

Introduction to STATA

Lina Cardona
University of Essex
October 2010

STATA is a flexible statistical package that provides important tools for data analysis. You can work in STATA in two ways, either using the menus of the toolbar or typing the commands. Most of the time, typing the commands is easier and faster. Once you start working out through the commands you will get a basic knowledge of the package.

CONTENTS

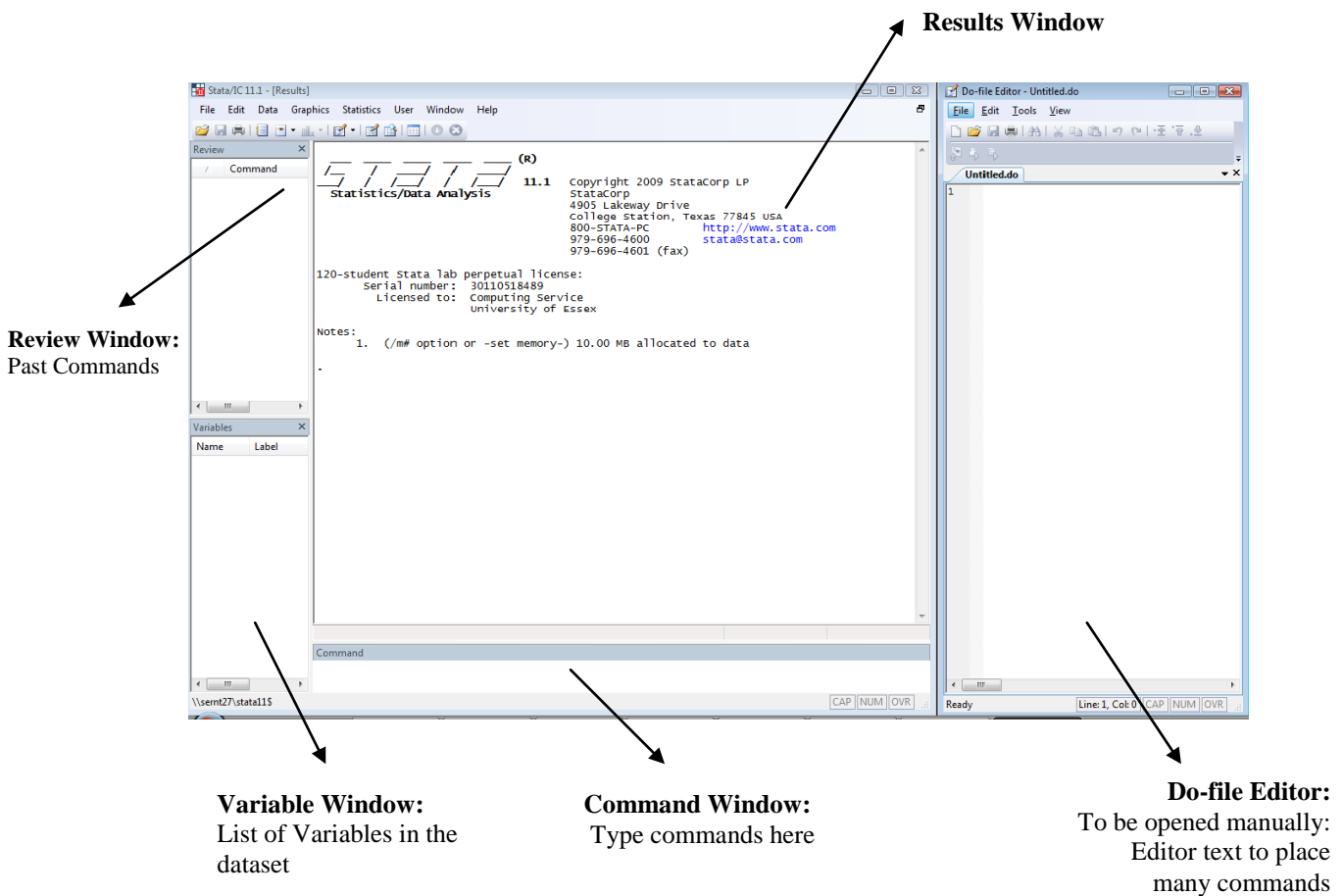
1. Starting STATA and Structure	2
2. Creating a Do-file	8
3. Setting Memory Parameters	8
4. Entering/Opening Data	9
4.1 Datasets in STATA format: ending in .dta	9
4.2 Datasets in other formats: ending in xls, txt, csv, raw	10
4.3 Entering the data manually	12
4.4 Copying and Pasting from and Spreadsheet files	13
5. Command Syntax	13
6. Starting with the Data	14
6.1 Listing, Browsing and Editing Data	14
6.2 Describing Data	16
6.3 Tabulating Data	17
6.4 Descriptive Statistics	23
6.5 Creating and Manipulating Variables	24
6.6 Labeling variables	26
6.7 Working with String Variables	27
6.8 Graphs	28
7. Joining two different datasets	30
7.1 Appending Data	30
7.2 Merging Data	32
8. Saving Output	35
9. Basic Econometric Analysis	36
10. Starting with Time Series	38
11. Summary	39
12. Useful sources for STATA Support	41
References	41

1. Starting STATA and Structure

- To start STATA: Select STATA from the Windows Start menu.
- To exit STATA: select File/Exit from the STATA's menus: **exit**

STATA Windows:

Upon opening STATA, you will see four different windows in the screen of your computer as it is shown below: the main screen or results window, the review window, the variable lists window and the command window. Please notice the do file Editor is an extra-window that need to be opened manually as it will be shown later.

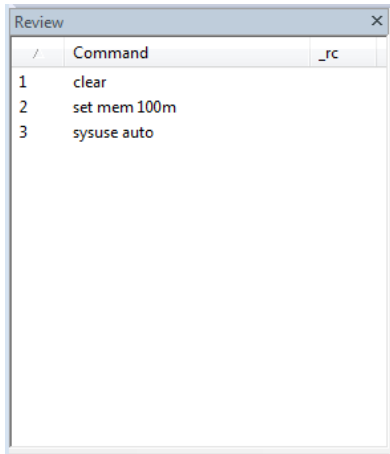


Command Window



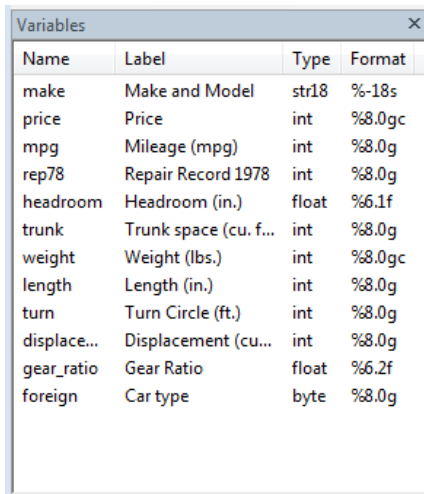
STATA commands are entered in the command window. It supports text editing: copying, pasting, a command history with page up (backward command history), page down (forward command history).

Review Window



This shows the history of commands that have been entered. It is possible to copy a past command if you click once to the past commands. If you want to execute them you can double click them.

Variables Window



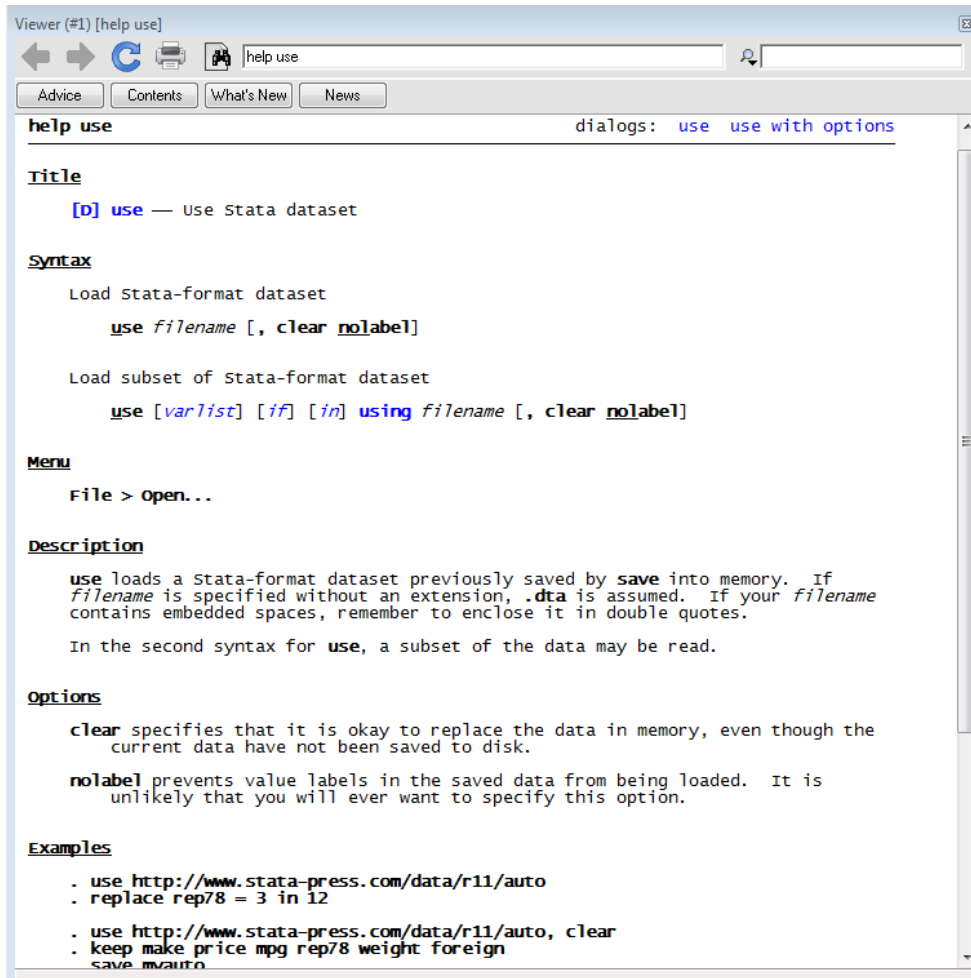
The screenshot shows a window titled "Variables" with a close button (X) in the top right corner. The window contains a list of variables in a table format:

Name	Label	Type	Format
make	Make and Model	str18	%-18s
price	Price	int	%8.0gc
mpg	Mileage (mpg)	int	%8.0g
rep78	Repair Record 1978	int	%8.0g
headroom	Headroom (in.)	float	%6.1f
trunk	Trunk space (cu. f...	int	%8.0g
weight	Weight (lbs.)	int	%8.0gc
length	Length (in.)	int	%8.0g
turn	Turn Circle (ft.)	int	%8.0g
displace...	Displacement (cu...	int	%8.0g
gear_ratio	Gear Ratio	float	%6.2f
foreign	Car type	byte	%8.0g

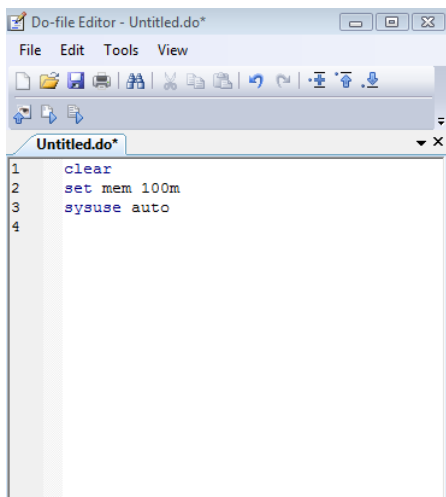
Shows the list of variables contained in the dataset. When you select a variable (by clicking once), the name of it appear into the command window.

Viewer Window (using the help viewer as an example)

The viewer window is not shown when STATA opens it is normally used to view help documents. It also allows you to print logs (saved results), add new commands from internet or install official updates to STATA. You can see it following the Menu Window->Viewer->New Viewer or you can select it through the STATA toolbar.



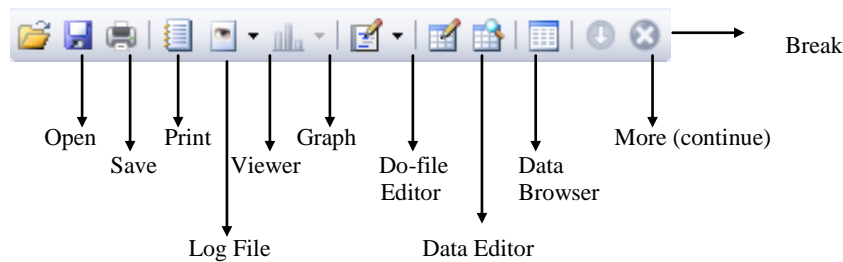
Do-file Editor



A do-file is a text file that is used to type and save all the commands you want to execute in STATA. You will be able to reproduce your work later without having to type each command again (by using the command window you will be unable to replicate your work later on). **You are recommended to start learning to use the do file editor immediately**

STATA 11 Toolbar

You don't need to memorize the tool bar icons. You can easily pass slowly your mouse through each icon and the name of each of them will appear.



- open:** to open a dataset in STATA version (.dta extension)
save: to save a dataset.
print: to print the contents of the window you have open.
log: to start, stop, pause or resume a log file.
viewer: to open a viewer window or to bring it into the front (i.e. graphs window or help window that has been already opened)
results: to open the results window, or bring it to the front.
graph: to open the graph window, or bring it to the front.
do-file editor: to open the do-file editor, or bring it to the front.
data editor: to open the data editor, or bring it to the front.
data browser: to open the data browser, or bring it to the front.
more: to continue with the output when long results are paused in the screen.
break: to interrupt the current task, ignoring the command was being executed.

Going further with more STATA tools

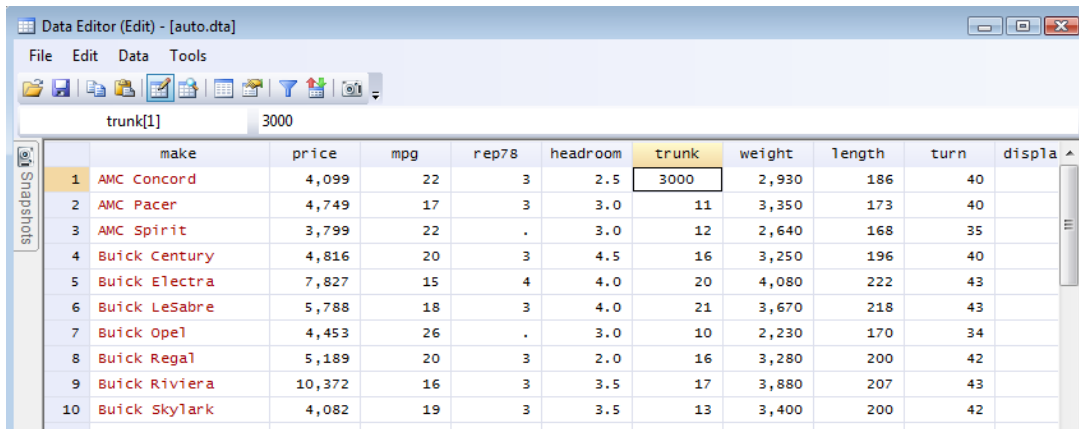
Using the Data Browser and the Data Editor.

Data Browser

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displacement	gear_ratio	foreign
2	AMC Pacer	4,749	17	3	3.0	11	3,350	173	40	258	2.53	Domestic
3	AMC Spirit	3,799	22	.	3.0	12	2,640	168	35	121	3.08	Domestic
4	Buick Century	4,816	20	3	4.5	16	3,250	196	40	196	2.93	Domestic
5	Buick Electra	7,827	15	4	4.0	20	4,080	222	43	350	2.41	Domestic
6	Buick LeSabre	5,788	18	3	4.0	21	3,670	218	43	231	2.73	Domestic
7	Buick Opel	4,453	26	.	3.0	10	2,230	170	34	304	2.87	Domestic
8	Buick Regal	5,189	20	3	2.0	16	3,280	200	42	196	2.93	Domestic
9	Buick Riviera	10,372	16	3	3.5	17	3,880	207	43	231	2.93	Domestic
10	Buick Skylark	4,082	19	3	3.5	13	3,400	200	42	231	3.08	Domestic
11	Cad. Deville	11,385	14	3	4.0	20	4,330	221	44	425	2.28	Domestic
12	Cad. Eldorado	14,500	14	2	3.5	16	3,900	204	43	350	2.19	Domestic
13	Cad. Seville	15,906	21	3	3.0	13	4,290	204	45	350	2.24	Domestic
14	Chev. Chevette	3,299	29	3	2.5	9	2,110	163	34	231	2.93	Domestic
15	Chev. Impala	5,705	16	4	4.0	20	3,690	212	43	250	2.56	Domestic

Stata allows you to have an overview of the data in spreadsheet form without modifying the information. In addition to the tool bar menu you can also access to the spreadsheet by typing **browse** or **br**. Browsing data is very useful when you are creating variables and want to check how they have been allocated across the different observations.

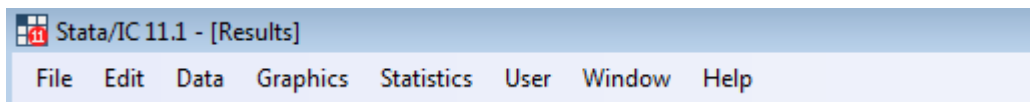
Data Editor



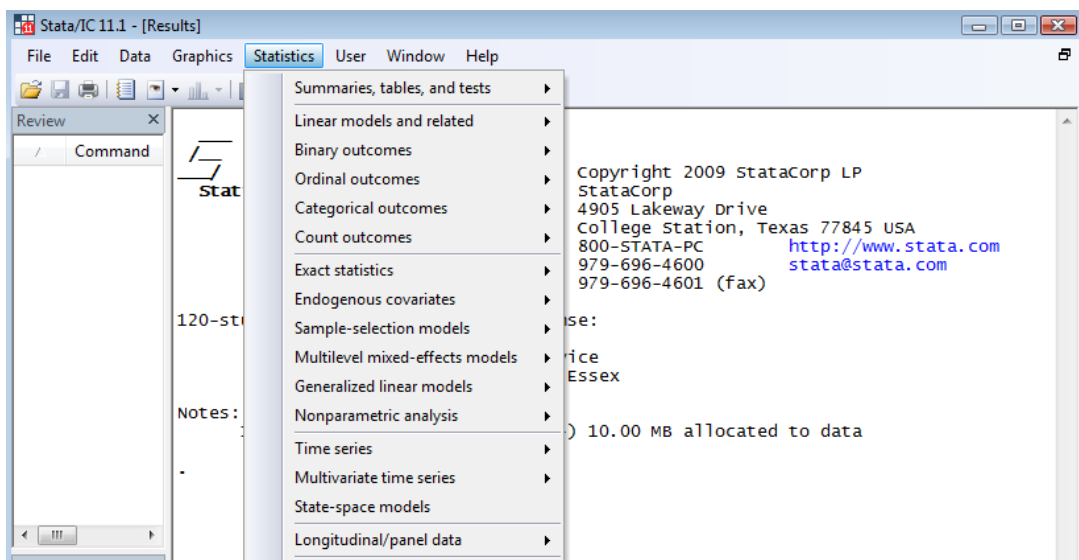
	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displa
1	AMC Concord	4,099	22	3	2.5	3000	2,930	186	40	
2	AMC Pacer	4,749	17	3	3.0	11	3,350	173	40	
3	AMC Spirit	3,799	22	.	3.0	12	2,640	168	35	
4	Buick Century	4,816	20	3	4.5	16	3,250	196	40	
5	Buick Electra	7,827	15	4	4.0	20	4,080	222	43	
6	Buick LeSabre	5,788	18	3	4.0	21	3,670	218	43	
7	Buick Opel	4,453	26	.	3.0	10	2,230	170	34	
8	Buick Regal	5,189	20	3	2.0	16	3,280	200	42	
9	Buick Riviera	10,372	16	3	3.5	17	3,880	207	43	
10	Buick skylark	4,082	19	3	3.5	13	3,400	200	42	

The Editor also presents the data in spreadsheet form but also allows the data to be edited.

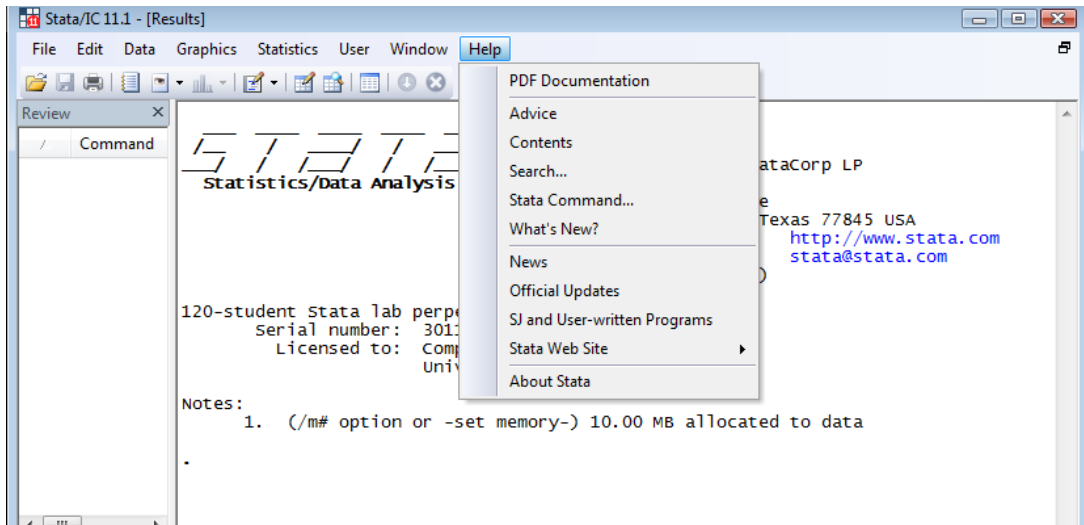
Header Bar



At the top of the screen you can find the following menus: File, Edit, Graphics, Statistics, User, Window, and Help. In all of them the functions and commands available in STATA are included. All the commands you execute using the menus also can be done by typing them manually in the command window or the do-file Editor.



Help

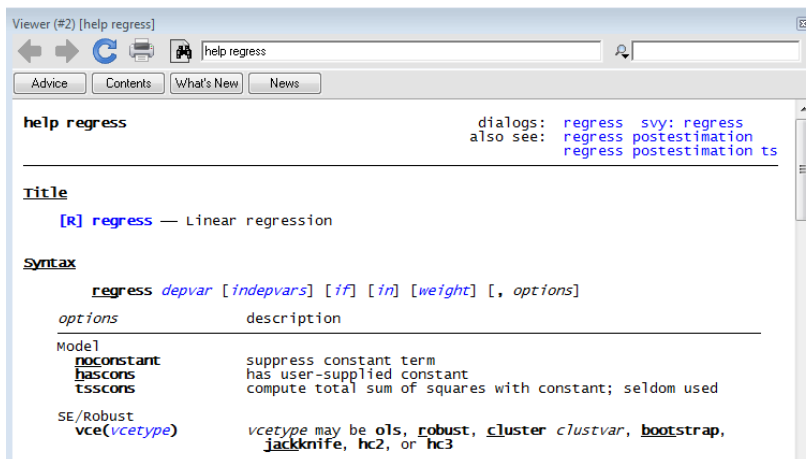


You can see the help table of contents (going through **help** from the menu bar and then **contents**) or you can search for help entries on a topic.

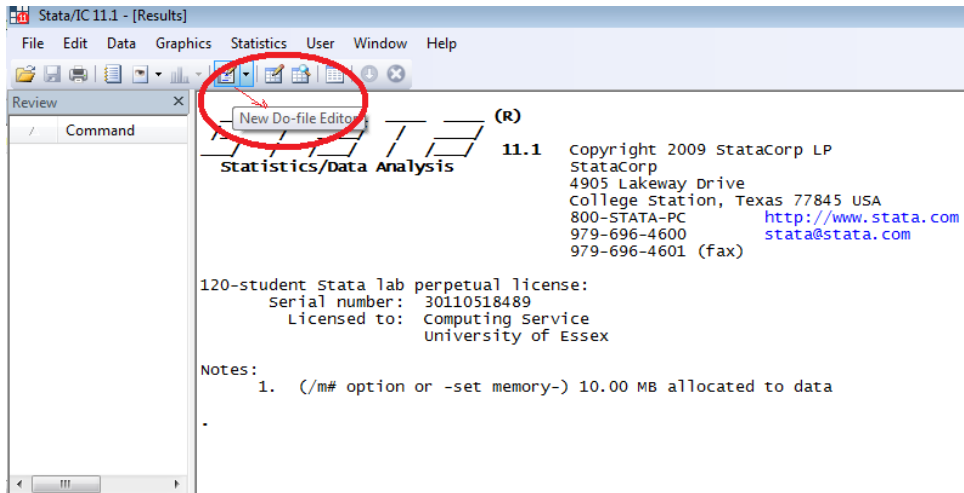
Alternatively you can use the help typing the command: to have more information about a command when you already know its name, type **help command-name**; for particular topics you can type **help contents topic**.

If you want to know the kind of commands and functions available in STATA to conduct linear regression you can type **help contents regression**.

If you want to know how the command regress works, type **help regress** to see the following help page:



2. Creating a Do-file



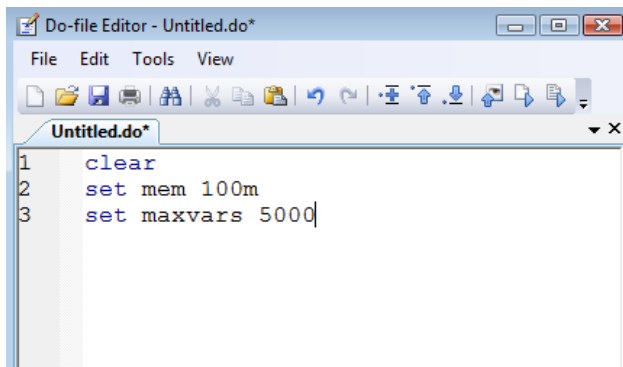
You are strongly recommended to store your commands in a do file. This allows you to replicate your work later, or recreate datasets. You just need to click **Do-file Editor** in the tool bar and save it with a name of your preference.

3. Setting Memory Parameters

STATA starts with a default memory of 1MB which could be enough for small datasets. However, for large datasets you will need to allocate additional to hold all the data you need. To do so type `set mem` and the corresponding amount of memory you would like to allocate (see below). STATA also counts with additional commands to increase the number of variables allowed, matrices and observations:

<code>set mem 100m</code>	set memory to 100 megabytes
<code>set maxvar 5000</code>	set maximum number of variables to 5000
<code>set matsize 500</code>	allows 600 explanatory variables
<code>set obs 100</code>	set number of current observations to 100

Clearing any information or data previously opened (by using `clear`) and allocating the memory (`set mem`) are the first things you need to do before opening a dataset:

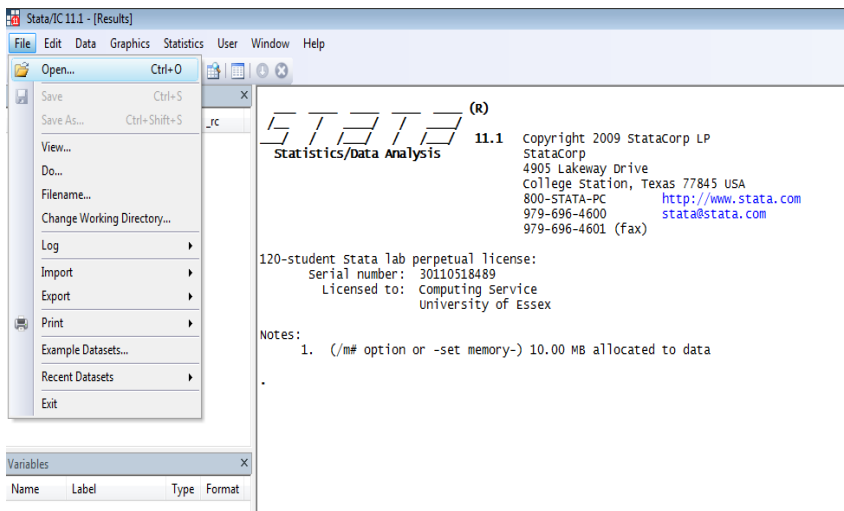


4. Entering/Opening Data

Before opening a dataset you need to identify the extension of the file containing the information. While many datasets are already in STATA format (i.e. the filename ends in .dta) many others may be available in spreadsheet format, text or comma separated values. It is important to be aware of this to use the appropriate command to open the dataset. For short datasets you can also enter the data by hand.

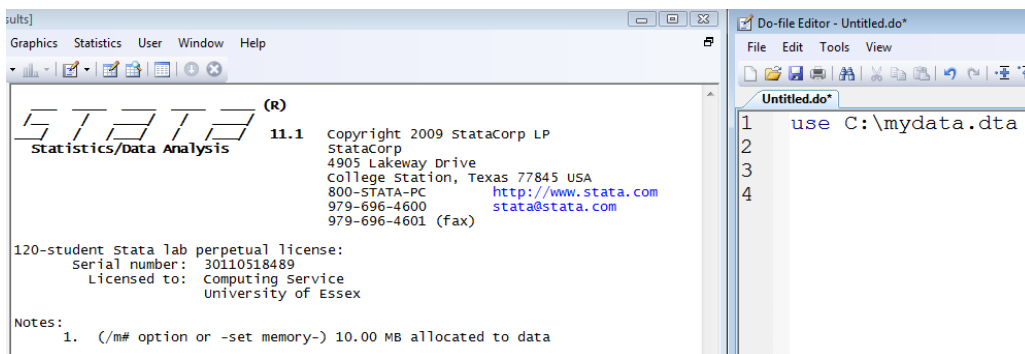
4.1 Datasets in STATA format: ending in .dta

Use the command **open** from the file menu and search for your dataset location.



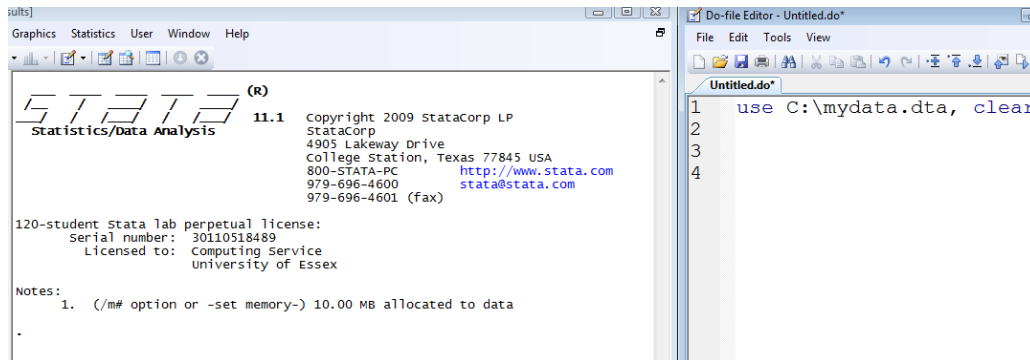
Alternatively, type the command **use** and the name of the dataset into the command window or into the do-file editor:

```
use C:\mydata.dta
```



You may also consider typing the command for opening the dataset by adding the option **clear** as follows to discard any other dataset that may be opened

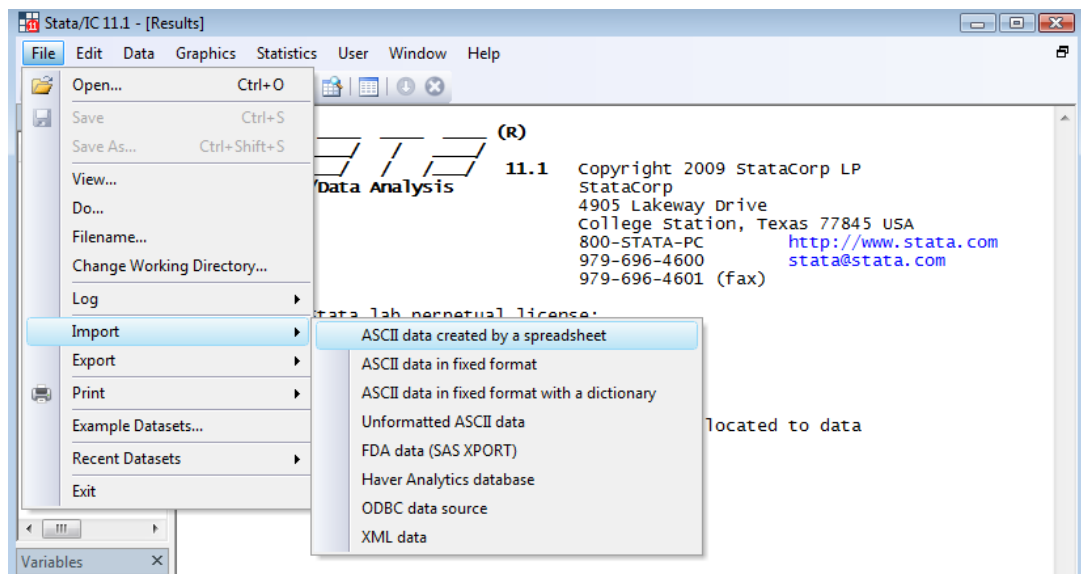
```
use C:\mydata.dta, clear
```



4.2 Datasets in other formats: ending in xls, txt, csv, raw

The command **insheet using** open a dataset created in a spreadsheet (in the case of excel files it opens the first sheet). This command also can read files determined by commas or tab characters. If the file contains spaces, put double quotes around the filename.

You can also open datasets by clicking on File >Import>, selecting the relevant data, and browsing to the dataset in your computer:



Alternatively you can type the command by hand and the name of the dataset as follows, which could be in some cases easy to understand:

insheet using C:\mydata.csv

To open a command delimited file. You can save the excel files as “comma separated values”, i.e. as a file ending with .csv

insheet using C:\mydata.txt delimiter (“;”) To open a text file where data points are separated by semicolons

insheet using C:\mydata.raw

To open the file using the default extension.

Example 1:
The following file **mycars.raw** contains the following lines that correspond to the variables make, price and MPG of a few cars. The variables names are not contained in the files but they are separated by commas.

```
Olds 98,8814,21,4060
Chev.Monza, 3667, ,2750,
Datsun 510,5079,24
```

Now, to read the data:

```
cd C:\
insheet using mycars
```

infile

To open a dataset where the data is separated by spaces and are numerical values. For instance to input three variables named a, b, c respectively you may type:

infile a b c using mydata .raw is the default extension.

Example 2:
The following file **mycars2.raw** contains the following lines that correspond to the variables make, price and MPG of a few cars. The variables names are not contained in the files and the first line is formatted different to the other lines. Notice also that the value for price is absent for Chev.Monza.

```
"Olds 98"           8814  21    4060
"Chev.Monza"       3667  .     2750
"Datsun"           510
5079
24
```

Now to upload the data:

```
infile str18 make price mpg using mycars2
save mycars2.dta
```

Additional useful commands to start working with STATA

cd c:\

to change the directory to the c:\ drive. Once specified the default path you just need to write down the name of the data set (notice that to open the dataset you do not need to specify the directory again since by executing

previously the command `cd c:\` you have already done it): `use mydata.dta`

`clear`

to drop all the dataset in memory.

`exit`

to exit STATA.

`save c:\mydata.dta`

to save the dataset “mydata”

`save c:\mydata.dta, replace`

to save the dataset “mydata”, replacing the old version with the new one.

4.3 Entering data manually

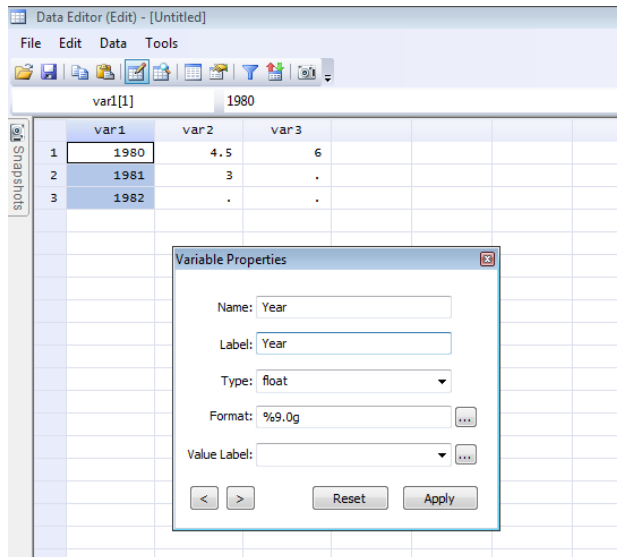
First, open the **Data Editor**. Click on the data editor button or typing `edit` into the command window. Then, enter the data: columns correspond to variables and rows to observations. Use the tab key to move from one cell to the next one. Use the enter key to move between values in the same column.

STATA 11 adds a period (‘.’) in the cells without data given that you have added in the previous rows some data:

The screenshot shows the STATA Data Editor window titled "Data Editor (Edit) - [Untitled]". The window has a menu bar with "File", "Edit", "Data", and "Tools". Below the menu bar is a toolbar with various icons. The main area displays a data grid with columns labeled "var1", "var2", and "var3". The first three rows are populated with data: Row 1 has values 1980, 4.5, and 6; Row 2 has values 1981, 3, and .; Row 3 has values 1982, ., and . with a cursor in the cell containing the period. A "Snapshots" panel is visible on the left side of the window.

	var1	var2	var3
1	1980	4.5	6
2	1981	3	.
3	1982	.	.

To rename the variable, double click the column and when the **Variable Properties dialog** appears type the new variable name and click OK. You can also use copy and paste to change the order of the columns or rows.



4.4 Copying and Pasting from and Spreadsheet files

For short datasets available in spreadsheet formats such as excel files, you can also copy the data and pasting into the data editor (by using copy/paste). However, you must be very careful when doing this since in some cases the data format, decimals or thousands are changed once you paste them into the editor leading to errors in your dataset. To guarantee a correct loading of your data it is better to use the command **insheet using** or you can also convert your datasets into STATA format by using additional software like **StatTransfer**¹

5. Command Syntax

Most of the commands follow the same syntax: the name of the command, the list of variables to work with the command and optional arguments after a semicolon.

command [*variables names*] [, options]

In STATA help arguments that are part of the syntax are written in *italics* while optional arguments (that may not be needed to be typed are in [brackets]).

By understanding the syntax you can take advantage not only of all the different options and functions that can be executed with a particular command but also to understand the **help** window when searching for new ones.

¹ <http://www.stattransfer.com/>

Example 3:
 The syntax for the command `browse` is the following.

```
browse [varlist] [if] [in] [, nolabel]
```

and you type the following:

```
browse year
```

In the above example, **browse** replaces the word *command* in the syntax and **year** corresponds to the *[varlist]* statement that followed it. As you can see you **do not need to type brackets** when writing the name of the variable.

Notice also that **[varlist]** is in brackets which mean that for this command in particular, typing the name of a variable is optional and without doing it, the command still works which is not always the case. Thus, by typing **browse** without specifying the variable, the command will be executed over the whole dataset, showing all the information in spreadsheet form:

The data can be also restricted by specifying a logical condition that is true or false. This can be made using **if** and some operators (`=`, `<`, `>`, `>=`, `<=`; negation: `!=`, `~`). We can also restrict the dataset by specifying the rank of the observations using **in** (e.g. in 1/5: the first five observations of the data). The following syntax will include such restrictions:

```
command [varlist] [if] [in] [, options]
```

if is a very useful function to restrict variables over a particular population or restrict commands to be executed only if certain conditions applied. In addition, it is a command that is also allowed by most of the commands.

STATA identify the first three letters of the name of the variable, then, the commands can be abbreviated writing down the first three letters of the command name followed by the name of the variable.

Thus, in the example above where the variable *year* is viewed in spreadsheet form, it would be enough by typing:

```
br year
```

STATA commands and variable names are **case sensitive**. Then, if I have a variable called **Year** is not the same as having **year**.

6. Starting with the Data

6.1 Listing, Browsing and Editing Data

Listing

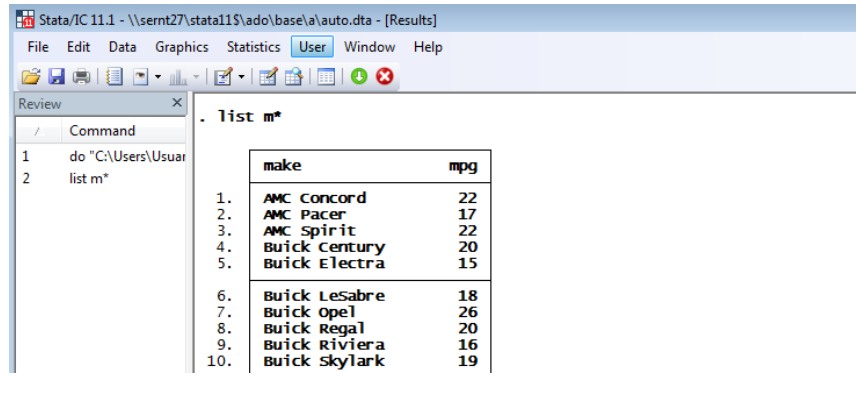
list list the *whole dataset* in the Result Window.
l The abbreviate command for **list**

list varname list one or more variables in the Result Window

Example 4:

You may want to list just the variables starting with m, then you use the command and then indicates the variable name as m* which implies all the variables starting with m.

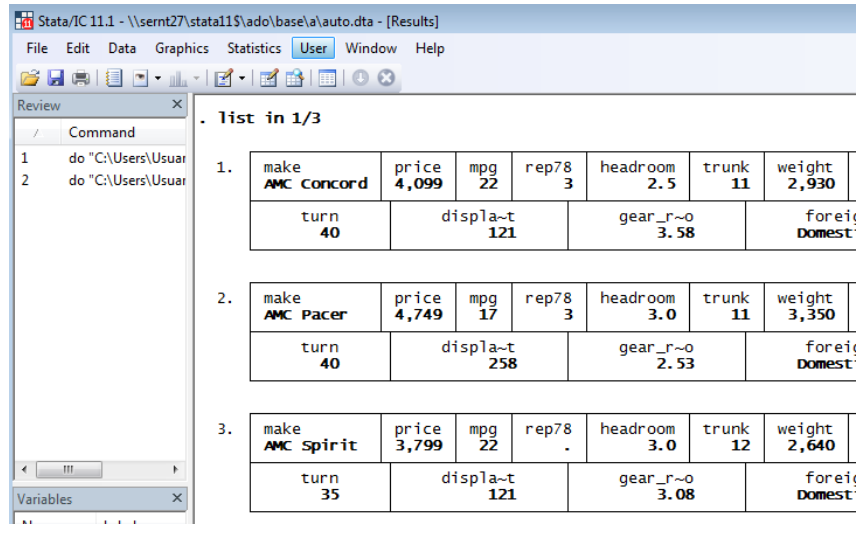
list m*



Example 5:

You want to list the first 3 observations of the data set as we show here:

list in 1/3.



Browsing and Editing

The commands browse and edit are useful commands that can be typed to list the information contained into the dataset, for that reason they are briefly described once again as follows:

browse shows you the dataset in spreadsheet format without options to modify the original data.

browse varname You can browse the whole dataset or you can specify the variables you wish to browse.

edit shows you the dataset in an excel type sheet with the possibility to be modified (Data Editor). You can specify also the particular variable you want to modify. As it can be seen, the Data Editor has the option to **delete** variables and observations.

To keep the changes click the button **Preserve** of the Data Editor Menu

To cancel the changes click the button **Restore** of the Data Editor Menu

6.2 Describing Data

When opening a new dataset that the user may not be very familiar with, it is always useful to have a general idea about its contents. To do so the command `describe` gives you the number of observations, the variable names, its description (or label) the storage type (string, float, etc). Some of the arguments more commonly used are:

describe to describe the contents of the whole dataset. The abbreviate version of the command is **des**

describe varname (s) describe only the variable(s) specified.

describe using filename describe the contents of the specified dataset: *filename*.

Example 6:

To describe the contents of the whole dataset called auto.dta

des

```

Stata/IC 11.1 - \\sernt27\stata11$\ado\base\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help

Review
/ Command
1 do "C:\Users\Usuar
2 do "C:\Users\Usuar
3 des

. des
Contains data from \\sernt27\stata11$\ado\base\auto.dta
  obs:      74      1978 Automobile Data
  vars:      12      13 Apr 2009 17:45
  size:    3,478 (99.9% of memory free)  (_dta has notes)

variable name  storage type  display format  value 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
headroom      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.)
displacement  int      %8.0g         Displacement (cu. in.)
gear_ratio    float   %6.2f         Gear Ratio
foreign       byte     %8.0g         origin
car type

```

The **variable label** gives you the description of the variable, which cannot be always inferred by reading the name. For instance the variable called **rep78** corresponds to the record of repairs in 1978 which would be difficult to guess without the description or a good manual for the survey nerby.

The **storage type** is another useful argument available with the description table. It allows you to know whether there are some information stored as **non-numerical** (i.e. strings) which must be taken into account when thinking in using such variable for numerical calculations.

The different types of storages used by STATA are described as follows:

float	for REAL numbers (7 digits of accuracy)
double	for REAL numbers (16 digits of accuracy)
byte	for INTEGERS (between -127 and 100)
int	for INTEGERS (between -32.767 and 32.740)
long	for INTEGERS (between -2.147.483.647 and 2.147.483.620)
str	for characters (from 1 to 80 characters. SE version allows until 244)

6.3 Tabulating Data

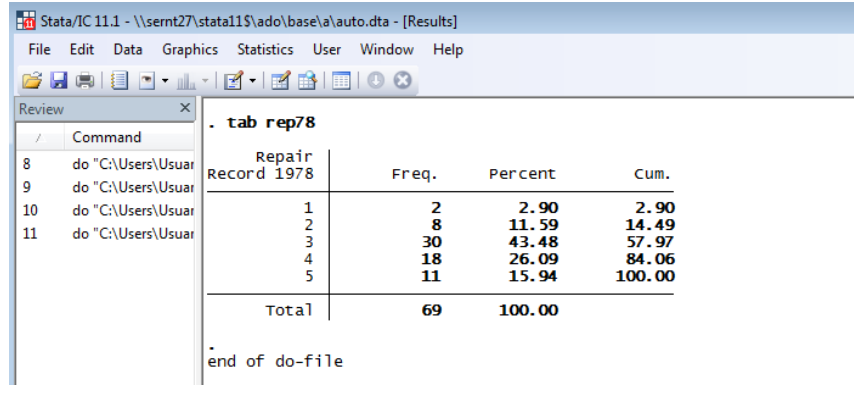
It is also possible to produce frequency counts for the variables of the dataset.

tabulate varname tabulates the values of the variable specified (varname) by showing the number of observations and its respective participation per case. The command abbreviation is **tab**

Example 7:

In the data `auto.dta` you are interested in tabulating the variable `rep78` you type in the do-file or command window the following argument getting a one-way frequency table as follows:

tab rep78



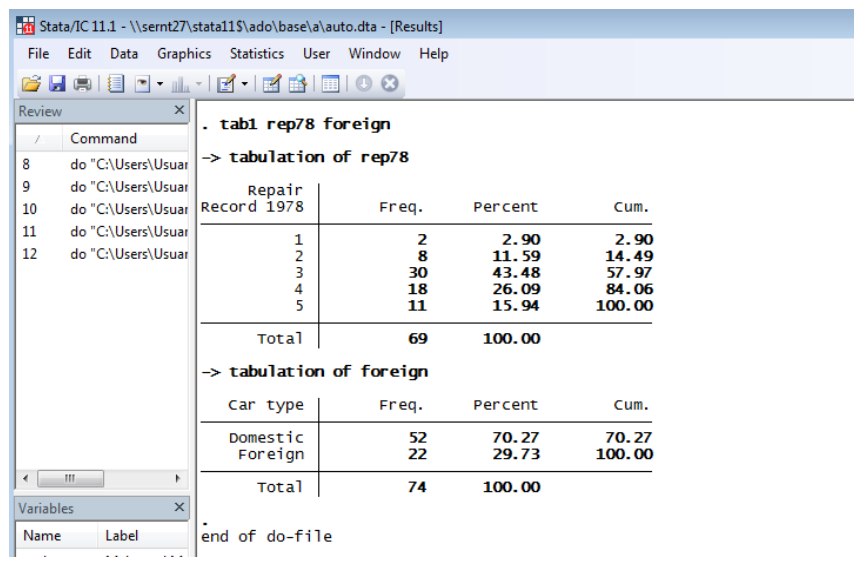
tab1 varlist

creates a one-way tabulation for each variable specified in the varlist. This is useful when you want to make one-way tabulation of different variables without typing the above command several times.

Example 8:

To tabulate one-way tables of `rep78` and `foreign` without typing twice the command:

tab1 rep78 foreign

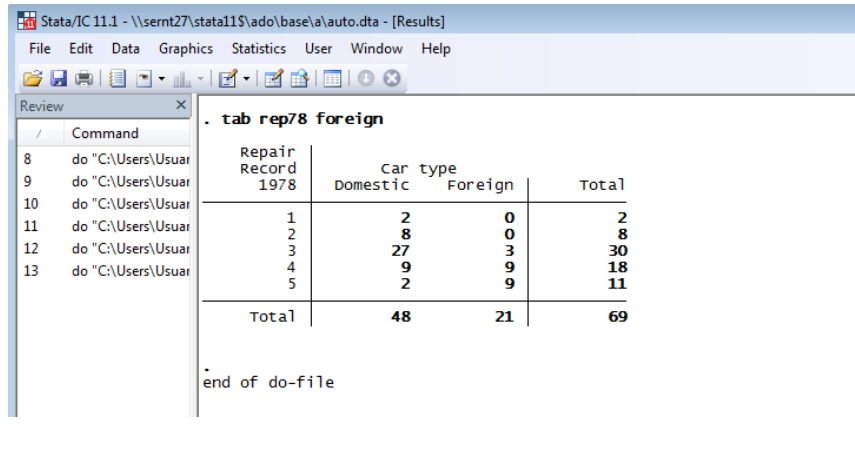


`tabulate varname1 varname2` shows a cross tabulation between two variables (`varname1 varname2`).

Example 9:

You are interested in seeing how the repairs vary per type of car (i.e. whether the car is domestic or foreign), to do so you can type the following:

tab rep78 foreign



```
Stata/IC 11.1 - \\sernt27\stata11$\ado\base\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
/ Command
8 do "C:\Users\Usuar
9 do "C:\Users\Usuar
10 do "C:\Users\Usuar
11 do "C:\Users\Usuar
12 do "C:\Users\Usuar
13 do "C:\Users\Usuar

. tab rep78 foreign

Repair Record 1978 | Car type | Total
                    | Domestic | Foreign |
-----|-----|-----|
1 | 2 | 0 | 2
2 | 8 | 0 | 8
3 | 27 | 3 | 30
4 | 9 | 9 | 18
5 | 2 | 9 | 11
-----|-----|-----|
Total | 48 | 21 | 69

.
end of do-file
```

Example 10:

Now, In addition to the above cross tabulation you are interested in whether the headroom changes by the origin of the car (i.e the foreign variable) or whether it changes by the number of repairs. To do it is possible to type **tab2**

tab2 rep78 foreign headroom

Stata/IC 11.1 - \\sernt27\stata11S\ado\base\auto.dta - [Results]

File Edit Data Graphics Statistics User Window Help

Review

Command

1 do "C:\Users\Usuar...

-> tabulation of rep78 by foreign

Repair Record 1978	Car type		Total
	Domestic	Foreign	
1	2	0	2
2	8	0	8
3	27	3	30
4	9	9	18
5	2	9	11
Total	48	21	69

-> tabulation of rep78 by headroom

Repair Record 1978	Headroom (in.)					Total
	1.5	2.0	2.5	3.0	3.5	
1	1	1	0	0	0	2
2	0	3	0	0	1	8
3	0	5	5	4	10	30
4	2	1	5	3	2	18
5	0	3	4	4	0	11
Total	3	13	14	11	13	69

Variables

Name	Label
make	Make and Model
price	Price
mpg	Mileage (mpg)
rep78	Repair Record
headroom	Headroom (in.)
trunk	Trunk space (in.)
weight	Weight (lbs.)
length	Length (in.)
turn	Turn Circle (ft.)

-> tabulation of foreign by headroom

Repair Record 1978	Headroom (in.)			Total
	4.0	4.5	5.0	
1	0	0	0	2
2	2	1	1	8
3	3	3	0	30
4	5	0	0	18
5	0	0	0	11
Total	10	4	1	69

tabulate varname, nolabel

tabulates the values of the variable specified (varname) in the same way described before, and when entries have been assigned a label the option **nolabel** ignores it showing the number of how the data was originally entered.

Example 11:

To see which values have been assigned to each entry of the *foreign* variable you can type the following:

```
tab foreign
tab foreign, nolabel
```

```
Stata/IC 11.1 - \\sernt27\stata115\ado\base\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
/ Command
2 do "C:\Users\Usuar

. tab foreign
+-----+-----+-----+-----+
| Car type | Freq. | Percent | Cum. |
+-----+-----+-----+-----+
| Domestic | 52    | 70.27   | 70.27 |
| Foreign  | 22    | 29.73   | 100.00|
+-----+-----+-----+-----+
| Total    | 74    | 100.00  |       |
+-----+-----+-----+-----+
. tab foreign, nolabel
+-----+-----+-----+-----+
| Car type | Freq. | Percent | Cum. |
+-----+-----+-----+-----+
| 0        | 52    | 70.27   | 70.27 |
| 1        | 22    | 29.73   | 100.00|
+-----+-----+-----+-----+
| Total    | 74    | 100.00  |       |
+-----+-----+-----+-----+
. end of do-file
.
```

tabulate varname, missing tabulates the values of the variable (varname), shows the number of observations per case and with the *option missing* shows the missing values (.), if any.

Example 12:

You are interested to know how many observations in your dataset are missing (as follows, sometimes the missing information is represented with a period "."): :

```
tab rep78, missing
```

```
Stata/IC 11.1 - \\sernt27\stata115\ado\base\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
/ Command
1 do "C:\Users\Usuar
2 do "C:\Users\Usuar
3 do "C:\Users\Usuar

. tab rep78, missing
+-----+-----+-----+-----+
| Repair Record 1978 | Freq. | Percent | Cum. |
+-----+-----+-----+-----+
| 1                   | 2     | 2.70    | 2.70  |
| 2                   | 8     | 10.81   | 13.51 |
| 3                   | 30    | 40.54   | 54.05 |
| 4                   | 18    | 24.32   | 78.38 |
| 5                   | 11    | 14.86   | 93.24 |
| .                   | 5     | 6.76    | 100.00|
+-----+-----+-----+-----+
| Total                | 74    | 100.00  |       |
+-----+-----+-----+-----+
. end of do-file
.
```

tabulate varname if tabulates the value of the variable (varname) that has been restricted by one condition and shows the number of observations per case.

Example 13:

You are interested in tabulating those cases that have 3 or less repairs

tab rep78 if rep78<=3

The screenshot shows the Stata command window with the command `. tab rep78 if rep78<=3` and the resulting output table. The output table has columns for Repair Record 1978, Freq., Percent, and Cum. The data is as follows:

Repair Record 1978	Freq.	Percent	Cum.
1	2	5.00	5.00
2	8	20.00	25.00
3	30	75.00	100.00
Total	40	100.00	

tabulate varname1 varname2, col row when using the two-ways tabulation, the options *col* and *row* indicates percentages by column and by rows including missing as an extra value for the variable.

Example 14:

You are interested in the distribution of cars by origin (i.e. domestic or foreign) per each of the reparation records

tab rep78 foreign, col row

The screenshot shows the Stata command window with the command `. tab rep78 foreign, col row` and the resulting output table. A key is provided to explain the percentages in the table: frequency, row percentage, and column percentage. The output table has columns for Repair Record 1978, Car type (Domestic, Foreign), and Total. The data is as follows:

Repair Record 1978	Car type		Total
	Domestic	Foreign	
1	2 100.00 4.17	0 0.00 0.00	2 100.00 2.90
2	8 100.00 16.67	0 0.00 0.00	8 100.00 11.59
3	27 90.00 56.25	3 10.00 14.29	30 100.00 43.48
4	9 50.00 18.75	9 50.00 42.86	18 100.00 26.09
5	2 18.18 4.17	9 81.82 42.86	11 100.00 15.94
Total	48 69.57 100.00	21 30.43 100.00	69 100.00 100.00

6.4 Descriptive Statistics

In addition to the general contents of the dataset provided by the command **describe**, the descriptive statistics are an important tool that helps to know better the data. In STATA the descriptive statistics are produced by the command **summarise**. It shows the number of observations, the mean, the standard deviation, the minimum and the maximum value of the variables.

summarise summarise the numerical values of **all the variables** in the dataset. The command abbreviation is **sum**

Example 15:

You are interested in the mean, minimum and maximum values per variable:

sum

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

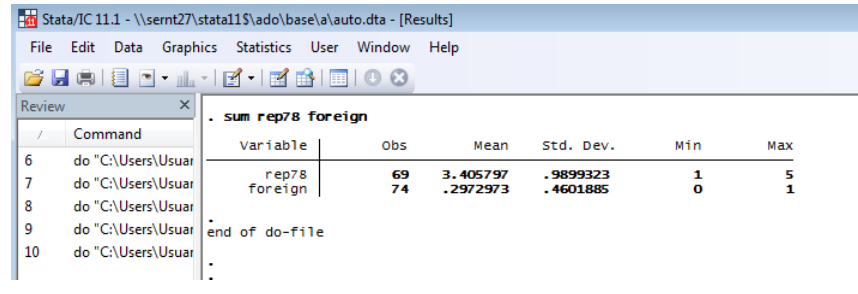
Notice that in the example the variable **make** appears as it would have 0 observations. Actually it has observations (you can check this by tabulating the variable) but the variable has been stored as a *string variable* or *non numerical variable* which in this case correspond to the name of the car maker which cannot be averaged in text format.

summarise varname(s) summarise the numerical values of **one or more variables** (varnames) in the dataset. It shows the number of observations, the mean, the standard deviation, the minimum and the maximum value of the variables.

Example 16:

You are interested in the descriptive statistics for the repairs record and for the foreign variable:

```
sum rep78 foreign
```



```
Stata/IC 11.1 - \\sernt27\stata11\ado\base\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
/ Command
6 do "C:\Users\Usuar
7 do "C:\Users\Usuar
8 do "C:\Users\Usuar
9 do "C:\Users\Usuar
10 do "C:\Users\Usuar

. sum rep78 foreign
Variable | Obs      Mean      Std. Dev.      Min      Max
-----+-----+-----+-----+-----+-----
rep78   |   69   3.405797   .9899323         1         5
foreign |   74   .2972973   .4601885         0         1
.
. end of do-file
.
```

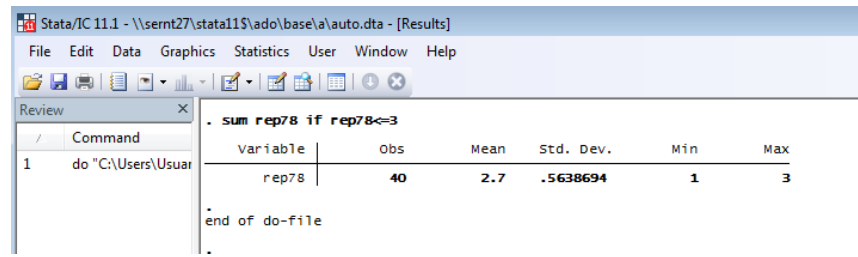
summarise varname(s) if

summarise the numerical values of **one or more variables** (varnames) in the dataset given the condition imposed by the argument **if**. It shows the number of observations, the mean, the standard deviation, the minimum and the maximum value of the variables.

Example 17:

You are interested in the descriptive statistics for the repairs record in cases with less than 3 repairs:

```
sum rep78 if rep78<=3
```



```
Stata/IC 11.1 - \\sernt27\stata11\ado\base\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
/ Command
1 do "C:\Users\Usuar

. sum rep78 if rep78<=3
Variable | Obs      Mean      Std. Dev.      Min      Max
-----+-----+-----+-----+-----+-----
rep78   |   40     2.7     .5638694         1         3
.
. end of do-file
.
```

6.5 Creating and Manipulating Variables

Generating New Variables

To create new variables STATA counts with the **generate** command. It allows the user to generate new variables that could be a mathematical combination or a particular condition of the existing ones.

generate newvar=exp

creates a new variable (newvar) equal to any valid expression (exp). The expression could be an algebraic expression of other variables or just a numerical value. The command may be abbreviated typing **gen**, **ge** or just **g**.

Example 18:

You can create new variables containing the values of one and two respectively and a new variable equal for the sum of both of them:

```
gen x=1
gen y=2
gen xy= x + y
```

The variable with ones has been named **x**, while **y** will correspond to the variable with twos and the variable with the name **xy** is the sum of both of them:

You can also create a variable with values only for those having less than three repairs:

```
gen repairs3 = rep78 if rep78<3
```

replace varname=exp changes the contents of an existing variable (varname). The command cannot be abbreviated.

Example 19:

You want to create a dummy variable indicating whether the car has three or less repairs (i.e. a variable taking the value of one if the condition of less than three repairs holds, zero otherwise)

There are different ways to create dummy variables in STATA we shows two of them:

```
gen dummyrepairs3=1 if rep78<3
replace dummyrepairs3=0 if dummyrepairs3!=1
```

Alternatively you can type:

```
gen dummyrepairs3=(rep78<3)
```

Notice that not always is possible to use the short version of the argument since you may have different conditions when allocating the ones and the zeros in a new variable. In fact, in some cases the entries with zero would not necessarily correspond to everything else after the one has been assigned.

egen newvar=fnc(arg) creates variables that are more difficult to compute. **egen** is more powerful than **gen**. For instance, you would be interested in create a new variable that is the mean of the other or a new variable that count the number of observations, etc. Thus, in the command syntax **fnc** correspond to the function you want to

execute: mean, count, min, total, sd, median, etc. **arg** correspond to the variables or numbers you need to generate the new variable.

Example 20:
Generating a new variable that is the average of the repairs records in the dataset

```
egen meanrep78=mean(rep78)
```

Now, you are interested in generating a new variable that is the total of the repairs records in the dataset

```
egen sumrep78=sum(rep78)
```

rename oldvar newvar change the name of an existing variable.

Example 21:
Changing the name of sumrep78 by totalrepairs

```
ren sumrep78 totalrepairs
```

Deleting and Keeping Variables

drop varname(s) eliminates the variable(s) specified from the memory.
drop _all eliminates the data set from memory.

keep varname(s) keep the variables you specified eliminating the other ones.

6.6 Labeling variables

label var sets or changes the variable's label. The label should be in double quotes.

Example 22:
Labeling a variable created as the average of an existent variable:

```
label var meanrep78 "Average Repairs"
```

label define creates a value label (it gives a label to the possible values of the variable):

```

Example 23:
Defining the ones and zeros in a dummy variable:

label define valuesdummyrepairs3 1 "Less than 3
repairs 0 "Otherwise"

```

label values associates the value label we created before with the variable.

```

Example 24:
Allocating the ones and zero definitions stored in
valuesdummyrepairs2 into a variable called repairs3
which contains the value of one for those
observations with less than 3 repairs and zero for
the remind cases.

label values repairs3 valuesdummyrepairs3

```

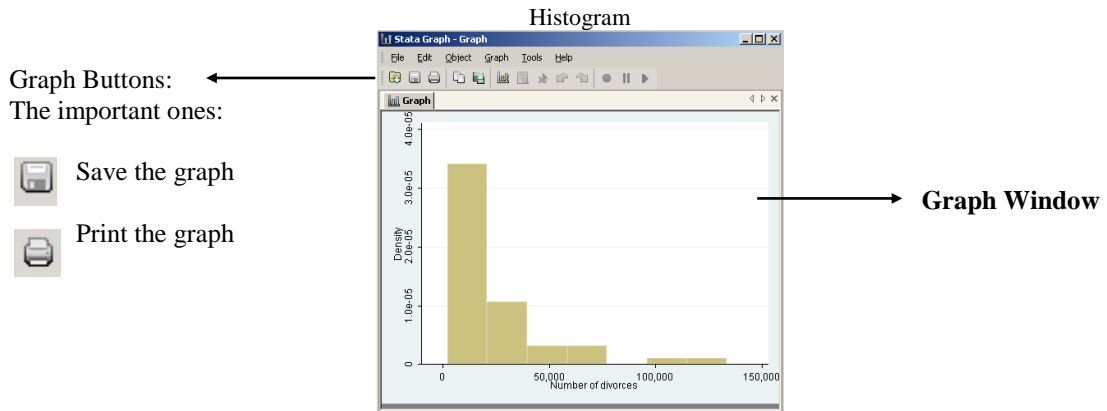
6.7 Working with String Variables

String Variables are variables that could be letters or special characters. Sometimes STATA stores numeric values as *strings*; this the main source of a “**type mismatch**” error. Then, we would like to convert this variable into a number or viceversa. The following list of commands will help with that:

destring varname, replace	converts the string variable <i>varname</i> into a numeric variable.
gen varname=real(string_num)	converts the string values into numeric values.
encode varname,gen(newvar)	convert a string variable into a numeric variable (using the character values as labels).
decode varname,gen(newvar)	convert a number into string (numeric values must have value labels assigned to use them as the new string values for the variable).
gen varname=string(var_num)	converts numeric values into string.
generate newvar="stringvar"	generates a string variable (notice that the expression is enclosed by double quotes).

6.8 Graphs

You can use the graph button or the graph command to make a graph of your data. The graph button from the menu brings the topmost graph window.



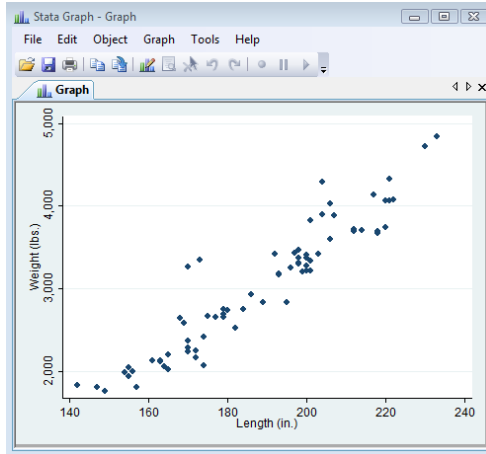
If you already know the kind of graph you need you can write down one of the following graph commands:

<code>histogram varname</code>	histogram of the variable <i>varname</i>
<code>graph twoway</code>	scatterplots, line plots, etc.
<code>scatter varname1 varname2</code>	scatterplots, line plots, etc.
<code>twoway scatter</code>	scatterplots.
<code>graph matrix</code>	scatter plot matrices.
<code>graph bar</code>	bar charts.
<code>graph dot</code>	dot charts.
<code>graph box</code>	box-and-whisker plots.
<code>graph pie</code>	pie charts.

Example 25:

To graph a *scatterplot* you type the command followed by the y-variable and x-variable as follows:

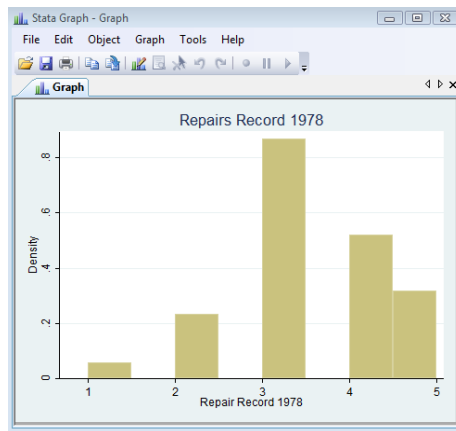
```
graph twoway scatter weight length
```



Example 26:

To draw a histogram of the repairs record in 1978 and to add a title to the graph:

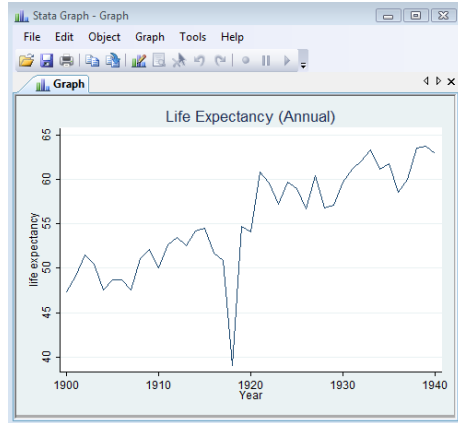
```
hist rep78, title(Repairs Record 1978)
```



Example 27:

To draw a time line for the life expectancy of a country and adding the respective title:

```
twoway (line le year),title(Life Expectancy  
(Annual))
```



where **le** and **year** are the y-variable and x-variable

Saving Graphs

To save the graph you can use the save button on the Graph Window or right click on the graph window and select **Save Graph** or select the *File>Save* from the Graph Window.

Printing Graphs

To print the graph you can use the save button on the Print Window, or right click on the graph window and select **Print Graph** or select the *File>Print* from the Graph Window.

7. Joining two different datasets

To join two different data sets we can use the commands **append** and **merge**.

7.1 Appending Data

Sometimes you would like to add observations to an existing dataset: additional respondents, additional years, etc. Since such information could come from a different dataset then you would want to join both datasets. The command **append** does this.

append combine two datasets vertically adding more observations to the end of the first dataset opened (master data). It is not necessarily that both datasets have the same variables.

Example 28:

You have two separate datasets with information about domestic (**domestic.dta**) and foreign (**foreign.dta**) cars respectively. Each dataset contains the same variables: height, weight, maker, etc:

The screenshot shows the Stata command window with the following commands and output:

```

. clear
. set mem 100m (102400k)
. cd C:\Users\Usuario\Desktop\
C:\Users\Usuario\Desktop
. use domestic.dta, clear (1978 Automobile Data)
. sum

```

Variable	Obs	Mean	Std. Dev.	Min	Max
make	0				
price	52	6072.423	3097.104	3291	15906
mpg	52	19.82692	4.743297	12	34
rep78	48	3.020833	.837666	1	5
foreign	52	0	0	0	0

```

. use foreign.dta, clear (1978 Automobile Data)
. sum

```

Variable	Obs	Mean	Std. Dev.	Min	Max
make	0				
price	22	6384.682	2621.915	3748	12990
mpg	22	24.77273	6.611187	14	41
rep78	21	4.285714	.7171372	3	5
foreign	22	1	0	1	1

```

. end of do-file
.
.

```

You can pool the information in one dataset by adding the information of the second dataset literally "below" the first dataset:

```

use domestic.dta
append using foreign.dta

```

The screenshot shows the Stata command window with the following commands and output:

```

. clear
. set mem 100m (102400k)
. cd C:\Users\Usuario\Desktop\
C:\Users\Usuario\Desktop
. use domestic.dta (1978 Automobile Data)
. append using foreign.dta (label origin already defined)
. sum

```

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
foreign	74	.2972973	.4601885	0	1

```

. end of do-file
.
.

```

7.2 Merging Data

Sometimes we need to add more variables to an existing dataset. Suppose for instance you have a State's census separated in different datasets, i.e in a first dataset you have the number of inhabitants while in a second dataset you have the marital status information:

census1.dta					census2.dta				
	state	region	pop	stateid		state	death	marriage	stateid
1	Alabama	South	3,893,888	1	1	Alabama	35,305	49,018	1
2	Alaska	West	401,851	2	2	Alaska	1,604	5,361	2
3	Arizona	West	2,718,215	3	3	Arizona	21,226	30,223	3
4	Arkansas	South	2,286,435	4	4	Arkansas	22,676	26,513	4
5	California	West	23,667,902	5	5	California	186,428	210,864	5
6	Colorado	West	2,889,964	6	6	Colorado	18,925	34,917	6
7	Connecticut	NE	3,107,576	7	7	Connecticut	26,005	26,048	7
8	Delaware	South	594,338	8	8	Delaware	5,123	4,437	8
9	Florida	South	9,746,324	9	9	Florida	104,190	108,344	9
10	Georgia	South	5,463,105	10	10	Georgia	44,230	70,638	10

You can join both datasets to increase the amount of information for the same observations, in the case above, for the same states. To do this the command `merge` is needed.

STATA 11 has easy the procedure of merging datasets with new arguments for `merge`. Although is more precise, the old fashion way of merging could help to understand what STATA is doing when merging datasets. The following examples consider the two ways of merging the above datasets.

Example 29:
Merging in Stata11

```
merge 1:1 varlist using filename
```

Merging one to one the observations from both datasets using a **key** variable (a unique identifier available in both datasets), in this we have an state' id:

```
use census1.dta, clear  
merge 1:1 stateid using census2.dta
```

```
Stata/IC 11.1 - C:\Users\Usuario\Desktop\census1.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
1 do "C:\Users\Usuario\Ap
. clear
. set mem 100m
(102400k)
. cd "C:\Users\Usuar io\Desktop\
C:\Users\Usuar io\Desktop
.
. use census1.dta, clear
(1980 Census data by state)
. merge 1:1 stateid using census2.dta
Result # of obs.
not matched 0
matched 10 (_merge==3)
.
end of do-file
```

After merging both datasets, the new dataset looks like the following:

	state	region	pop	stateid	death	marriage	_merge
1	Alabama	South	3,893,888	1	35,305	49,018	matched (3)
2	Alaska	West	401,851	2	1,604	5,361	matched (3)
3	Arizona	West	2,718,215	3	21,226	30,223	matched (3)
4	Arkansas	South	2,286,435	4	22,676	26,513	matched (3)
5	California	West	23,667,902	5	186,428	210,864	matched (3)
6	Colorado	West	2,889,964	6	18,925	34,917	matched (3)
7	Connecticut	NE	3,107,576	7	26,005	26,048	matched (3)
8	Delaware	South	594,338	8	5,123	4,437	matched (3)
9	Florida	South	9,746,324	9	104,190	108,344	matched (3)
10	Georgia	South	5,463,105	10	44,230	70,638	matched (3)

STATA 11 generates a variable called `_merge` which summarises the merging procedure as it is shown below. Notice that the **master data** corresponds to the dataset currently in memory, **using data** corresponds to the dataset you are merging with. In the example above, *census1.dta* corresponds to the master data while *census2.dta* corresponds to the using data.

`_merge = 1` **master** originally appeared in **master dataset** only:
It refers to the number of observations from the master data that didn't match with the using data.

`_merge = 2` **using** originally appeared in **using dataset** only:
It refers to the number of observations from the using data that didn't match with the master data.

`_merge = 3` **match** originally appeared in both datasets:
It refers to the number of observations from the master data that matched with the observations in the using data.

It may be the case that some datasets present the information in the following way: one dataset for individual information about the patients in a hospital and a separate dataset with information about the characteristics of the hospital as it is shown below:

patients_info.dta

	patient	sex	agegrp	bp_before	bp_after	hospital_id
1	1	Male	30-45	143	153	H35
2	2	Male	30-45	163	170	H24
3	3	Male	30-45	153	168	H13
4	4	Male	30-45	153	142	H24
5	5	Male	30-45	146	141	H35
6	6	Male	30-45	150	147	H24
7	7	Male	30-45	148	133	H13
8	8	Male	30-45	153	141	H24
9	9	Male	30-45	153	131	H13
10	10	Male	30-45	158	125	H15

hospitals_info.dta

	hospital_id	Private	beds_avail~k
1	H13	1	35
2	H15	1	40
3	H24	0	8
4	H35	0	0

You can add the characteristics of the hospital to the patients thanks to the fact that both datasets share one identifier: the hospital code. This is called in STATA *many to one* or *one-to many merge* as it will be shown in the examples below:

Example 30:
Merging *one-to-many*

```
use hospitals_info.dta, clear
merge 1:m using patients_info.dta
save patients_hospitals.dta
```

After browsing the new dataset, it will look like:

	hospital_id	Private	beds_avail~k	patient	sex	agegrp	bp_before	bp_after	_merge
1	H13	1	35	9	Male	30-45	153	131	matched (3)
2	H15	1	40	10	Male	30-45	158	125	matched (3)
3	H24	0	8	2	Male	30-45	163	170	matched (3)
4	H35	0	0	5	Male	30-45	146	141	matched (3)
5	H13	1	35	3	Male	30-45	153	168	matched (3)
6	H13	1	35	7	Male	30-45	148	133	matched (3)
7	H24	0	8	6	Male	30-45	150	147	matched (3)
8	H24	0	8	4	Male	30-45	153	142	matched (3)
9	H24	0	8	8	Male	30-45	153	141	matched (3)
10	H35	0	0	1	Male	30-45	143	153	matched (3)

Example 31:
Merging *many-to-one*

```
use patients_info.dta, clear
merge m:1 using hospitals_info.dta
save patients_hospitals.dta
```

After browsing the new dataset, it will look like:

	patient	sex	agegrp	bp_before	bp_after	hospital_id	Private	beds_avail~k	_merge
1	9	Male	30-45	153	131	H13	1	35	matched (3)
2	3	Male	30-45	153	168	H13	1	35	matched (3)
3	7	Male	30-45	148	133	H13	1	35	matched (3)
4	10	Male	30-45	158	125	H15	1	40	matched (3)
5	4	Male	30-45	153	142	H24	0	8	matched (3)
6	8	Male	30-45	153	141	H24	0	8	matched (3)
7	6	Male	30-45	150	147	H24	0	8	matched (3)
8	2	Male	30-45	163	170	H24	0	8	matched (3)
9	5	Male	30-45	146	141	H35	0	0	matched (3)
10	1	Male	30-45	143	153	H35	0	0	matched (3)

Notice that in both examples the final output only differs in the way the variables are organised. In the *one-to-many* merging procedure, the information from the hospital data (which contain *one* observation per hospital) is presented first in the new dataset, while in the *many-to-one* merge the hospital' patients (*many* observations per hospital) are the first variables in the dataset.

It is useful to delete the `_merge` variable created by STATA to avoid error messages in the future when you want to conduct new merges.

For instance, the arguments for the above example would be the following in versions before 11:

```
use hospitals_info.dta, clear
sort hospital_id
save, replace

use patients_info.dta, clear
sort hospital_id
save, replace

use hospitals_info.dta, clear
merge hospital_id using patients_info.dta
tab _merge
drop _merge
save patients_hospital.dta
```

Summarizing the steps to conduct merging with previous versions to STATA 11:

Step 1: sort the key variable or identifier in both datasets.

Step 2: open the dataset you want to be the “master data” followed by the merge command.

Step 3: Tabulate the variable **_merge** to have information about the merging procedure and drop it afterwards.

Step 4: Save with the new dataset.

8. Saving Output

Logs

Logs are files that store the results of your analysis such as summary statistics, variables frequencies, etc. The logs can be viewed using the viewer and can be easily printed. To start and finish the log you should type **log using** and **log close** respectively. You can also suspend the storage of some results by typing **log off** and to restart the storage again you type **log on**.

```
Example 32:

log using census.txt
sum
log off

gen gend=(e03==1)

log on
reg wage gend
log close
```

9. Basic Econometric Analysis

To run a linear regression between a dependent variable and the independent variables use the command **regress** (abbreviated with **reg**) with the following syntax:

```
regress depvar indvar(s)
```

The variable after the command **regress** is the dependent variable or the variable you want to explain. The variables written after that variable are those that you will use to explain the first one (independent variables) .

Example 33:

To run an OLS regression where car's price is explained by its weight, length, origin and a constant you type the following:

```
reg price weight length foreign
```

Source	SS	df	MS			
Model	348565467	3	116188489	Number of obs =	74	
Residual	286499930	70	4092856.14	F(3, 70) =	28.39	
Total	635065396	73	8699525.97	Prob > F =	0.0000	
				R-squared =	0.5489	
				Adj R-squared =	0.5295	
				Root MSE =	2023.1	

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
weight	5.774712	.9594168	6.02	0.000	3.861215	7.688208
length	-91.37083	32.82833	-2.78	0.007	-156.8449	-25.89679
foreign	3573.092	639.328	5.59	0.000	2297.992	4848.191
_cons	4838.021	3742.01	1.29	0.200	-2625.183	12301.22

end of do-file

To exclude the constant, you can add the following the option *noconstant* as follows:

```
reg price weight length foreign, noconstant
```

predict newvarname create a variable with the values predicted by the model after the regression was run (it must be executed after the estimation command, e.g. regress).

Example 34:

Calculating the fitted values for the dependent variable in a new variable called **pricehat**:

```
reg price weight length foreign  
predict pricehat, xb
```

If your dependent variable is binary (i.e. 0 or 1), then you will have to use alternative models that require different commands such as **logit** and **probit**.

Presenting regression results

The regression output that STATA shows by default is not the most attractive way to present the results in a document. However, there are a series of commands that make possible to convert STATA' regression results into a standard table.

estimates store *name* save the results of the current estimation with the *name* you specified (the abbreviation is eststo).

estimates table *name* tabulates the results saved under the specified *name* (the abbreviation is esttab).

esttab *name using filename.rtf* tabulates the results saved under the specified name and export them into a file rtf format, readable in text processors such as Word.

Example 35:

Tabulating the results of two different regressions, the first one including the constant and the second one excluding it. In addition to this you would like to present the standard errors (se) rather than the t-statistic (default) and you would like to name your table "Table 1: Regression Results", then you follow the next arguments:

```
reg price weight length foreign
estimates store Reg1
```

```
reg price weight length foreign, noconstant
estimates store Reg2
```

```
esttab Reg1 Reg2 using OLS_Results.rtf, se
title("Regression Results")
```

The following table is produced in rtf file:

Table 1: Regression Results		
	(1)	(2)
	price	price
weight	5.775*** (0.959)	4.868*** (0.658)
length	-91.37** (32.83)	-51.51*** (11.33)
foreign	3573.1*** (639.3)	3792.4*** (619.3)
_cons	4838.0 (3742.0)	
<i>N</i>	74	74

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

10. Starting with Time Series

First, we need to specify that we are working with time series data and the date variable that tells STATA the date format (day, quarter, year, etc.). To do that we type:

```
tsset datevar
```

where *datevar* is the name of the date variable in your dataset. After that you can continue performing the statistical analysis using the additional commands available to deal with time series.

Example 36:

Defining the time variable for the following dataset:

	date	open	high	low	close	volume	change
1	03jan2001	1283.27	1347.76	1274.62	1347.56	18,807	64.29004
2	04jan2001	1347.56	1350.24	1329.14	1333.34	21,310	-14.22009
3	05jan2001	1333.34	1334.77	1294.95	1298.35	14,308	-34.98999
4	08jan2001	1298.35	1298.35	1276.29	1295.86	11,155	-2.48999
5	09jan2001	1295.86	1311.72	1295.14	1300.8	11,913	4.940063
6	10jan2001	1300.8	1313.76	1287.28	1313.27	12,965	12.46997
7	11jan2001	1313.27	1332.19	1309.72	1326.82	14,112	13.54993
8	12jan2001	1326.82	1333.21	1311.59	1318.55	12,760	-8.269897
9	16jan2001	1318.32	1327.81	1313.33	1326.65	12,057	8.099976
10	17jan2001	1326.65	1346.92	1325.41	1329.47	13,491	2.819946
11	18jan2001	1329.89	1352.71	1327.41	1347.97	14,450	18.5

```
tsset date
```

Inputting Time Series Dates

When we have the date component in separated variables, we would like to combine them by creating a new date variable. To do that we can use the command **gen** again and the following expressions:

mdy (month, day, year)	for daily data
yw (year, week)	for weekly data
ym (year, month)	for monthly data
yq (year, quarter)	for quarterly data
yh (year, half-year)	for half yearly data

For instance, you have two variables, month and year. You can combine them in order to have one date variable by typing:

```
gen month=ym(year, month)
```

11. Summary

Useful Commands, recalling the command syntax:

`command [varlist] [if] [in] [, options]`

Command Name	Abreviation	Function	Example
Starting...			
<code>clear</code>	<code>clear</code>	Clear Memory (closes any other dataset previously opened)	<code>clear</code>
<code>set memory</code>	<code>set mem</code>	Change amount of memory allocated to Stata	<code>set mem 100m</code>
<code>set maxvar</code>	<code>set maxvar</code>	Set maximum number of variables in Stata/MP and Stata/SE	<code>set maxvar 2000</code>
<code>cd:\</code>		Change directory	<code>cd "C:\MyDocuments\MyProject"</code>

Opening Datasets			
<code>use</code>	<code>use</code>	Open a Stata-format dataset (ending .dta)	<code>use mydata.dta, clear</code>
<code>insheet</code>	<code>insheet</code>	Read ASCII (text) data created by a spreadsheet	<code>insheet using mydata.csv</code>
<code>infile</code>	<code>infile</code>	Read unformatted ASCII (text) data	<code>infile str18 make price mpg rep78</code>
Saving Datasets			
<code>save</code>	<code>save</code>	Save data in memory to file	<code>save mydata.dta</code>
<code>save, replace</code>	<code>save, replace</code>	Overwrite existing dataset	<code>save mydata.dta, replace</code>
Exit Stata			
<code>exit</code>	<code>exit</code>	Exit Stata	<code>exit</code>

Describing and Listing the Data			
<code>describe</code>	<code>desc</code>	Produces a summary of the dataset in memory or of the data stored in a Stata-format dataset	<code>desc</code> <code>desc price</code>
<code>tabulate</code>	<code>tab</code>	One-way tables of frequencies and Two-way tables of frequencies	<code>tab gender</code> <code>tab gender age</code>
<code>tab1</code>	<code>tab1</code>	One-way tables of frequencies	<code>tab1 gender</code> <code>tab1 gender age</code>
<code>tab2</code>	<code>tab2</code>	Two-way tables of frequencies	<code>tab2 gender age</code> <code>tab2 gender age education</code>
<code>summarise</code>	<code>sum</code>	Calculates and displays a variety of univariate summary statistics	<code>sum</code> <code>sum age gender</code>
<code>edit</code>	<code>edit</code>	Brings up a spreadsheet-style data editor for entering new data and editing existing data.	<code>edit</code>
<code>list</code>	<code>l</code>	List (displays in the screen) values of the variables	<code>list</code> <code>list in 1/10</code> <code>list age</code>

Manipulating the Data			
<code>generate</code>	<code>gen</code>	Creates a new variable. The values of the variable are specified by =exp.	<code>gen x = 1</code> <code>gen age = year-yearborn</code> <code>gen dummy=(sex==1)</code>
<code>egen</code>	<code>egen</code>	Creates newvar of the optionally specified storage type equal to fcn(arguments), i.e. mean, sd, etc.	<code>egen average=mean(price)</code> <code>egen total=sum(price)</code>
<code>replace</code>	<code>replace</code>	Replace contents of existing variable	<code>gen x= 1</code> <code>replace x=2 if year==1980</code>
<code>keep</code>	<code>keep</code>	Keep variables / Observations	<code>keep age gender / keep if age<10</code>
<code>drop</code>	<code>drop</code>	Drop variables / Observations	<code>drop education / drop if age>10</code>
<code>rename</code>	<code>ren</code>	Rename variable	<code>ren years_school education</code>

<code>edit</code>	<code>edit</code>	Brings up a spreadsheet-style data editor for entering new data and editing existing data.	<code>edit</code>
<code>destring</code>	<code>destring</code>	Convert string variables to numeric variables and vice versa	<code>destring code, replace</code>
			<code>tostring code, replace</code>

Joining Datasets			
<code>append</code>	<code>app</code>	Appends Stata-format datasets stored on disk to the end of the dataset in memory	<code>use data1.dta</code> <code>append using data2.dta</code>
<code>merge</code>	<code>mer</code>	Merge joins corresponding observations from the dataset currently in memory with a second dataset (before Stata 11)	<code>use census1978.dta</code> <code>merge id using census1988.dta</code>
<code>merge 1:1</code>		One-to-one merge on specified key variables	<code>use census1978.dta</code> <code>merge 1:1 id using census1988.dta</code>
<code>merge 1:m</code>		One-to-many merge on specified key variables	<code>use individuals.dta</code> <code>merge m:1 id using families.dta</code>
<code>merge m:1</code>		Many-to-one merge on specified key variables	<code>use families.dta</code> <code>merge 1:m id using individuals.dta</code>
<code>sort</code>		Sort data	<code>sort</code>

Basic Regression Analysis			
<code>regress</code>	<code>reg</code>	Fits a model of depvar on indepvars using linear regression.	<code>reg price weight length type</code>
<code>estimates store</code>	<code>eststo</code>	Store and restore estimation results	
<code>estimates table</code>	<code>esttab</code>	Displays a table of coefficients and statistics for one or more sets of estimation results	<code>reg price weight length type</code> <code>eststo Reg1</code> <code>esttab Reg1 using Results.rtf</code>
<code>tsset</code>	<code>tsset</code>	Declare data to be time-series data	<code>tsset year</code>

Additional Operators

<code>if</code>	<code>if</code>	<code>if</code> at the end of a command means the command is to use only the data specified. <code>if</code> is allowed with most Stata commands.	<code>reg price weight length type if foreign==1</code>																								
			<code>sum weight if price<5000</code>																								
			<code>tab age if age>16</code>																								
Basic Operators																											
			<table border="0"> <thead> <tr> <th>Arithmetic</th> <th>Logical</th> <th>(numeric and string)</th> </tr> </thead> <tbody> <tr> <td>+ addition</td> <td>& and</td> <td>> greater than</td> </tr> <tr> <td>- subtraction</td> <td> or</td> <td>< less than</td> </tr> <tr> <td>* multiplication</td> <td>! not</td> <td>>= > or equal</td> </tr> <tr> <td>/ division</td> <td>~ not</td> <td><= < or equal</td> </tr> <tr> <td>^ power</td> <td></td> <td>== equal</td> </tr> <tr> <td>- negation</td> <td></td> <td>!= not equal</td> </tr> <tr> <td>+ string concatenation</td> <td></td> <td>~= not equal</td> </tr> </tbody> </table>	Arithmetic	Logical	(numeric and string)	+ addition	& and	> greater than	- subtraction	or	< less than	* multiplication	! not	>= > or equal	/ division	~ not	<= < or equal	^ power		== equal	- negation		!= not equal	+ string concatenation		~= not equal
Arithmetic	Logical	(numeric and string)																									
+ addition	& and	> greater than																									
- subtraction	or	< less than																									
* multiplication	! not	>= > or equal																									
/ division	~ not	<= < or equal																									
^ power		== equal																									
- negation		!= not equal																									
+ string concatenation		~= not equal																									

12. Useful sources for STATA Support

STATA website:

<http://www.stata.com/support/>

<http://www.stata.com/links/resources1.html>

UCLA STATA Reference Website:

<http://statcomp.ats.ucla.edu/stata/>

References

Anderson, Mike. An Introduction to Stata.<http://mit.edu/14.33/www/stata_B.pdf>

Panu Pelkonen, 2007. Class Notes: Introduction to Stata, Department of Economics, University of Essex.

Perez-Truglia, Ricardo, 2009. Applied Econometrics using Stata. Manual Department of Economics, Harvard University.

StataCorp LP, 2009. Stata User's Guide, Release 11. *Stata Press*

Yaffe, Robert, 2002. Getting Started with Stata for Ms Windows: A Brief Introduction. Information Technology Services, New York University.