HG
Revised September 2011

A quick introduction to STATA:

(by E. Bernhardsen, with additions by H. Goldstein)

1. How to access STATA from the pc's at the computer lab and elsewhere at UiO.

At the computer lab, after having logged in, you may find STATA 11 (or StataIC 11 or *Small Stata* if the StataIC11 does not open) under Programs on your computer, and you can start it straight away.

Otherwise you have to log in again at a remote desktop at the server "statwin2" or "statwin3". In that case do the following: Go to

Start -> Programs -> Accessories ->(Communications ->)[1] *Remote Desktop Connection*

Write *statwin2* (or *statwin3*) in the window in the menu that appears, and then press connect. Log in, and a new desktop should appear on the screen.

Now you can start STATA by:

Start -> Programs -> Stata 11 -> StataIC 11 (or *Small Stata*)

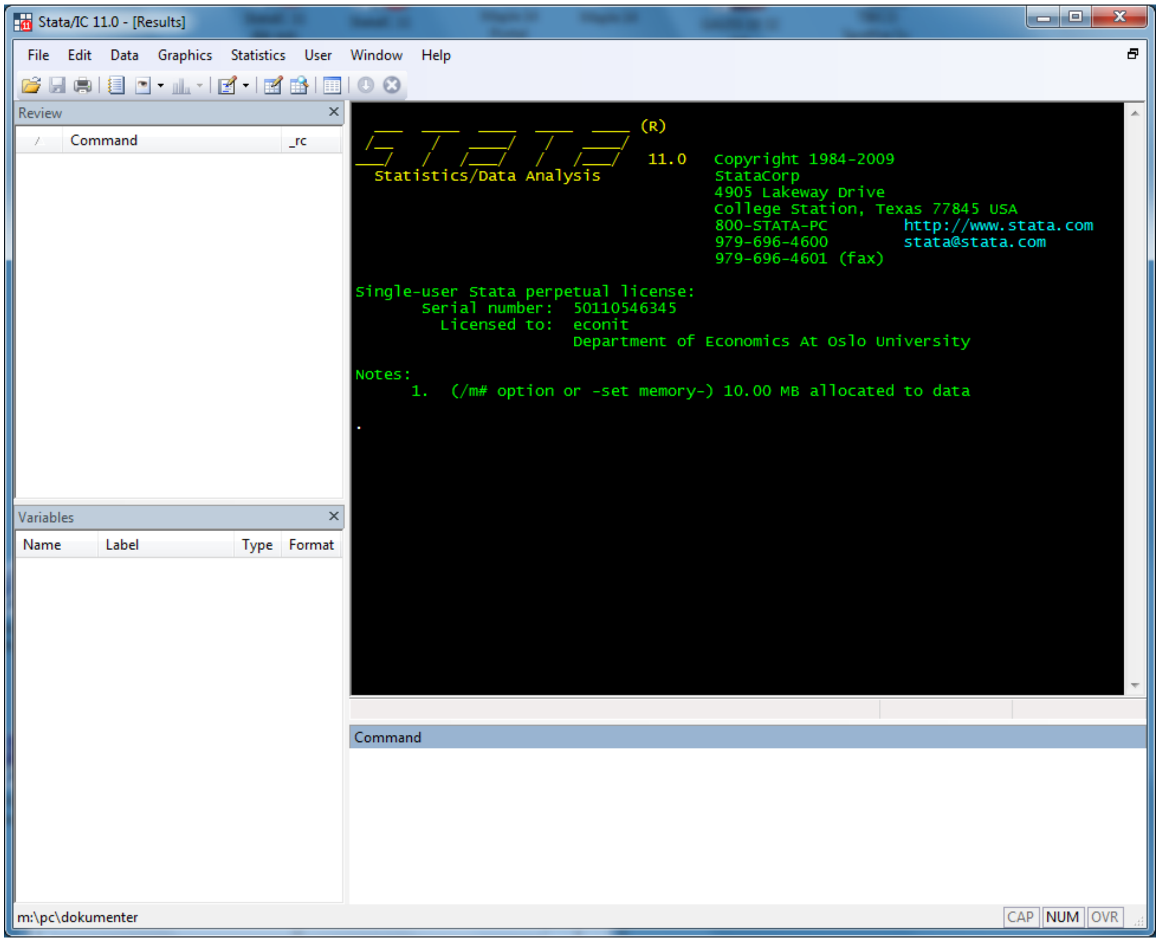
In addition, in contrast to earlier versions of Stata, a complete documentation of Stata in the form of a series of manuals - all collected in a single big *pdf*-file called *Stata 11 documentation* – can be found at the same place as the Stata program. It is a very good idea to open this *pdf* documentation in addition to the Stata program (normally you can open it via the Help-menu in Stata). It offers great help in exploring the Stata language and methods on your own. The documentation contains descriptions of methods and formulaes used and lots of examples. It is also easy to navigate using the bookmark system in in the documentation file.

The present tutorial is a very short introduction to some of the topics from the *Getting started* ([GS] -> [GSW]) manual in the pdf documentation file. Later you may find it useful to familiarize yourself with the *Getting started* [GSW] manual (and the *User* [U] manual) on your own as need arises.

[1] On some computers you do not need to enter *Communications*.

2. *The windows:*

STATA has 4 separate windows for typing in commands and for viewing results. The *command* window is for writing and running commands interactively. In the *review* window you can view (and activate) the command lines you have previously written. Clicking a command in the review window reproduces the command int command window for editing and rerunning. In the *variables* window all variables and labels are listed. Clicking on a variable name in the varaiables window reproduces that name in the command window.


Stata/IC 11.0 - [Results]
File Edit Data Graphics Statistics User Window Help
Review
Command
_rc
Variables
Name
Label
Type
Format
Command
m:\pc\dokumenter
CAP
NUM
OVR

```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   11.0   Copyright 1984-2009
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
                                      College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

Single-user Stata perpetual license:
       Serial number:  50110546345
         Licensed to:  econit
                       Department of Economics At Oslo University

Notes:
      1.  (/m# option or -set memory-) 10.00 MB allocated to data

.
```

When loading Stata the first time you probably get the Stata window in standard form – i.e. without any colors. By going to the menu[2]

edit -> preferences ->General preferences ->Result colors

[2] Or, shorter: simply right-click inside the result window and choose *preferences*…

you can choose color combinations for your output with different colors for different kinds of information – which makes the output easier to read (a matter of taste). In the figure I have chosen “Classic” which is one of the options.

Excercise; Load the exercise data, stored in a file called *auto.dta*, which is included in STATA. I.e., write *sysuse auto* (or *sysuse auto.dta*) in the command window and press enter (more details below).

You can obtain a description of the data set via the menu:
File -> Example datasets -> Example datasets installed with Stata
and then click the *describe* link outside *auto.dta*.

In any case you will see that the command line enters the review window and the results window (this illustrates how the menus can be used to learn the command lines. Learning the commands facilitates greater flexibility, quicker computing, and clearly a better understanding of how the program operates). For example, from the review window we see that the data description we obtained via the menus, we could have obtained directly by entering the command *sysdescribe auto.dta*. Clicking on that line reproduces the command in the command window. Pressing enter then runs the command again.

Note also that now all the variables in the data set are listed in the variables window.

3. More on how to load, save and use the built-in data set, auto.dta

To obtain a list of all built-in data sets that follow with STATA, you can use the command *sysuse dir*.

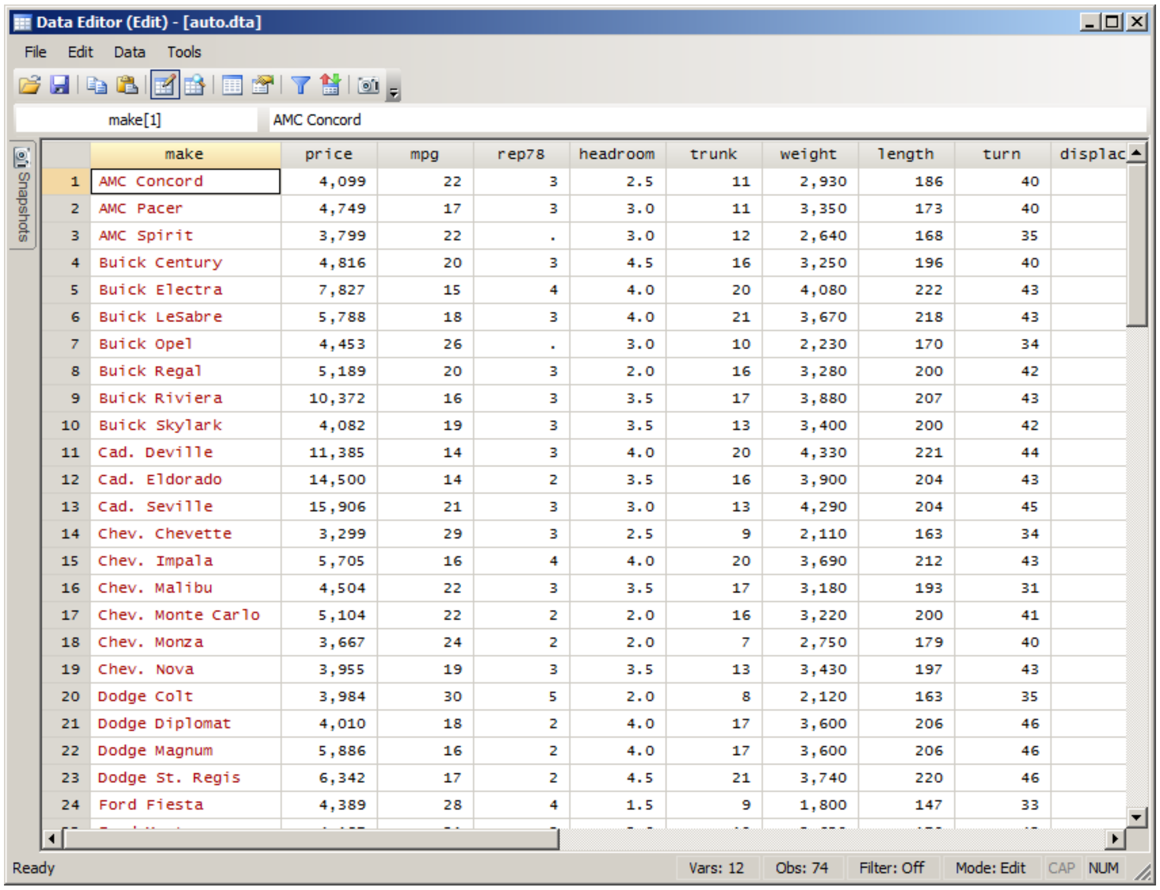
The command *dir* gives a listing of your own home directory (i.e., M:\).

You can save a copy of the data set (or any other data set) in your own directory by the command *save*, and later retrieve it by the command *use*. By *save* the data are stored in a binary file (with extension *.dta*) – in a special STAT format that is easy to read for STATA. For example, the command *save auto*, makes a copy of the data set, with name “auto.dta”, in your home directory. This is specially useful if you have made changes and additions to the data set during your STATA session. Then, next time you start STATA for a new session, you can simply give the command *use auto*, and the data are reloaded into STATA as you left them last time.

If you have many files in your home directory, it may be smart to make a new subdirectory and change the working directory to that one. For example, before starting STATA, suppose you have already made a subdirectory, called *statadat* (say), to contain all your stata files. Then, within STATA, the command *cd statadat* changes the working directory to M:\statadat, and commands like *save*, *dir*, *use*, etc. will refer to that directory (instead of to M:\ which is the default). The next time you open STATA, the first thing you should do, is to give the command *cd statadat*, (or with your chosen name for the directory) so that your working directory is the correct one.

4. The spreadsheet:

If you write *edit* or *browse* in the command box (or press the *edit* or *browse* icon), a spreadsheet window will pop up (there are also a menu and short cut buttons for opening the spreadsheet window). Note that if you used the *browse* command*, you can only view and not edit the spreadsheet.* To edit the data you need to use the *edit* command or press the edic icon.



make[1] AMC Concord

	make	price	mpg	rep78	headroom	trunk	weight	length	turn	displac
1	AMC Concord	4,099	22	3	2.5	11	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	
11	Cad. Deville	11,385	14	3	4.0	20	4,330	221	44	
12	Cad. Eldorado	14,500	14	2	3.5	16	3,900	204	43	
13	Cad. Seville	15,906	21	3	3.0	13	4,290	204	45	
14	Chev. Chevette	3,299	29	3	2.5	9	2,110	163	34	
15	Chev. Impala	5,705	16	4	4.0	20	3,690	212	43	
16	Chev. Malibu	4,504	22	3	3.5	17	3,180	193	31	
17	Chev. Monte Carlo	5,104	22	2	2.0	16	3,220	200	41	
18	Chev. Monza	3,667	24	2	2.0	7	2,750	179	40	
19	Chev. Nova	3,955	19	3	3.5	13	3,430	197	43	
20	Dodge Colt	3,984	30	5	2.0	8	2,120	163	35	
21	Dodge Diplomat	4,010	18	2	4.0	17	3,600	206	46	
22	Dodge Magnum	5,886	16	2	4.0	17	3,600	206	46	
23	Dodge St. Regis	6,342	17	2	4.5	21	3,740	220	46	
24	Ford Fiesta	4,389	28	4	1.5	9	1,800	147	33	

Ready Vars: 12 Obs: 74 Filter: Off Mode: Edit CAP NUM

If you right click on the variable names and choose "variable properties", you can edit the name, or the variable labels. (Alternatively you can edit the variable information by pressing the *variable manager* icon on the tool bar.)

You are also given information on *the format* of the variable. In the spreadsheet you have opened, clicking on the variable name "make" tells you that the label is "Make and model" and the format is "%-18s". The s indicates that the variable "make" is a string variable (consists of letters, not numbers), and that it will be stored using (maximum) 18 letters. The variable "price" has the different format "%8.0gc". Here, the letter "c" indicates that a comma is used to separate at the thousands, while “g” indicates that the variable is stored as an integer. If we change the format to read “%8.1fc”, the variable is no longer stored as an integer, but as a number on the real line where one decimal place is shown. If we edit it to “%8.2fc”, two decimal places are shown etc. Writing only “%8.2f” will take away the comma separation at the thousands.

A note on formats. Number variables can indeed be stored as string variables. This will often be the case when the data that are loaded are not originally in STATA format. When such data are loaded, it is therefore good practice to check whether the number variables are stored correctly.
In STATA you can refer to each variable by the variable name. You can also refer to the line number by using the reference “in”.

Important! *Note that you cannot run commands when either the “edit spreadsheet” window or the “browser” window is open. You have to close these windows before you can issue any commands.*

Exercise; Write the command *list make in 2*, and the command *list weight in 1/7*. What is returned in the results window?

5. The help facility:

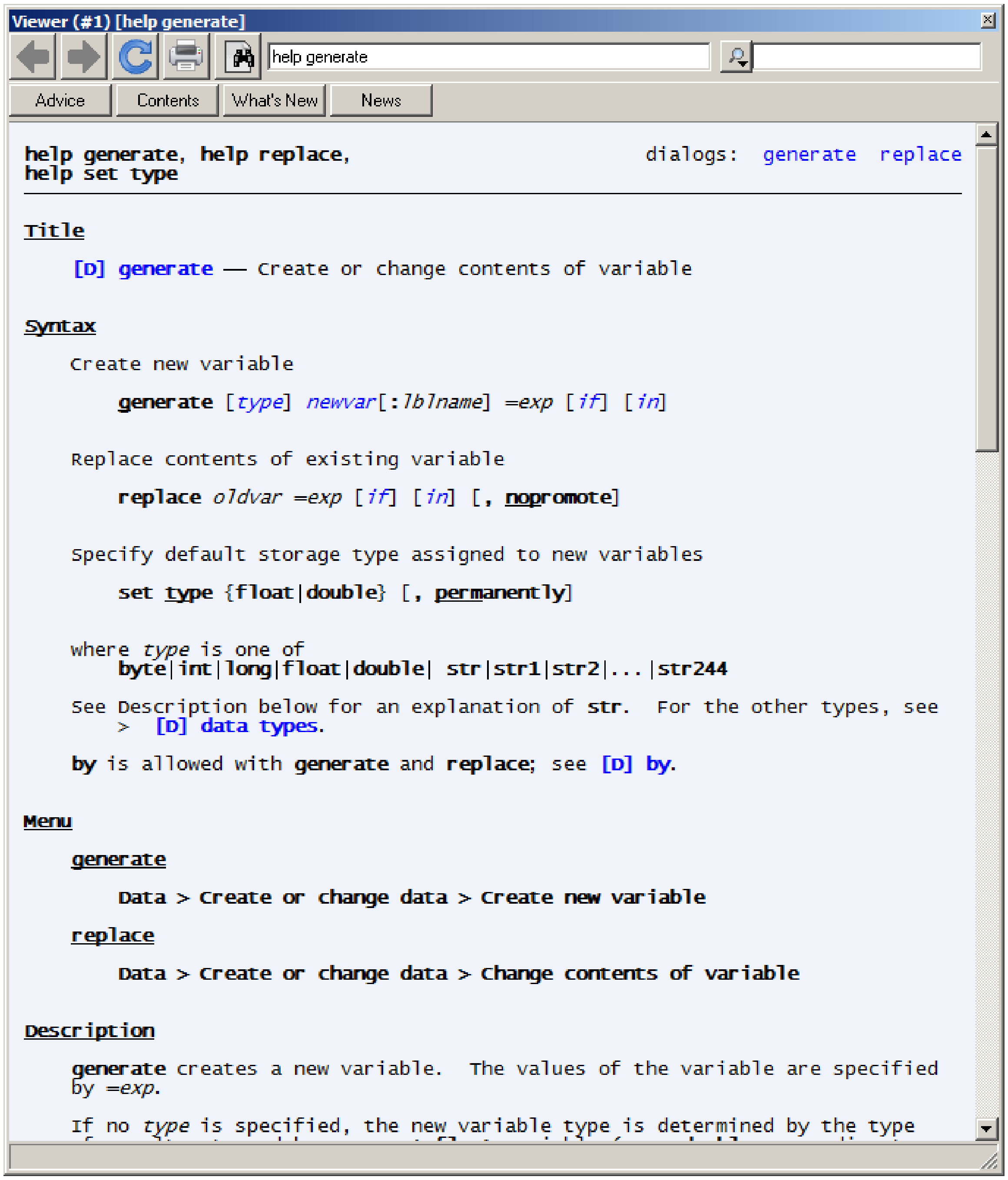
Suppose you want to use the *generate* command (used to create a new variable in the data set based on variables already there), and cannot quite remember how it is used. You can then type *help generate* in the command window. Then a new “viewer window” with the help appears (see below). (The “viewer” is an attached program that is designed to read STATA output files. It can read other files as well.) This window can be printed by specification on the *file* menu. The view editor can also be used to view and print contents of the results window. See “using log files”, later in this document.

The information given by the help command is quite detailed and concentrated, and can be hard to read by the beginner. Alternatively it may be easier to locate *generate* in the [D] manual, *Data management*, in the documentation pdf-file, and study the examples given there.

Or, by going via the menu to *help -> search…* and search for *generate*, you obtain much information on various sources where you can find information on *generate*.

You can read more on the help facilities in the [GSW] manual section 4.

The following figure shows the result of the command: *help generate*



help generate, help replace,
help set type

dialogs: generate replace

Title

[D] **generate** — Create or change contents of variable

Syntax

Create new variable

```
generate [type] newvar[:lblname] =exp [if] [in]
```

Replace contents of existing variable

```
replace oldvar =exp [if] [in] [, nopromote]
```

Specify default storage type assigned to new variables

```
set type {float|double} [, permanently]
```

where *type* is one of
byte|**int**|**long**|**float**|**double**| **str**|**str1**|**str2**|...|**str244**

See Description below for an explanation of **str**. For the other types, see
> [D] **data types**.

by is allowed with **generate** and **replace**; see [D] **by**.

Menu

generate

Data > Create or change data > Create new variable

replace

Data > Create or change data > Change contents of variable

Description

generate creates a new variable. The values of the variable are specified by *=exp*.

If no *type* is specified, the new variable type is determined by the type

6. The command syntax:

The command syntax is almost always of the general form:

[by *varlist*:] *command* [*varlist*] [if *exp*] [in *range*] [*,options*]

Words in italics are to be filled in by the user.

varlist refers to a list of variables, e.g. *mpg weight length price.*
exp refers to a logical expression
range refers to a range of line numbers
options, will depend on the command in question. The options must be specified at the end of the command line, *after a comma separator*. Anything following a comma is thus to be interpreted as one or several options to the command. If there are no options, the comma should not be present.

The brackets indicate that specification within the brackets is optional for the command. For example, the [by *varlist*:] formulation is optional (i.e. is not required) and specifies that the command is to be repeated for each variable in the variable list. Not all commands can use this formulation.

The command syntax is best illustrated by a few simple examples:

EXAMPLE; In the tutorial dataset we may want to construct a new variable that equals mpg/weight. The command *generate* (or *gen* for short) is one of the commands to create new variables. Writing *help generate* in the command window returns the following syntax from the help window.

generate [type] *newvar*[:*lblname*] = *exp* [if *exp*] [in *range*]

Here the command name (*generate)*, the name of the new variable to be generated (*newvar*) chosen by you, and the function that describes how the new variable is to be constructed (*=exp*) has to be specified. The help text explains that [type] has to be specified only if the variable that you want to create is to become a string variable, or if it is important to specify the decimal precision of the new variable. If a string variable is to be generated type can be specified to *str10* showing that the variable is to be stored with 10 letters. If a number variable that is generated has to have decimal precision type can be specified to *double*. The :lbname formulation is optional and allows you to specify a variable label that describes the content of the new variable.

To generate the new variable (that we for example call x) we type

```
generate x = mpg/weight                    (or shorter:  gen x=mpg/weight)
```

If you want to change the content of an existing variable in the data base, you can use the *replace* command:

replace *oldvar* = *exp* [if *exp*] [in *range*] [, nopromote]

Exercise. Do this and check in the browser (or edit) window that a new variable has been created. Remember to close the data window before you can proceed.

Exercise; You can use the help function to establish what the following commands does; (these are some must-to-know STATA commands). You do not need to go through all of them now, but try, e.g., "list" and "summarize". Try also to execute the command, *summarize x*.

save	label	drop	merge
correlate	describe	keep	collapse
summarize	list	regress	test
tabulate	count	egen	predict
sort	mark	rename	clear

7. Logical expressions:

If you decide to use the optional [if *exp*] specification you must use a special syntax for logical operators.

== equals to (note that we use 2 equality signs)
~= not equal to
>= larger than or equal to, etc..
> larger than
< less than
& and
| or

EXAMPLE (do this)

```
tabulate make rep78 if foreign==1
tabulate make rep78 if foreign==1& price<4000
tabulate make rep78 if foreign==1| price<4000
```

How many different makes did you get in each of the three cases?

Note 1: Note (in the *browse* or *edit* window) that the variable "foreign" has two values, 1 (with label "Foreign") and 0 (with label "Domestic"). The actual values, 1 and 0, are stored, but the labels "Foreign" and "Domestic" are displayed in the data base. If you click one value (i.e. one of the "Foreign"s), you will see the corresponding numerical value in the small window at the top of the data base window. The command, *label list*, will give a list of labels defined. You can learn how to define labels in your data set by *help label*.

Note 2: *Remember to close the data window before issuing any command.*

Note 3: When you need to write a command that is similar to a command you have used before that is shown in the review window, then often the easiest way is to click on the command in the review window. This causes the command to appear in the command window where you can edit it before running it. Similarly, if you click on a variable name in the variables window, the variable name pops up in the command window where the cursor is.

8. Graphics

The graphics facility in STATA is quite well developed and allows numerous variations and editing facilities. For a start it is recommended to experiment with the graphics menu. You can then note the syntax that is automatically written in the results window. Use the auto dataset. Make a histogram over price using 10 bins (Command: *histogram price, bin(10)*). You can also use the menu, *Graphics -> Histogram*. Compare box-plots of foreign and domestic cars (*graph box price, medtype(line) over(foreign)*). Draw a scatter diagram of miles pr. gallon and weight (*twoway (scatter mpg weight)*). Try also to reproduce these three graphs by using the *graphics* – menu.

9. Linear regressions:

To fit simple or multiple linear regressions, use the *regress* command (or by the menu: *statistics -> linear regression ….*). Using the auto data, *generate variable x=mpg/weight* (if you have not done it before) and type:

regress price mpg weight x foreign

Exercise; Interpret the estimated model[3]. Check out the syntax for the *predict* command used after the *regress* command and use it to obtain the predicted price, predicted residuals and squared residuals. [e.g. *predict predy* produces a new variable in the database containing the predicted values. It is here called *predy*, or another name that you choose. The command *predict res , residuals* produces a variable, containing the residuals (i.e. observed price minus predicted price), with name *res*, or another name of your choice.]

[3] In regression analysis we try to explain a dependent variable (price) by a linear combination of some explanatory variables (mpg, weight, x, foreign). In other words:

$$\text{price} = a_0 + a_1 \cdot \text{mpg} + a_2 \cdot \text{weight} + a_3 \cdot \text{x} + a_4 \cdot \text{foreign} + \text{error}$$

where the constants $a_0, a_1, \ldots, a_4$, which in the output from Stata appear under "Coef", are determined (estimated) by the program from the data in an optimal manner so that the last term "error" (i.e. the unexplained part of price - also called residual) becomes as small as possible on average (in a certain sense). The linear combination (before the error term) represents the explained part of price and is called predicted price. If the model is good, the error term should be evenly distributed around zero (i.e. have constant variance) whatever the value of predicted price. If that is not the case, we call the model *heteroscedastic*.

Can you use the *scatter* command, or the *graphics->Two way graphics (scatterplot, line etc)* menu to assess whether the model specification is likely to be heteroscedastic (e.g. plot the residuals against the predicted price)? Use the help facility to list the functions library. Generate a variable that equals the natural logarithm of the price and re-estimate the model. How would you interpret the model now?

10. Using log-files:

This facility allows you to print or save all commands you used with the output produced in the result window for a session with STATA. It is particularly useful when you hand in written papers in class, so that the teacher can see how you obtained your results. To start logging a session, type *log using sessionname*, where sessionname is the name you decide for the session. This logging you can turn off and on again temporarily during the session by the commands *log off* and *log on*. When the session is completed,however, type; *log close*. In the results window you will now be told where the log file is saved. When you want to view or print the log file you type; view address\sessionname.smcl.

Note. "smcl" is the program STATA uses to produce output files (see *help smcl* for more information). It produces files with the extension smcl. You can copy content in the viewer to a word document in the usual manner (i.e., marking, copy and paste). If you want the log-file in the usual text format, you must add the option "text" to the *log using* – command, i.e.,

log using sessionname, *text*   (don't forget the comma).

If you wish to log only the commands you have used in a session for use in a new and later session, you can do that by the command *cmdlog*, or, if you late in a session think it would be useful to save all the commands in the review window for a later session, simply right-click on the review window and save the list of commands in a do-file (see section 13 below). For more information on this read page 118 in the *Getting started* manual [GSW] under [GS] in the documentation pdf-file.

11. Num(ber)lists:

Often you will find reference to *numlist* in the STATA syntax description. *Numlist* is simply a sequence of numbers, which can be specified in various ways. As an example; the sequence 2 4 6 8 10 and the numlist 2(2)10 will be synonymous to STATA. To get an overview of different ways to specify numlists, type *help numlist*.

12. Make patterned/random data

Input the following lines and figure out what they do.

Command	Notes
`clear`	
`browse`	Close the browse window to get back to the command level.
`Set obs 100`	
`browse`	
`egen year = fill(1900 1901)`	"egen" is an extended version of "generate" that we need for defining new variables, e.g. consisting of patterned data and other types.
`browse`	
`egen trend = fill(0.1(0.1)10)`	
`browse`	
`generate a = sin(trend)`	
`browse`	
`generate cycle=trend+a`	
`twoway (line trend cycle year)`	
`set seed ?`	Replace ? by an integer of your choice, e.g. your birthday like for example 100786. This starts the algorithm for generating random data. By using the same seed you can produce the same data later. If seed is not specified, stata will choose a seed by default which changes every time you draw random numbers.
`drawnorm u`	
`generate gdp = cycle+u`	
`twoway (line trend cycle gdp year)`	

Exercise; Load the auto dataset. Explain why this sequence of commands can be used to draw a random sample of 20 cars:

```
gen u = uniform()
sort u
mark sample in 1/20
```

13. The do-file editor

Often you will need to type a sequence of commands several times. In this case you should use the do-file editor (press the short cut icon with a picture of an envelope). In the do-file you can write in multiple lines and run them in a sequence. You can save the do-file for later use. Often you will want to specify loops in the do-file editor. As an example, suppose you have variables; year1, year2, year3, … , year100, and that you want to transform these variables from variables with string values to variables real numbers as values. You can then type (*don't do this here* unless you actually have 100 string variables called year1, year2,… defined in the data base!)

```
forvalues x = 1/100 {
   generate y`x' = real(year`x')
   }
```

STATA will then perform this command successively for \`x' running from 1 to 100. Note that all the definitions for *numlist* can be used with this command. The command will generate new variables, y1, y2, …., y100 that represent nummeric versions of year1, year2,…,year100.

To test out a loop, try the following command in a do-file. (For the *di* command see below. Note also that the single quotes in \`x' are different. The first one I find on my keyboard on the top of the back-slash (\) key, and the last one on my *-key.).

```
forvalues x = 2/20 {
di "I will do `x' attempts to do my homework properly"
}
```

To execute the commands in the do-file directly from the do-file editor, push the execute icon or use the menu with *tools -> execute (do)* . If you want to use commands in the do-file in a later session, you can save the do-file under some name, *exmpldo.do,* for example, and load into the do-file editor in a later session.

Note: This small program, consisting of only one command, you could also have entered directly in the command window. Note that the three lines must be entered separately: After the first line press enter. Then a number, 2, appears waiting for the second line and so on. The last line consisting of the end braceonly, closes the program input and runs the program. (Note that the single quotes in \`x' are different. The first one I find on my keyboard on the top of the back-slash (\) key, and the last one on my *-key.)

14. Calculator

di is short for the *display* command. *display* is used for printing strings or scalar numbers. It can be used as a calculator. Try out the following (-> denotes output):

You want the value of *e*:

```
di exp(1)
-> 2.7182818
```

You want higher precision (10 decimal places):

```
di %12.10f exp(1)
-> 2.7182818285
```

You want to describe the output, you can add a comment in double quotes:

```
di "e = " exp(1)
```

You want to calculate $\sqrt{2\pi}$ (π in STATA is _pi):

```
di  sqrt(2*_pi)
```

15. Loading data in ASCII format:

If data is in ASCII format, you cannot use the *use* command. Then use instead the *insheet* command. You can check out the syntax for *insheet* using the help facility.

16. Exercise – calculating gamma probabilities

a. STATA has implemented the cdf for the $\Gamma(\alpha,1)$-distribution, i.e., the function *gammap(a,x)*. [See *help probfun* for a list of probability functions implemented, and *help function* for a list of functions in general. More information can be found in the *Data manual* [D] -> *functions*]. For example, *di gammap*(2, 1.5) gives .4421746, which is the probability $P(T \le 1.5)$ where $T \sim \Gamma(2,1)$ (i.e. $\alpha = 2$)[4].

Now, make a table of $P(T \le t+1 \mid T > t)$ for $t = 0.5,\ 1,\ 1.5,\ 2,\ 2.5,\ 3,\ 3.5$ and for $\alpha = 0.5,\ 1,\ 2$ (with $\lambda = 1$ always). Note that writing $F(t)$ for the cdf of *T*, we get

$$P(T \le t+1 \mid T > t) = 1 - P(T > t+1 \mid T > t) = 1 - \frac{P(T > t+1 \cap T > t)}{P(T > t)} = 1 - \frac{P(T > t+1)}{P(T > t)}$$

Hence

[4] Note that *gammap()* assumes that $\lambda = 1$ in the $\Gamma(\alpha,\lambda)$-distribution. If you need to caculate $P(T \le t)$ when $\lambda \ne 1$, use that $\lambda T \sim \Gamma(\alpha,1)$ (see e.g. Rice exercize 2.61), which implies $P(T \le t) = gammap(\alpha, \lambda t)$ in Stata. For example, if $T \sim \Gamma(2,2)$, then you get $P(T \le 1.5) = 0.8008...$ using the command *di gammap(2, 3)*. **Note also** that when STATA talks of $\Gamma(a,b)$, it means that *a* is the same as our α, but b means $1/\lambda$. So, e.g., that our $\Gamma(2,2)$ - distribution corresponds to STATA's $\Gamma(2, 0.5)$ - distribution.

$$P(T \le t+1 \mid T > t) = 1 - \frac{1-F(t+1)}{1-F(t)} = \frac{F(t+1)-F(t)}{1-F(t)}$$

[**Hint:** First empty STATA of all data by the *clear* command. Then open the data spreadsheet e.g. by the data editor icon. Enter the data for *t* (0.5, 1, …., 3.5) in the first column. I.e., start with writing 0.5. Note that it appears in the little window at the top. Press enter (or the ↓ arrow). Now, the number is entered in the first cell of column 1 (with name var1). Write the second number, 1, and press enter and so on. Change the name of the first column from var1 to, for example, t, by right- clicking on the head of the column and *variable properties*, and then write in the new name.
After thus having entered the numbers for *t*, generate a new column named e.g. y1 with $P(T \le t+1 \mid T > t)$ for $\alpha = 0.5$. This can be done by the *gen* command:

```
gen y1 = (gammap(.5,t+1) – gammap(.5,t))/(1 – gammap(.5,t))
```

Repeat this for α equal to 1 and to 2.

Note that you now can save the results in a Stata data set by the *save* command. Say you want to name the dataset "gam", for example. Then use the command *save gam*. The table is then saved as a Stata data set in your working directory under the name gam.dta. You can easily retrieve the data in a later session (using the same working directory; see section 3), simply by the command *use gam*.]

b. Suppose we want to calculate $P(T \le 4)$ when $T \sim \Gamma(\alpha = 2,\ \lambda = 3.2)$, with λ different from 1. In order to use gammap for this we must transform *T* to $\Gamma(\alpha, 1)$. According to the solution of Rice exercise 2:61, $Y = \lambda T \sim \Gamma(\alpha, 1)$. Hence, for $\lambda = 3.2$ we get

$P(T \le 4) = P(3.2T \le (3.2)4) = P(Y \le 12.8)$
Then you can use gammap. (Confirm that the answer is 0.9999619).