15.275

------------------------------------------------------------------------------
       close |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       close |
         L1. |   1.038259   .0650353    15.96   0.000      .910138     1.16638
         L2. |  -.0822642   .0934924    -0.88   0.380    -.2664465    .1019182
         L3. |   .0517224   .0933797     0.55   0.580    -.1322379    .2356827
         L4. |  -.0012996   .0908984    -0.01   0.989    -.1803717    .1777725
         L5. |  -.0256297   .0620189    -0.41   0.680    -.1478084    .0965491
             |
       _cons |   22.35052   13.77517     1.62   0.106    -4.786888    49.48793
------------------------------------------------------------------------------
```

and the model is estimated. In order to quickly analyze the results, we can type

```
.  test L1.close = 1 // does not reject

 ( 1)  L.close = 1

       F(  1,   237) =    0.35
            Prob > F =    0.5569

. testparm L(2/5).close // also does not reject

 ( 1)  L2.close = 0
 ( 2)  L3.close = 0
 ( 3)  L4.close = 0
 ( 4)  L5.close = 0

       F(  4,   237) =    0.31
            Prob > F =    0.8680
```

leading us to a conclusion that the modeled series may probably follow an AR(1) process. Suppose now we suspect that *volume* follows an ARMA(3,3) model. We can specify it as

```
.  arima volume, arima(3,0,3)

(setting optimization to BHHH)
Iteration 0:   log likelihood = -2248.9815
Iteration 1:   log likelihood =  -2246.146
Iteration 2:   log likelihood = -2243.9246
```

```
Iteration 3:   log likelihood = -2243.3201
Iteration 4:   log likelihood = -2243.1788
(switching optimization to BFGS)
Iteration 5:   log likelihood =  -2243.115
Iteration 6:   log likelihood = -2243.0645
Iteration 7:   log likelihood = -2243.0486
Iteration 8:   log likelihood = -2243.0456
Iteration 9:   log likelihood = -2243.0451
Iteration 10:  log likelihood =  -2243.045

ARIMA regression

Sample:  1 - 248                                Number of obs     =        248
                                                Wald chi2(6)      =     492.38
Log likelihood = -2243.045                      Prob > chi2       =     0.0000

------------------------------------------------------------------------------
             |                 OPG
      volume |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
volume       |
       _cons |   12293.35   473.8014    25.95   0.000     11364.72    13221.99
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L1. |   1.331802   .0881622    15.11   0.000     1.159007    1.504597
         L2. |  -1.297003   .1066762   -12.16   0.000    -1.506084   -1.087921
         L3. |   .7376227   .0684974    10.77   0.000     .6033703    .8718752
             |
          ma |
         L1. |   -.814375   .1015451    -8.02   0.000      -1.0134   -.6153503
         L2. |   .8508878   .1125454     7.56   0.000     .6303028    1.071473
         L3. |  -.2830203   .0995935    -2.84   0.004    -.4782199   -.0878206
-------------+----------------------------------------------------------------
      /sigma |     2045.8   48.16453    42.48   0.000     1951.399      2140.2
------------------------------------------------------------------------------
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
```

or as

```
.arima volume, ar(1/3) ma(1/3) // output omitted - identical as above
```

Notice however, that the second form must include the list in form of `1/3` as typing `ar(3)` will include the third autoregressive lag *only* and so on:

```
.  arima volume, ar(3) ma(3) // Includes ONLY 3rd lags

(setting optimization to BHHH)
Iteration 0:   log likelihood = -2295.5831
```

```
Iteration 1:   log likelihood = -2294.9267
Iteration 2:   log likelihood = -2292.5972
Iteration 3:   log likelihood = -2292.3021
Iteration 4:   log likelihood = -2292.0031
(switching optimization to BFGS)
Iteration 5:   log likelihood =  -2291.913
Iteration 6:   log likelihood = -2291.7989
Iteration 7:   log likelihood = -2291.7626
Iteration 8:   log likelihood = -2291.7583
Iteration 9:   log likelihood = -2291.7583

ARIMA regression

Sample:  1 - 248                                Number of obs     =        248
                                                Wald chi2(2)      =      36.99
Log likelihood = -2291.758                      Prob > chi2       =     0.0000

------------------------------------------------------------------------------
             |                 OPG
      volume |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
volume       |
       _cons |   12330.75   274.7728    44.88   0.000     11792.21     12869.3
-------------+----------------------------------------------------------------
ARMA         |
          ar |
         L3. |   .6510915   .1624308     4.01   0.000     .3327329      .96945
             |
          ma |
         L3. |  -.4409724   .1794792    -2.46   0.014    -.7927452   -.0891996
-------------+----------------------------------------------------------------
      /sigma |   2493.625   76.13095    32.75   0.000     2344.411    2642.839
------------------------------------------------------------------------------
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
```

5.8.3 Working with Panel Data

This last section of the file will briefly explain what is needed in order to work with panel data in Stata. For a more detailed discussion on panel data treatment in Stata, see `help xt`.

Firstly, notice that all Stata's panel procedures require the dataset to be in long rather than the wide form - if your dataset comes in the wide form, you first need to `reshape` it into the long form.

Secondly, in order to apply panel techniques and methods, you need to tell Stata that the dataset actually is a panel. This is done either using `tsset` or `xtset`. Let's now illustrate the concept on a fictional illustrative dataset coming from Kohler & Kreuter (2012):

```
.  use beatles, clear
(Kohler/Kreuter)

. list // long form -> we're OK
```

```
     +----------------------------+
     | persnr   time   lsat   age |
     |----------------------------|
  1. |      1   1968      8    28 |
  2. |      1   1969      6    29 |
  3. |      1   1970      5    30 |
  4. |      2   1968      5    26 |
  5. |      2   1969      2    27 |
     |----------------------------|
  6. |      2   1970      1    28 |
  7. |      3   1968      4    25 |
  8. |      3   1969      3    26 |
  9. |      3   1970      1    27 |
 10. |      4   1968      9    28 |
     |----------------------------|
 11. |      4   1969      8    29 |
 12. |      4   1970      6    30 |
     +----------------------------+
```

```
. tsset persnr time // first panelvar, then timevar
       panel variable:  persnr (strongly balanced)
        time variable:  time, 1968 to 1970
                delta:  1 unit
```

Notes:

- As long as your data contain panel as well as time variable, it does not matter whether you specify the panel characteristics using `tsset` or `xtset`, they both "understand each other".
- *panelvar* has to be a numeric variable, otherwise Stata returns an error:

```
.  tostring persnr, gen(id)
id generated as str1

. tsset id time
string variables not allowed in varlist;
id is a string variable
r(109);
```

- All of the time series operators described above work if we have a properly `tsset` data:

```
.  list persnr time L(0/2).age in 1/6, sepby(persnr)
```

```
     +---------------------------------+
     |                          L.   L2.|
     | persnr   time   age   age   age |
     |---------------------------------|
  1. |      1   1968    28     .     . |
  2. |      1   1969    29    28     . |
  3. |      1   1970    30    29    28 |
```

```
     |---------------------------------|
  4. |      2   1968    26     .     . |
  5. |      2   1969    27    26     . |
  6. |      2   1970    28    27    26 |
     +---------------------------------+
```

In order to run a fixed effect regression, we can type

```
. xtreg lsat age, fe

Fixed-effects (within) regression               Number of obs      =         12
Group variable: persnr                          Number of groups   =          4

R-sq:                                           Obs per group:
     within  = 0.9320                                         min =          3
     between = 0.8841                                         avg =        3.0
     overall = 0.1648                                         max =          3

                                                F(1,7)             =      95.92
corr(u_i, Xb)  = -0.8333                        Prob > F           =     0.0000

------------------------------------------------------------------------------
        lsat |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |     -1.625    .165921    -9.79   0.000    -2.017341   -1.232659
       _cons |   49.92708   4.606299    10.84   0.000     39.03492    60.81925
-------------+----------------------------------------------------------------
     sigma_u |  4.9227099
     sigma_e |  .46929532
         rho |  .99099353   (fraction of variance due to u_i)
------------------------------------------------------------------------------
F test that all u_i=0: F(3, 7) = 100.90                     Prob > F = 0.0000
```

References

Baum, C. (2006). *An Introduction to Modern Econometrics Using Stata*. Stata Press publication. Taylor & Francis.

Baum, C. F. (2007). Stata tip 40: Taking care of business. *Stata Journal*, 7(1), 137–139.

Kohler, U. & Kreuter, F. (2012). *Data Analysis Using Stata, Third Edition*. Taylor & Francis.