# Introduction to Stata

Arnaud Chevalier- Centre for the Economics of Education- London School of Economics
Version 1.0- September 2001;

**-What this course is about?**
This course aims at taking you from stata absolute beginner to stata programmer in 5 lectures. The first two lectures are about basic commands and data manipulation. Lecture 3 reveals the basic of programming, and the use of macro. Proper programming starts in lecture 4, where the secret of branching and looping would be revealed. Lecture 5 deals with the issue of writing more advance commands and inputting instructions to a stata program. I'm not expecting you to remember everything at the first read, so I have compiled a set of lecture notes, which I hope you will find useful. Don't forget, that the stata help files are well done and that the manuals are an invaluable source of information.
This course is not about econometrics, so most of the examples are not econometric related (or nothing more advanced than OLS). If you are interested in Econometric issues, have a look at the stata manuals and on-line help files, and find out, how estimates are programmed.

**-Plan of the course**
-lecture 1: the basic, and how to organise your data
-lecture 2: advance data manipulation
-lecture 3: programming: basic
-lecture 4: programming: branching and looping
-lecture 5: programming: parsing

In this set of notes, all stata output will appear in `courier font` while the rest of the text should be in TNR. *Comments will also appear sporadically in italic.*

# Introduction to Stata- A. chevalier

## Lecture 1: The basics

Content of Lecture 1:

-Use of directories
-Stata and the net: upgrades, search
-Commands everybody should know
-Organising your data

A- Use of Directories

Stata executable files are scattered in different folders on your computer. You can find out where they are by typing:

```
sysdir
   STATA:  C:\Stata7\
 UPDATES:  C:\Stata7\ado\updates\
    BASE:  C:\Stata7\ado\base\
    SITE:  C:\Stata7\ado\site\
 STBPLUS:  c:\ado\stbplus\
PERSONAL:  c:\ado\personal\
OLDPLACE:  c:\ado\
```

Later, we are going to discuss what is in these folders, but for the moment, the only thing you need to know is that you should not work from any of this directory to prevent you for inadvertently deleting a file. Also, stata updates could erase some of your work.

To find out, which directory you are in when stata fires up:

```
. pwd
C:\Stata7
```

Which is the official stata directory. I organise my computer by projects, so I have on for this course on my D drive

```
. cd D:\teaching
D:\teaching
```

I can also create a directory, and change to it.

```
. mkdir stata1
. cd stata1
D:\teaching\stata1
. pwd
D:\teaching\stata1
```

The advantage of working from within a non-stata directory is not only that stata and your work are safe, you can now use files without spelling out the full path (which can be quite handy).

Stata comes with various “tuition” datasets, we are going to use one of them, to check how stata conceptualise data.

```
. use "C:\stata\auto"
(1978 Automobile Data)
```

B] Stata and the net

Stata files are regularly updated. Do not confuse updates with a new version of stata, which also come with some regularity, but do not share the best quality of updates: they are free. Updates are easy if you have permanent access to the web, otherwise have a look at stata website.

Stata is composed of an executable file and official ado files. You should never modify those yourself. However, regularly, stata corp finds mistakes in these codes or way to make things fasters, user friendly,….and updates these files.

**Making sure you are running the most up to date stata is the first thing to check, please do so regularly.**

This is extremely simple, and you only need to follow the instruction on the screen, stata does everything else for you….

```
. update

Stata executable
    folder:               C:\Stata7\
    name of file:         wstata.exe
    currently installed:  08 Aug 2001

Ado-file updates
    folder:               C:\Stata7\ado\updates\
    names of files:       (various)
    currently installed:  14 Aug 2001

Recommendation
    Type -update query- to compare these dates with what is available
from
    http://www.stata.com.
------------------------------------------------------------------------
----------
```

My executable was modified by stata corp on 8 Aug 2001, while the latest set of ado files date from the 14th of Aug.
To find out whether more recent updates exist, you need to check stata.com which update query does for you.

```
. update query
(contacting http://www.stata.com)

Stata executable
    folder:               C:\Stata7\
    name of file:         wstata.exe
    currently installed:  08 Aug 2001
    latest available:     08 Aug 2001

Ado-file updates
    folder:               C:\Stata7\ado\updates\
    names of files:       (various)
    currently installed:  14 Aug 2001
    latest available:     06 Sep 2001
```

```
Recommendation
    Type -update ado-
```

My executable is Ok, but my ado files are out of date for some of them, update ado will get me the latest.
In the case where my executable is also out of date, stata will advise me to update all

```
update ado
(contacting http://www.stata.com)
Ado-file update log
    1.  verifying C:\Stata7\ado\updates\ is writeable
    2.  obtaining list of files to be updated
    3.  downloading relevant files to temporary area
[ output omitted]
    4.  examining files
    5.  installing files
    6.  setting last date updated

Updates successfully installed.

Recommendation
    See help whatsnew to learn about the new features
```

In the case you have to upgrade your executable file, more work is needed from you. After downloading the new executable,
Go to explorer, find where stata.exe is (should be in C:\stata\stata.exe)
Change stata.exe to stata.old (you don't want to delete the executable just yet, in case of problem you can always come back to it, rename it stata.exe, so that you would not have to re-install stata
Change stata.bin to stata.exe

If you were not previously convinced by the benefit of updating, you should also know that it keeps you up to date with the search facilities.

Search allow you to find
- stata command that you do not know (ado file)
- ado files published in the stata technical bulletin (those are user ado files)

In version 7, you can also get ado files from the Harvard archive (more, newer ado files). If you use windows, just click on the hyperlink to install the file of interest.
However, in version 6, after finding the name of the file of interest you can use webseek to download it straight to the right spot.

For example;

```
webseek psmatch
```

If you have to copy the file yourself, it should go into C:\ado\personal\p
Remember those are not official stata files, which is why you put them in the ado directory.

You can keep track of all the users ado files that you have added to your package over time. Ado will list all of them, with information on where you got it from and what it does.

```
.  ado

[1] package sg97 from http://www.stata.com/stb/stb46
      STB-46  sg97.    Formatting  regression  output  for  published
tables.

[2] package heckman2 from http://fmwww.bc.edu/RePEc/bocode/h
      'HECKMAN2': module to estimate Heckman selection model using
Heckman's tw
> o-step approach

[3] package dm80_1 from http://www.stata.com/stb/stb57
      STB-57 dm80.1.  Update to changing numeric variables to string

[4] package rowsort from http://fmwww.bc.edu/RePEc/bocode/r
      'ROWSORT': module to row sort a set of integer variables
…
[26] package gr33_1 from http://www.stata.com/stb/stb61
      STB-61 gr33_1.  Violin plots for Stata 6 and 7
```

you can also delete these ado files:

```
ado uninstall gr33_1
```

To summarise, the websites of interest are:

* Stata corp “frequently asked questions” http://www.stata.com/support/faqs/ should answer most of your queries.

*Before re-inventing the wheel, have a look at the archive: http://ideas.uqam.ca/ideas/data/bocbocode.html or through webseek or findit commands.

* If you are still stuck, have a go at the stata list. You can register from http://www.stata.com/support/statalist/. Users are usually friendly enough and will point you in the right direction. Archive from the list can be searched at: http://groups.yahoo.com/group/statalist

C] Commands every body should know.

Stata works like a word processor. What you see on your screen is only a copy of the original file. What ever you do to the data, will not affect the original copy. The only way to change that original file and make your changes permanent is to save the data.

Stata commands all work with the same syntax. Stata allows you to specify options to the command after comma

The command to load a dataset (assuming your dataset is in stata format otherwise see at the end of this lecture) is: use
A useful option with use is clear, which mean that you allowing stata to clear the memory (previous data set) in order to load the new one.

```
. use "C:\stata\auto",clear
(1978 Automobile Data)
```

The command to save it is: save
Save as also some options, - replace to overwrite on top of a previous copy of your data - old: to save your data in a previous version of stata (from 7 to 6 for example). This is especially useful when a new version of stata is available and not everybody has access to it yet.

```
. save auto,replace
(note: file auto.dta not found)
file auto.dta saved
```

*Comments:* Since I'm working from a safe directory, I do not need to specify the path while saving my data.
As auto did not exist in D:\teaching\stata1, stata keeps me informed that he could not find a file to overwrite.

To have a look at the data;

```
. browse
```
or

```
. edit
- preserve
```
I would not recommend the latter if you are not planning to make any changes to the data. –preserve indicates that stata is making a copy of the file as it is at the moment, and that the file will be altered only if you save these changes.

A better way to look at your data is to use;

```
. describe

Contains data from auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          11 Sep 2001 18:35
 size:         3,478 (99.5% of memory free)
------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
make            str18  %-18s                  Make and Model
price           int    %8.0gc                 Price
mpg             int    %8.0g                  Mileage (mpg)
rep78           int    %8.0g                  Repair Record 1978
hdroom          float  %6.1f                  Headroom (in.)
trunk           int    %8.0g                  Trunk space (cu. ft.)
weight          int    %8.0gc                 Weight (lbs.)
length          int    %8.0g                  Length (in.)
turn            int    %8.0g                  Turn Circle (ft.)
displ           int    %8.0g                  Displacement (cu. in.)
gratio          float  %6.2f                  Gear Ratio
foreign         byte   %8.0g       origin     Car type
------------------------------------------------------------------------------
Sorted by:  foreign
```

This reports the number of observation, number of variables, the size of your data set and the type of storage of your data. You can also describe a list of variables rather than all of them.

```
. describe make

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
make            str18  %-18s                  Make and Model
```

You can also organise your data set with various commands;
First, I will reduce my dataset to make it more manageable. Keep and drop are self-explanatory names…
If executes the command if the condition is satisfied. If can be combined with most but not all stata commands.

```
. keep if _n<=10
(64 observations deleted)
. drop  rep78 hdroom trunk length-foreign
```

I only kept the first 10 observations and 4 variables:

*Comment:_n refers to the row where your data is store. We will come back to its use at a later stage.*

*The – sign mean all the variables (columns) between the column length and foreign. This syntax can be used for all stata commands where a list of variables is needed.*

I can also sort my data ;

```
. sort mpg
```

*Comment: It is possible to sort on a list of variables, then stata in the order the variables appear in the list. Sort, only sort data in ascending order. It is possible to sort data in descending order but this is quite unusual.*

It is also possible to look at your data without using browse or edit (at least in small data sets).

```
. list

     make                  price       mpg    weight
  1. Linc. Continental     11,497        12     4,840
  2. Linc. Mark V          13,594        12     4,720
  3. Linc. Versailles      13,466        14     3,830
  4. Merc. XR-7             6,303        14     4,130
  5. Cad. Deville          11,385        14     4,330
  6. Peugeot 604           12,990        14     3,420
  7. Cad. Eldorado         14,500        14     3,900
  8. Merc. Cougar           5,379        14     4,060
  9. Buick Electra          7,827        15     4,080
 10. Merc. Marquis          6,165        15     3,720
```

When working with larger data sets, you may want to have a look at only some observations. You can do that using in

```
. list in 3/5

     make                  price       mpg    weight
  3. Linc. Versailles      13,466        14     3,830
  4. Merc. XR-7             6,303        14     4,130
  5. Cad. Deville          11,385        14     4,330
```

Or you may want to execute a command for a subset of the variable (as with if), but this time the restriction is terms of observation numbers . In can be combined with a large number of stata commands. I usually prefer using if.

```
. drop in 3/5
(3 observations deleted)
```

looking and checking at your data in this visual way in probably OK for small data set, but becomes tedious and less accurate when the number of variables or observations increase. A useful command for this kind of work is assert.

Assert check that the statement made is true, if not it stops.

This does not seem much, but prove to be a great time saver.

To demonstrate its use, let's pretend that a mistake had been made in the auto file.

```
. recode foreign *=3 if make=="Toyota Corolla"
(1 changes made)

. assert foreign==0 |foreign==1
1 contradiction in 74 observations
assertion is false
r(9);
```

knowing that there is a mistake in foreign, we can list or tabulate that variable to find the observation that causes the problem. Doing this exploratory work prevents you realising that there was something at a rather advance stage of your analysis…

Another set of useful commands for data manipulations are: generate, replace, recode.

**The most useful command are ……help, search** and if you have an on line version of stata **webseek or findit** (you need to download the ado file for the later one)

So I recommend that you spend some time looking at the following commands;

Use, save, append, merge, compress
Descrive, codebook, list, browse, count, inspect, summarize, table, tabulate
Generate, replace, egen, rename, drop, keep, sort, encode, reshape

Finally, a list of command to keep track of your work:

You can add notes to your data set, for example to remind you where you got the data from.

```
. note: TS obtained from official stata directory
```

*Comments: TS will automatically include the date and time the note was included*

Notes will be display when you `describe` the data or by typing `notes`. You can have as many notes as you want, and drop some of them or all of them.

```
. notes

_dta:
  1.  12 Sep 2001 16:00 obtained from official stata directory
  2.  12 Sep 2001 16:05 blabla
```

```
. note drop _dta in 2
(1 note dropped)

. notes

_dta:
  1.  12 Sep 2001 16:00 obtained from official stata directory
. note drop _dta _all
(1 note dropped)
```

Stata 6 and onward allows you to scroll back your screen, however, a safer option to keep record of your work is to open a log file at the beginning of your session, that you will close at the end. (stata 7 will also add the date and time at opening and closing).

```
. log using stata1
-------------------------------------------------------------------------------
       log:  C:\Stata7\stata1.smcl
  log type:  smcl
 opened on:  12 Sep 2001, 16:09:01

. log close
       log:  C:\Stata7\stata1.smcl
  log type:  smcl
 closed on:  12 Sep 2001, 16:09:12
-------------------------------------------------------------------------------
```

The log command allows the replace, append options.

D] Organising your data

We have already seen the describe command you can also get more information by using codebook. Codebook without a varlist, will describe all variables in the dataset.

```
. codebook make mpg

make ---------------------------------------------------------- Make
and Model
                  type:  string (str18), but longest is str17

         unique values:  74                   coded missing:  0 / 74

              examples:  "Cad. Deville"
                         "Dodge Magnum"
                         "Merc. XR-7"
                         "Pont. Catalina"

               warning:  variable has embedded blanks

mpg ----------------------------------------------------------
Mileage (mpg)
                  type:  numeric (int)

                 range:  [12,41]                     units:  1
         unique values:  21                   coded missing:  0 / 74

                  mean:   21.2973
             std. dev:    5.7855

          percentiles:        10%       25%       50%       75%       90%
                              14        18        20        25        29
```

**Storage, format**

There are two types of data in stata numeric and string. Numeric will store numbers while string will typically but not exclusively store text.
String can be up to 80 characters long (str80), while there are various types of numeric variables, differing by storage and precision. Stata like any computer program stores number in a binary format (what you see is not what is in memory). Binary number tends to take a lot of space, so when stata "recognises" that precision is not an issue, it will try to store the data in the format that is the most compact. Those are;

| | | | |
|---|---|---|---|
| Byte | integer between | -127 and | 126 |
| Int | integer between | -32,767 and | 32,766 |
| Long | integer between | –2,147,483,647 and | 2,147,483,646 |
| Float | real with roughly 8 digits of accuracy | | |
| Double | real with roughly 16 digits of accuracy | | |

Things to remember about internal storage;
-The default in stata is to store variables as float.
-An accuracy of 8 digits is usually enough for most of our work.
-Some numbers cannot be reproduced in binary format

identification numbers can be sensitive to the rounding and approximation. It is therefore desirable to store them as string (even so they are numbers). This should not affect you (when was the last time you did calculations or a regression on the id) but will guarantee that individual 1234567890 does not get mixed up with 1234567891. With strings, what you see is what you get.

**Important example**
We will now demonstrate this point and at the same time have a look at the infile command which allows you to import a dataset that is not in stata format. First have a look at the help file for infile.

Say that we have a file containing an id number, a name and a variable x

Begin xfile.raw

| | | |
|---|---|---|
| 392938483 | Bilal | 5.7 |
| 113482938 | Frank | 11.5 |
| 443251024 | Rabbit | 7.8 |

end xfile.raw

Cut and paste the data into a text editor (notepad) and save it as a raw file in your stata directory.

Infile allows you to use an ASCII file and create names as well as specifying the format of each variable. In this case, the name is obviously a string, and the other two variables are going to be saved in the default format (float).

```
. clear

. infile id str14 name x using xfile
(3 observations read)

. list

          id            name          x
  1.  3.93e+08           Bilal        5.7
  2.  1.13e+08           Frank       11.5
  3.  4.43e+08          Rabbit        7.8
```

Everything seems to have worked Ok, and the data have been transferred to stata, and could be saved as a stata file now. The display format of id can be changed to make it more readable.

```
. format id %12.0g

. list

             id            name          x
  1.    392938496           Bilal        5.7
  2.    113482936           Frank       11.5
  3.    443251008          Rabbit        7.8
```

However, if you check id and the original number, they do not matched. This is because, float is accurate up to 8 digit and our id number was 9 digit long, it suffers of approximation when converting. This is highly undesirable with an id number- if we had had another individual with id 392938484, it will also have ended up coded as 392938496….
The solution to that is to change the format of id when making the transfer from the raw file to stata. You can increase the precision to real or double, depending on the length of your id, but what I prefer is to enter the id as a string, since anyway, we are not going to do any calculation on the id.

So here is what we should have typed:

```
. clear

. infile str9 id str14 name x using xfile
(3 observations read)

. list

            id            name          x
  1. 392938483           Bilal        5.7
  2. 113482938           Frank       11.5
  3. 443251024          Rabbit        7.8
```

Which is exactly what we want

You can also alter the “display” format of integer (this does not affect their storage format)

```
. describe mpg

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
----------
mpg             int    %8.0g                  Mileage (mpg)
```

So here we find that mpg will be displayed with 8 digits (including an eventual minus sign). So imagine that we had a value like 1,250,000,000 it will appear as 1.3e+09.

It is possible to change this default format. So it is possible to increase the number of digit in front and after the comma.

```
. format mpg %8.2f

. list make mpg in 1/5

     make                      mpg
  1. Merc. Cougar            14.00
  2. Subaru                  35.00
  3. Dodge Diplomat          18.00
  4. Cad. Eldorado           14.00
  5. Pont. Le Mans           19.00
```

the f means fixed that is your results will line-up.

Browsing, through the auto data, foreign appear to be a string: the values are domestic or foreign. However, looking at our describe output, foreign is reported to be an integer with a label

In our example using infile, we were in a special case were the ASCII raw data was in a nice format (all variables were separated), but this is not always the case. When the data comes in a long uninterrupted string of characters, you need to write a dictionary. (see lecture 3)

**Label**

Labels help you remembering what the codes 0 and 1 stand for:

```
foreign -------------------------------------------------------------
- Car type
                  type:  numeric (byte)
                 label:  origin

                 range:  [0,1]                        units:  1
         unique values:  2                    coded missing:  0 / 74

            tabulation:  Freq.   Numeric  Label
                            52         0  Domestic
                            22         1  Foreign
```

you can see that the label of foreign is called origin and that origin maps 0 with domestic and 1 with foreign
We will now see how to create labels, to do so we have first to get rid of the original "origin" label.

```
label drop origin
```

you can also list all the labels in memory (label dir) , or list a specific label.

```
. label dir
origin

. label list origin
origin:
           0 Domestic
           1 Foreign
```

2 steps to create labels:

```
. label define origin 0 "Domestic" 1"Foreign"

. label values foreign origin
```

In the first step, we define the label by a name and then specify the mapping of integer and names, in the second we map the label to a variable. The same label can be used for more than one variable.

Some stata command have the option to temporary break the connection between a variable and its label. For example, compare:

```
. tabulate foreign

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00

. tabulate foreign,nolabel

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         52       70.27       70.27
          1 |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

In addition, variable can have labels (those you see in the variable window) and so do datasets.
Those are defined as follow;

```
label var make "make and model"
```

You can see the change on your variable window (no more capital letter)

Similarly, you can label the dataset (but I don't find that extremely useful)

```
. label data "1978 car data"

. describe

Contains data from D:\Teaching\stata1\auto.dta
  obs:            74                          1978 car data
 vars:            13                          11 Sep 2001 18:35
 size:         3,774 (99.9% of memory free)
```

[output omitted]

you may also want to add some information associated with a particular variable (for example a reference to where you got it from)

```
. note rep78: data from the us association of car mechanics, 1978

. notes

rep78:
  1.  data from the us association of car mechanics, 1978
```

Exercise:  The variable rep78 has no label so far and frankly I'm not always going to remember that 1 is poor, 2 fair, 3 average, 4 good and 5 excellent.

Create this label and save the new and improved version of the dataset.