HDCA SUMMER SCHOOL ON
CAPABILITY AND MULTIDIMENSIONAL POVERTY

Topic: Introduction to STATA
Instructor: Rosaria Vega Pansini and Maria Emma Santos

Lecture Notes

This session is designed to provide a basic understanding of the statistical package called STATA. STATA is a powerful programme containing a large number of tools for statistical and econometric analysis and estimation procedures. STATA contains lots of built-in (and downloadable from the web) commands useful for data management, descriptive statistical analysis, basis regression analysis, probit/logit/tobit models, etc. It is then also extensively used in poverty and inequality unidimensional and multidimensional analysis. As an introduction, this session will provide you with some basic features of the programme useful to start any kind of analysis. In particular, this lecture will cover the following topics:

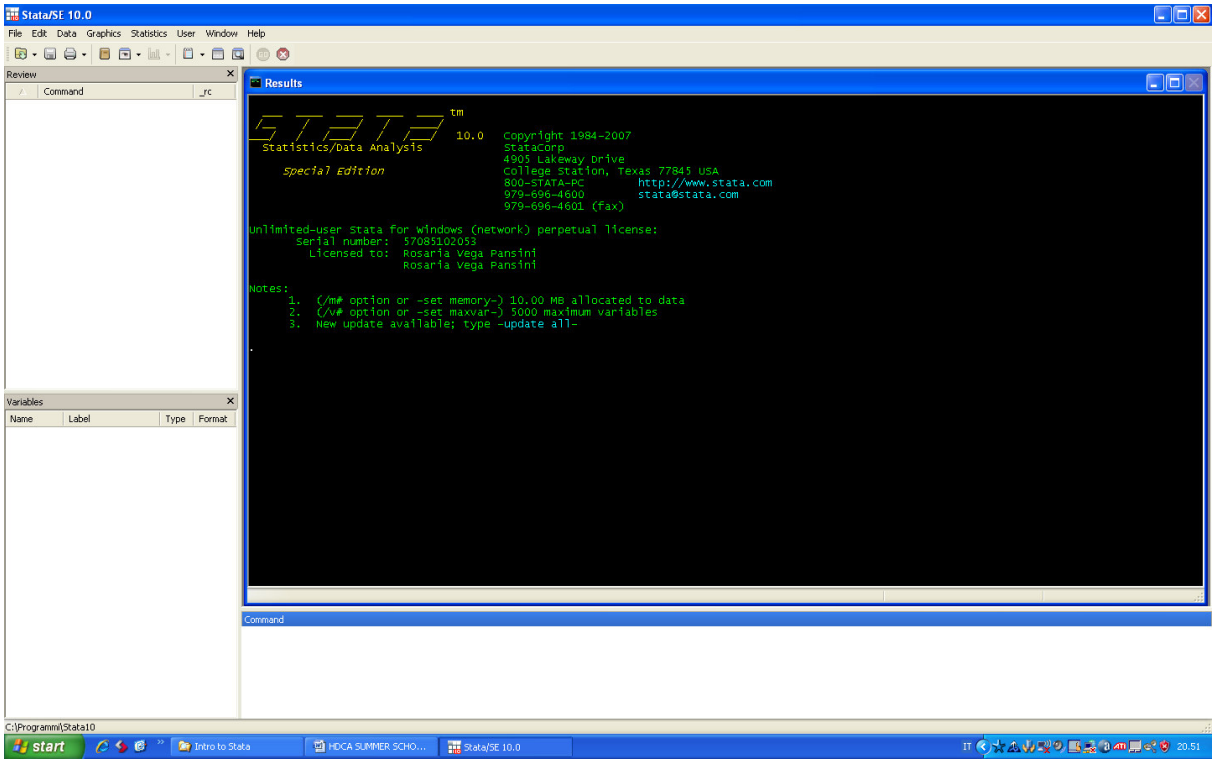
- Starting Stata
- File management: keeping track of the work in Stata
- Data management: opening, creating and saving a dataset in Stata
- Data editing and modification
- Descriptive statistics with Stata
- Regression Analysis
- Basic Matrix Programming

1. Starting Stata

When you start Stata double-clicking on the programme's icon, you will notice that Stata's interface has, in the top of the screen, different top-down menus and short-cut bottoms to various commands that Stata can perform for saving file, edit/browse data, doing statistical analysis, etc. Among them, the help menu will guide you through STATA commands and contents of the pre-installed reference manual. It will also allow you to look for updates and to directly connect to the STATA's official website. The most of the screen is occupied by the working area of the programme, which is typically made up by four windows (see figure 1 below).
These four windows correspond to four different environments, respectively:

Review Window	**Results Window**
Displays previous Stata commands entered in the current session	Displays all the commands and relative results or feedbacks (fro example, error messages)
Variables Window	**Stata Command Window**
Lists all the variables in the currently open dataset	It is where all the entered commands

Figure 1. Typical Stata's interface.



As you can see from the Figure 1, when starting Stata, in the Results Window, there are some notes giving information on the current setting of the programme. In particular, Note 1 indicate the total amount of memory allocated to the data (10.00 MB). It is preferable to start a new session with a sufficient amount of memory to perform all the operations and calculations needed in order not to loose all of your work in case the memory is full at half of the session. In order to increase the amount of memory, enter the following command:

```
set memory Xm
```

with X equal to the amount of MB allocated to the programme, which should be slightly larger than the dataset you are using and smaller than the RAM memory. Sometime before setting a new amount of memory, it's better first to clear the STATA environment with all the possible dataset previously opened by typing the following command:

```
clear    or    drop_all
```

You can also type:

```
help memory
```

if you have an error message when trying to set a new amount of memory.

Note that all Stata commands are typed in the lower case and they can be abbreviated together with the name of the variables unless there is no ambiguity, otherwise Stata will not perform the command and will return an error message. There also some keystrokes that can be used to recall some Stata commands. The most important are **Page-Up** and **Page-Down**. Pressing the **Page-Up** key it is possible to display the previous command typed in the Stata command window, while pressing the **Page-Down** key allows to display the command hat follows the currently displayed command in the Command window.

2. File management: keeping track of the work in Stata

Sometime it can be recommended to keep all the work needed for a project or a paper in the same directory. It is possible then to create a separate subdirectory, for example **HDCAsumsch** and tell Stata to save everything is done in the session in the same directory. This can be done by typing at the beginning of the session:

```
cd "c:\HDCAsumsch"
```

The subdirectory will contain all the datasets and file created and saved during each work session.

As seen before, the results window displays the last command typed and the related results produced by Stata. It always possible to scroll up the results windows to see what has been done before, but it's usually not possible to display all the commands from the beginning of the session because the contents of the window are limited. In order to keep track of the work done and in order to be able to repeat the command and the operations done during a particular session or exercise, it's necessary to use other tools offered by Stata. One possibility is to open a **log file** which keeps record of everything, commands and results, is done and appears in the results window. There are two types of log files that can be created, one only for the commands using the **cmdlog** command and one for the results, using the **log** command. The results logs can be of two formats: one a plain ascii test that can be saved using the suffix .log and the Stata Markup Control Language format (SMCL – the default format) with suffix .smcl that maintains fonts and colours of the Stata session. For switching between the two formats you can use the **translate** command. It preferable to open a new log file at the beginning of each session using the following command:

```
log using filename.log/smcl, [append replace]
```

(the commands are similar for the cmdlog command). In parenthesis there are indicated to possible options of command for creating a log file. **Append** simply add the last commands and results to an already existing and using lof file and **replace** overwrite the already existing file. When a log file is currently opened, in order to see its contents you can go on the Log windows. At the end of each session it'ìs necessary to close the log file in order to save everything it records y typing **log close**. If, otherwise, one wants to switch off and on again the log file just have too type **log off** and **log on**.
You can submit one-by-one command in the command window and see what happen in the results window. It's usually very useful to keep a track also of all the command submitted during a session and submit a bunch of them not just one at a time. Stata allows to do it using .**do files**. They are plain text files crated using any editor or the Stata's built in editor for do files (these have the usual extension assign by Stata to do files, **.do**). While using a do file, one can easily perform a certain task all together and also correct any error in the commands without going back at the beginning and redo the entire exercise. If you choose to use the Stata's do file editor, you can type:

```
doedit
```

to created a do file that can be easily saved and then executed in Stata using the following command:

```
do filename
```

An example of a do file in which you assign a work subdirectory to the current session, you open a log file at the beginning and close it at the end of the session is the following:

```
cd "c:\HDCAsumsch"
log using ex1.txt
set more off
clear
…….
…….
log close
exit
```

Note that the command **set more off** simply tells Stata to execute the commands contained in the do file without any interruption.
There are several two different ways to run commands contained in a do file. First, using the **do** command to execute the Stata commands in ex1.do and display the related output:

```
.     do e1.do
```

Second, using the **run** command or press ctrl+D key to execute the Stata commands in ex1.do, but without displaying the related output.

```
.     run ex1.do
```

If you would like to add comment in a do file, but do not want Stata to execute your notes because it will return an error message, /* */ is used:

```
/* This is an introductory session on how to use Stata program */
     cd c:\HDCAsumsch
     log using ex1.txt
```

```
set more off
clear
……..
……..
log close
exit
```

3. Data management: opening, creating and saving a dataset in Stata

To perform any analysis with Stata, it's necessary to open a Stata dataset. Before doing it, it' s better to clear out any other dataset currently in memory, typing:

```
.     clear
```

To open a Stata dataset for use, type the **use** command, eventually followed by the option **clear**.
For example, for the Buthan dataset:

```
.     use Half_sample_Buthan.dta
.     use Half_sample_Buthan.dta, clear
```

At the end of each session, to save a dataset in Stata (.dta file extension) use the **save** command specifying the mane assigned to the database.

```
.     save Buthan_Hdca.dta
```

It is possible to save a dataset anytime during your session and the continue working on the data and made changes. These changes are not saved in the memory and do not change data already stored on the pc. The **replace** option allows to save a changed file to the disk, replacing the original file.

```
.     save Buthan_Hdca.dta, replace
```

There are different method for creating a database in Stata.
The first and the easiest method when one have raw data, is using the Stata data editor, which is very similar an Excel spreadsheet. You ca access the editor by typing:

```
.      edit
```

and then enter values and press return. To change the name of the variable (originally called var1, var2, etc.), double click on the column head and change the name of the variables.
If you have data already organize n an Exel spreadsheet, it's also possible to just select, copy and paste them into the Stata editor.

A second way of creating dataset in Stata is to use **input** command, then enter your own data set in command window or do file editor.

```
input hhid sex location
10030 1 urban
10031 0 rural
10032 1 rural
end
```

The third option is to use the **insheet** command. This command is used to read data from a comma separated or tab delimited file (.csv) created by a spreadsheet (like Exel) or database program. The values in the file must be either comma or tab delimited. The names are included in the file. To use the insheet command, the original database should appear in the following way:

```
hhid,sex,location
10030,1,urban
10031,0,rural
10032,1,rural
```

With the name of the variable in the first line and the values separated by a comma. If these data are contained in a file called Bhutan.csv, in order to transform them into a Stata dataset, type:

```
.    clear
.    insheet using buthan.csv
```

If the name of variables is not included in the data, it is still possible to transform them into a Stata format using insheet command followed by the name of the variables:

```
.    insheet hhid sex location using buthan.csv
```

If, otherwise, the raw data are organized in an external ascii file, a fourth method for having a Stata dataset is to use the **infile** command. Using the ASCII file buthan.raw like the following:

```
"hhid"      10030     1    urban
"sex"       10031     0    rural
"location"  10032     1    rural
```

You have then to type the following command to load the data in Stata:

```
.    clear
.    infile hhid sex location using buthan.raw.
```

Sometimes it happens that during a session not all the variable are useful for the analysis. It can be then easy to work only with selected variables and crop the others. There are two alternative commands to do this: **keep** and **drop** command. To use the first you have to type the name of the variable that you want to keep, while with the other you indicate the name of he variable that you don't need and want to drop. Using the Buthan dataset, you can alternatively type:

```
.    keep houseid hseize poor pce_real
```

if you want to keep these three variable

```
.    drop houseid hseize poor pce_real
```

if you ant to eliminate these three variables.

Together with variables, in Stata it's also possible to preserve some observation and drop others. It can be done using again keep and drop command together with the if option and specifying the conditions to met in order to preserve only the useful information.
Using our Buthan dataset, you can keep only the observations related to the rural households (for which the variable 'area' takes the value 2):

```
.      keep if area==2
      (6508 observations deleted)
```

In the results windows, Stata returns, after the command, the number of observations that have been deleted.
Similarly you can use **drop if** command. For example, if you want to drop the observations related to urban households, we have to type:

```
.      drop if area==1
      (6508 observations deleted)
```

Also in this case, the note in parenthesis after the command, reports the number of observation dropped. Note that in this case keep and drop do the same operations reporting the same number of deleted observations.

Once you have selected the variables and observations useful for our analysis, you can arrange data into your preferred order using the **sort** command. There is no limit to the number of variables in the variable list after the sort command. Missing numeric values are interpreted as being larger than any other number, so they are placed last. For example:

```
      sort houseid pce_real
```

It can append that sometimes you have different datasets with different variables to be used in the same exercise and session. It's then necessary to create a single dataset. Depending on the design of the unique dataset that should be used, there are different commands.
If you have two files, buthan1.dta and buthan2.dta, with the same variables but different observations (buthan1 for urban and buthan2 for rural households), you can use the **append** command.

```
.     use buthan1.dta, clear
.     append using buthan2.dta
```

The **append** command does not require that the two datasets contain the same variables, even though this is typically the case. But it highly recommended to use the identical list of variables for **append** command to avoid missing values from one dataset. Any variables with different names in the two files will have missing values for the observations from the other dataset.

If the two or more datasets have the same observations with different variables, you can use the **merge** command. While the append command stick the datasets vertically, the merge command does it horizontally. Before typing the command merge, the two datasets must be sorted by the same criteria and the same variables, typically an individual or an household id code. This is why there should be at least one variables defined in the same way in both datasets and by which the observations can be matched.
The Stata procedure to merge the two files buthan1 and buthan2 if the following:

```
.     use buthan1.dta, clear
.     sort houseid
.     save buthan1.dta, replace

.     use buthan2.dta, clear
.     sort houseid
.     save buthan2.dta, replace

.     use buthan1.dta, clear
.     merge houseid using buthan2.dta
```

In the previous example, it's important to recall that the merge command requires that one datasets should be already open. This will be called the master datasets. To this file, Stata merge what is called the using dataset.
After **merge** command, a **_merge** variable appears. The _merge variable indicates, for each observation, how the merge goes. This is especially useful in identifying mismatched records. _merge can have one of three values in merging master file using using file:

_merge==1 the records contains information from master data that did not match observations form the using dataset
_merge==2 the records contains information from using data that did not match the observations in the master dataset.
_merge==3 the records contains information from both files

When there are many records, **tabulating _merge** is very useful to summarize how many mismatched observations you have. In the case, all of the records match the value for _merge is always 3.

4. Data editing and modification

Once that your datasets is complete and ready to be used in the analysis, you can use some of the command to explore the dataset. The first command used to obtain information about the data is the describe command:

```
describe
```

which provides details about the number of observations, the list of variables in the dataset, the storage type assigned to each variable, the display format, variable labels and a note reminding that the dataset has eventually changed since the last save.

You can also look at the data by typing the list command:

```
list
```

The list command can be use also to look at some variables or some observation in the dataset. In the first case you have to specify the name of the variable you want to list; in the second case, you can use the logical operators to specify for which range or subsample you want to list observations.

```
list poor pce_real
list poor pce_real if area==2
list houseid poor pce_real in 1/10
```

which gives the following Stata output that lists the variables specified in the first 10 observations.

```
     +----------------------------+
     |  houseid   poor   pce_real |
     |----------------------------|
  1. | 11010103   poor   988.6982 |
  2. | 11010103   poor   988.6982 |
  3. | 11010103   poor   988.6982 |
  4. | 11010103   poor   988.6982 |
  5. | 11010103   poor   988.6982 |
     |----------------------------|
  6. | 11010103   poor   988.6982 |
  7. | 11010103   poor   988.6982 |
  8. | 11010103   poor   988.6982 |
  9. | 11010103   poor   988.6982 |
 10. | 11010103   poor   988.6982 |
     +----------------------------+
```

In the following table, there is a list of some logical operators used in Stata:

~	not
==	equal
~=	not equal
!=	not equal
>	greater than
>=	greater than or equal
<	less than
<=	less than or equal
&	and
|	or

Note that == used in Stata after the option 'if' helps evaluating expression, while the symbol = is used to assign values. There also some options that ca be used with lots of other Stata command that help in list and obtain information about the data:

'**in'** allows to restrict the number of observations to those specified in one range;
'if' allows to restrict the sample to those cases (variable and observations) specified in the expression that follows;
'by' allows to divide the data in different groups based on different values for the by variable that it's specified,

The **browse** command is similar to **edit** , except that it will not allow users to change the data. The **browse** command is a convenient alternative to **list** command, but users can view the data through data editor instead of results window.

The **codebook** command is a tool for getting a quick overview of the variables in the dataset displaying information about variables' names, labels and values.

```
. codebook
. codebook houseid poor
```

```
houseid
-------------------------------------------------------------------------------

                  type:  numeric (long)

                 range:  [11010103,30530310]          units:  1
          unique values:  4899                     missing .:  0/24786

                  mean:   2.1e+07
             std. dev:   5.5e+06

           percentiles:        10%       25%       50%       75%       90%
                            1.3e+07   1.7e+07   2.2e+07   2.5e+07   2.8e+07

-------------------------------------------------------------
poor
poor indicator

                  type:  numeric (float)
                 label:  poorlab

                 range:  [0,1]                        units:  1
          unique values:  2                        missing .:  0/24786

             tabulation:  Freq.   Numeric  Label
                          19083         0  nonpoor
                           5703         1  poor
```

Another useful command for getting a quick overview of a data file the **inspect** command. **inspect** command displays information about the values of variables and is useful for checking data accuracy.

```
. inspect
. inspect houseid poor

houseid:                                             Number of Observations
----------                                    -------------------------------
                                                 Total   Integers   Nonintegers
|              #             Negative              -          -           -
|              #             Zero                  -          -           -
|  #       #   #             Positive          24786      24786           -
|  #   #   #   #   #                            -----      -----       -----
|  #   #   #   #   #         Total             24786      24786           -
|  #   #   #   #   #         Missing               -
+----------------------                        -----
1.10e+07       3.05e+07                        24786
(More than 99 unique values)

poor:  poor indicator                                Number of Observations
---------------------                         -------------------------------
                                                 Total   Integers   Nonintegers
|  #                         Negative              -          -           -
|  #                         Zero              19083      19083           -
|  #                         Positive           5703       5703           -
|  #                                           -----      -----       -----
|  #                         Total             24786      24786           -
|  #   #                     Missing               -
+----------------------                        -----
0                      1                       24786
   (2 unique values)
   poor is labeled and all values are documented in the label.
```

Finally, **count** command can be used to show the number of observations that satisfying if options. If no conditions are specified, count displays the number of observations in the data.

```
. count
24786

. count if area==2
 18278
```

In order to perform your descriptive and statistical analysis, Stata allows to modify the dataset creating new variables using different criteria and options. There are three main ways of creating variables. The first uses the **generate** command:

```
. [by varlist:] gen newvar= exp [if exp][in range]
```

where varlist specifies the criteria and the order by which the new variable will be crated. `exp` indicates the expression and if and range specify the eventual subsample for which to create the new variable. Using the buthan datatset, you can crate for example a new variable which is equal to the squared value of the household size in the following way.

```
by houseid: gen hsize2= hsize^2
```

or alternatively:

```
by houseid: gen hsize2= hsize*hsize
```

For existing variables, the **replace** command is can be used to replace existing value of a variable with a new value.

```
. replace hsize2= hsize^2
```

The command **egen** stands for extended generate and is a powerful command with many options for creating new variables. It adds summary statistics to each observation. As an extension of generate, the command egen can create new variables using not only expression but also functions, as in the following example:

```
egen hhincome=sum(pce_real), by(houseid)
```

Here the function sum allows to create a new variable hhincome which is derived as the cumulative sum of the pce_real individual income.

A third method to create a new variable is using the **collapse** command that transform data to a more aggregate level of observation. If, for example, you have an individual-level dataset and want to create a household level dataset, you can type:

```
. collapse (fcn1) varlist1 by (varlist)
```

Where fcn1 is a function generating a summary statistic (mean, max, min, sum) and varlist1 is the list of variables to which the fcn1 apply. Example:

```
. collapse (mean)  pce_real, by(houseid)
```

5. Descriptive statistics with Stata

In order to have an overview of the means, standard deviation, max and min values of the variables in he dataset, you can use the **summarize** command, with or without specifying the variables for which you want to obtain the descriptive statistics.

```
. summarize
. summarize houseid poor pce_real area
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     houseid |     24786    2.08e+07     5464314   1.10e+07   3.05e+07
        poor |     24786    .2300896    .4208985          0          1
    pce_real |     24786    2309.655     2037.38   297.9957   74659.33
        area |     24786    1.737432    .4400382          1          2
```

The option **detail** provide some additional detailed summary statistics.

```
. su poor, detail
```

```
                        poor indicator
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%            0              0
10%            0              0       Obs                24786
25%            0              0       Sum of Wgt.        24786

50%            0                      Mean           .2300896
                        Largest       Std. Dev.      .4208985
75%            0              1
90%            1              1       Variance       .1771555
95%            1              1       Skewness        1.28257
99%            1              1       Kurtosis       2.644986
```

The options **by** and **if** are also allowed in order to get the summary statistics for a subsample of the data after having sorted first the data.

```
. sort area
. by area: summarize pce_real
```

```
-------------------------------------------------------------------------------
-> area = urban

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    pce_real |      6508    3869.433    2748.501   476.8156   74659.33

-------------------------------------------------------------
-> area = rural

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    pce_real |     18278    1754.285    1328.432   297.9957   21988.93
```

Sometimes it can be time consuming deriving the summary statistics for a categorical variable, i.e a variable taking different value. Another concise way, which does not require the data to be sorted, is by using the **summarize()** option as part of the tabulate command.

```
. tab  b11q2, sum(pce_real)

           |     Summary of Per Capita Real
relationshi |   Consumption (Nu per person per
   p to the |                month)
       head |        Mean   Std. Dev.       Freq.
------------+------------------------------------
  self(head |   2748.4123    2625.614        4899
  wife/husb |   2516.6097   2092.6725        3751
  son/daugh |   2205.3024     1807.28       10434
  father/mo |   2072.5247   1650.4377         510
  sister/br |   2627.8678   2349.1063         728
  grandchil |   1445.6804   982.42383        2016
  niece/nep |   2537.2337   2081.3024         693
  son-in-la |   1514.5365   1159.7204         722
  brother-i |   2560.0665   2058.6757         256
  father-in |   2453.1584   2031.8844         183
  other fam |   2060.2038   1702.6889         318
  live-in-s |   3731.4912   2031.8828         129
  other non |   2418.5256   1667.7917         147
------------+------------------------------------
      Total |   2309.6547   2037.3801       24786
```

Another useful command to obtain frequency tables is the **tabulate** command.

```
tab  area

urban/rural |      Freq.     Percent        Cum.
------------+-----------------------------------
      urban |      6,508       26.26       26.26
      rural |     18,278       73.74      100.00
------------+-----------------------------------
      Total |     24,786      100.00
```

It's also possible to create crosstables using tabulate. In the following example use tab to create a cross table to see the frequencies of poor people by area of location.

```
tab  area poor

urban/rura |    poor indicator
         l |   nonpoor       poor |     Total
-----------+----------------------+----------
     urban |     6,413         95 |     6,508
     rural |    12,670      5,608 |    18,278
-----------+----------------------+----------
     Total |    19,083      5,703 |    24,786
```

You can use the **plot** option to make a plot to visually show the tabulated values.

```
. tabulate poor, plot
```

```
      poor |
 indicator |      Freq.
-----------+-----------+-----------------------------------------------------
   nonpoor |     19,083 |**************************************************
      poor |      5,703 |***************
-----------+-----------+-----------------------------------------------------
     Total |     24,786
```

6. Regression analysis

In Stata it is possible to estimate regression models using the regress command :

```
. regress depvar varlist
```

If the dependent variable is depvar and varlist is the list of independent variables. For example, using our dataset on buthan, we can estimate a regression model with per capita real consumption as dependent variable and b11q1 (sex), b11q2 (relationship to the househld head) and hsize as independent variables.

```
reg  pce_real b11q1 b11q2 hsize
```

```
      Source |       SS       df       MS              Number of obs =   24786
-------------+------------------------------           F(  3, 24782) = 1010.56
       Model |  1.1214e+10     3  3.7380e+09           Prob > F      =  0.0000
    Residual |  9.1667e+10 24782  3698917.37           R-squared     =  0.1090
-------------+------------------------------           Adj R-squared =  0.1089
       Total |  1.0288e+11 24785   4150917.7           Root MSE      =  1923.3

------------------------------------------------------------------------------
    pce_real |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       b11q1 |   45.61553   24.45171     1.87   0.062    -2.311288    93.54235
       b11q2 |  -2.550175   5.435696    -0.47   0.639    -13.20446    8.104113
       hsize |  -265.5406   5.012639   -52.97   0.000    -275.3657   -255.7156
       _cons |   3881.013   49.77576    77.97   0.000     3783.449    3978.576
------------------------------------------------------------------------------
```

You can run the regression with robust standard errors, when the residuals are not iid. This is very useful when there is heterogeneity of variance. The **robust** option does not affect the estimates of the regression coefficients but only the value of standard errors.

```
reg  pce_real b11q1 b11q2 hsize, robust

Linear regression                               Number of obs =   24786
                                                F(  3, 24782) =  760.87
                                                Prob > F      =  0.0000
                                                R-squared     =  0.1090
                                                Root MSE      =  1923.3

------------------------------------------------------------------------------
             |               Robust
    pce_real |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       b11q1 |   45.61553    24.3869     1.87   0.061    -2.184257    93.41532
       b11q2 |  -2.550175   5.081302    -0.50   0.616    -12.50983     7.40948
       hsize |  -265.5406   5.657117   -46.94   0.000    -276.6289   -254.4523
       _cons |   3881.013   61.74175    62.86   0.000     3759.995    4002.03
------------------------------------------------------------------------------
```

In the regression analysis it's also possible to leave the constant from the regression analysis using the **nocons** option.

```
reg  pce_real b11q1 b11q2 hsize, nocons
```

Finally, as for other Stata command, also for regression analysis, it's possible to estimate regression models over different subsample using the option 'if'.

```
reg  pce_real b11q1 b11q2 hsize if sex==1
```

Stata stores results from estimation commands in e(), and you can see a list of what exactly is stored using the **ereturn list** command.

```
. ereturn list

scalars:
                  e(N) =  24786
               e(df_m) =  3
               e(df_r) =  24782
                  e(F) =  760.8709069645669
                 e(r2) =  .1089995225620176
               e(rmse) =  1923.256969858369
                e(mss) =  11213924854.48303
                e(rss) =  91666570315.60019
               e(r2_a) =  .1088916619602779
                 e(ll) =  -222593.9745525338
              e(ll_0) =  -224024.2545945871

macros:
           e(cmdline) : "regress pce_real b11q1 b11q2 hsize, robust"
             e(title) : "Linear regression"
               e(vce) : "robust"
            e(depvar) : "pce_real"
               e(cmd) : "regress"
        e(properties) : "b V"
           e(predict) : "regres_p"
             e(model) : "ols"
         e(estat_cmd) : "regress_estat"
           e(vcetype) : "Robust"
```

```
matrices:
                  e(b) :  1 x 4
                  e(V) :  4 x 4

functions:
             e(sample)
```

Using the **generate** command, we can extract those results, such as estimated coefficients and standard errors, to be used in other Stata commands.

```
reg  pce_real b11q1 b11q2 hsize.
gen intercept=_b[_cons]
. display intercept
. gen slope=_b[hsize]
. display slope
```

The **estimates table** command displays a table with coefficients and statistics for one or more estimation sets in parallel columns. In addition, standard errors, t statistics, p-values, and scalar statistics may be listed by **b, se, t, p** options.

```
. estimates table, b se t p

----------------------------
    Variable |   active
-------------+--------------
       b11q1 |  45.615532
             |  24.386904
             |       1.87
             |     0.0614
       b11q2 | -2.5501753
             |  5.0813019
             |      -0.50
             |     0.6158
       hsize | -265.54063
             |  5.6571174
             |     -46.94
             |     0.0000
       _cons |  3881.0126
             |  61.741745
             |      62.86
             |     0.0000
----------------------------
           legend: b/se/t/p
```

Together with simple linear regression analysis, Stata can perform lots of regression models like those applied when the dependent variable is dichotomous. **Logit** and **probit** commands allow infact performing regression analysis when the distribution of the dependent dichotomous variable is either logistic or normal.
Both command are very similar to the regress command.

```
logit poor  area  hsize b11q1

Iteration 0:   log likelihood = -13369.185
Iteration 1:   log likelihood = -11021.409
Iteration 2:   log likelihood = -10682.657
Iteration 3:   log likelihood = -10618.269
Iteration 4:   log likelihood = -10611.562
Iteration 5:   log likelihood = -10611.423
Iteration 6:   log likelihood = -10611.422

Logistic regression                                     Number of obs   =      24786
                                                        LR chi2(3)      =    5515.53
                                                        Prob > chi2     =     0.0000
Log likelihood = -10611.422                             Pseudo R2       =     0.2063

------------------------------------------------------------------------------
        poor |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        area |   3.215123   .1060674    30.31   0.000     3.007235    3.423011
       hsize |   .3099366   .0070043    44.25   0.000     .2962086    .3236647
       b11q1 |  -.0611571   .0338054    -1.81   0.070    -.1274145    .0051004
       _cons |  -9.244637   .2225047   -41.55   0.000    -9.680738   -8.808535
------------------------------------------------------------------------------
```

Or alternatively,

```
Probit poor  area  hsize b11q1
```

Stata reports the maximum likelihood estimates for the original coefficient in both probit and logit commands. The **logistic** command by default produces the output in odds ratios but can display the coefficients if the **coef** options is used.

```
. logistic  poor  area  hsize b11q1

Logistic regression                                     Number of obs   =      24786
                                                        LR chi2(3)      =    5515.53
                                                        Prob > chi2     =     0.0000
Log likelihood = -10611.422                             Pseudo R2       =     0.2063

------------------------------------------------------------------------------
        poor | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        area |   24.90635   2.641751    30.31   0.000     20.23137     30.6616
       hsize |   1.363339   .0095492    44.25   0.000     1.344751    1.382184
       b11q1 |   .9406755      .0318    -1.81   0.070     .8803687    1.005113
------------------------------------------------------------------------------

. logistic  poor  area  hsize b11q1, coef
```

In the same way, **dprobit** command estimates a Probit model, but reports the change in the probability for an infinitesimal change in each independent variable instead of the coefficient fro each independent variable.

7. Basic Programming

In the previous session we have seen that after the estimation command like regress, Stata saves matrices with the parameters estimated and with the variance-covariance matrices. These can be created, used and manipulated.

If you define a matrix called beta as e(b) and another called covb as e(V), you can create them and then list them.

```
reg  pce_real b11q1 b11q2 hsize
```

```
      Source |       SS       df       MS              Number of obs =   24786
-------------+------------------------------           F(  3, 24782) = 1010.56
       Model |  1.1214e+10     3  3.7380e+09           Prob > F      =  0.0000
    Residual |  9.1667e+10 24782  3698917.37           R-squared     =  0.1090
-------------+------------------------------           Adj R-squared =  0.1089
       Total |  1.0288e+11 24785   4150917.7           Root MSE      =  1923.3

------------------------------------------------------------------------------
    pce_real |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       b11q1 |   45.61553   24.45171     1.87   0.062    -2.311288    93.54235
       b11q2 |  -2.550175   5.435696    -0.47   0.639    -13.20446    8.104113
       hsize |  -265.5406   5.012639   -52.97   0.000    -275.3657   -255.7156
       _cons |   3881.013   49.77576    77.97   0.000     3783.449    3978.576
------------------------------------------------------------------------------
```

```
. matrix beta =e(b)

. matrix covb = e(V)

. matrix list beta
```

```
beta[1,4]
         b11q1       b11q2       hsize       _cons
y1   45.615532  -2.5501753  -265.54063   3881.0126
```

```
matrix list covb
```

```
symmetric covb[4,4]
            b11q1       b11q2       hsize       _cons
b11q1    597.88633
b11q2   -4.5573915    29.546789
hsize    1.3114836   -7.147535    25.126555
_cons   -895.64546  -48.375006  -132.38563   2477.6262
```

These matrices can then be stored and used for successive statistical and econometric analysis.

Suggested Basic textbook on this topic:

Getting Started with Stata for Windows. STATA Press.
User's Guide. STATA Press.

Suggested Readings/Resources on this topic:

There are a lot of useful resources on 'how to use Stata' that you can access online. In general, in the website, http://www.stata.com/links/resources.html you can find links to other online websites where to find tutorials on Stata. Moreover, Stata's Frequently Asked Questions (FAQs) section of the website http://www.stata.com/support/faqs and the email discussion group with contributes and answer from StataCorp staff available at http://www.stata.com/support/statalist can be also very useful.

In particular, very good online resources are: the UCLA Academic Technology Services available at http://www.ats.ucla.edu/stat/stata and the Prof. Cameron's webpage at the University of Davis, available at http://cameron.econ.ucdavis.edu/**stata**/**stata**.html