

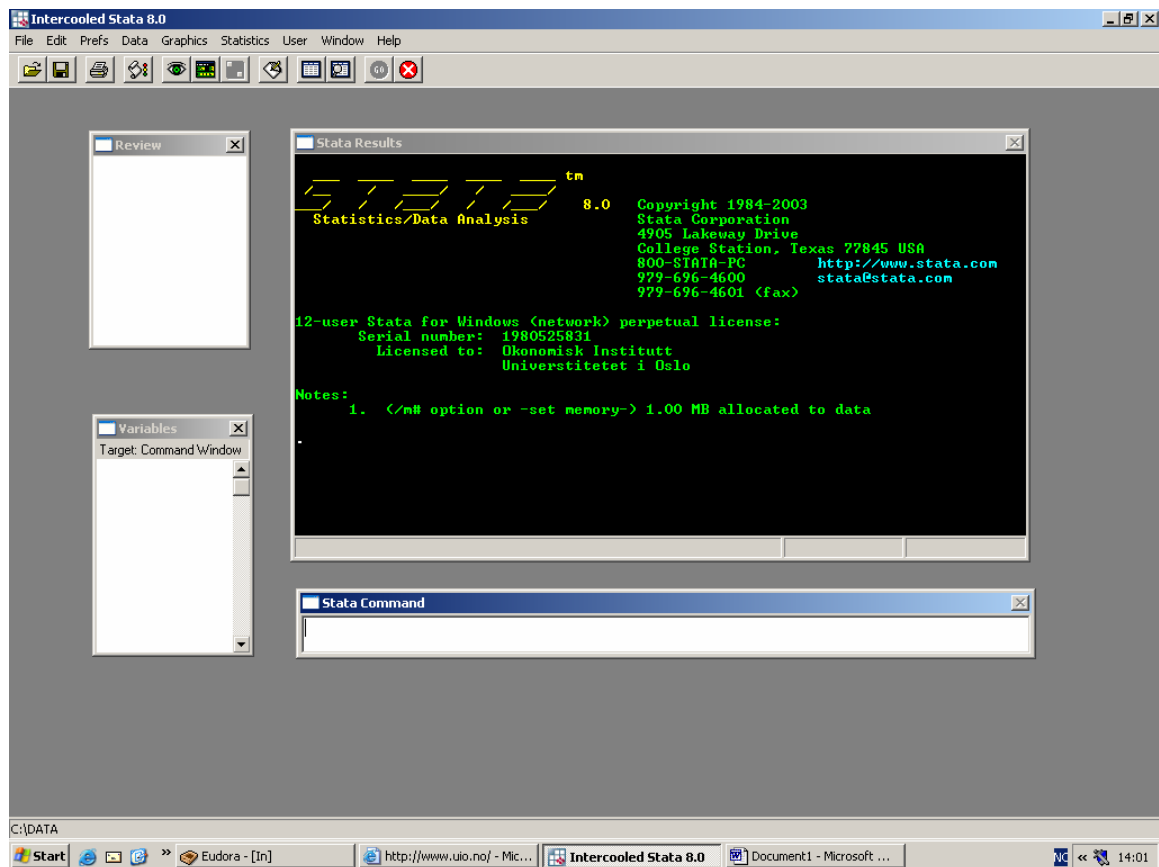
September 2004

A quick introduction to STATA:

(by E. Bernhardsen and some minor additions by H. Goldstein)

The windows:

STATA has separate windows for typing in commands and for viewing results. In the review window can view (and activate) the command lines you have previously written. In the variables window all variables and labels are listed.



Exercise; Load the exercise data. Write `use C:\Stata8\auto.dta` in the command window and press enter. Alternatively, press the open file menu, navigate to `C:\Stata8\auto.dta` and press OK. 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).

The spreadsheet:

If you write *edit* or *browse* in the command box, you the spreadsheet window will pop up (there are also a menu and short cut buttons for opening the spreadsheet window). If you used the browse command, you can only view and not edit the spreadsheet.

The screenshot shows the Stata 8.0 spreadsheet editor window. The title bar reads "Intercooled Stata 8.0". The menu bar includes File, Edit, Prefs, Data, Graphics, Statistics, User, Window, and Help. The toolbar contains icons for file operations and editing. The main window displays a spreadsheet with the following data:

	make	price	mpg	rep78	headroom	trunk	weight	length
1	AMC Concord	4,099	22	3	2.5	11	2,930	18
2	AMC Pacer	4,749	17	3	3.0	11	3,350	17
3	AMC Spirit	3,799	22	.	3.0	12	2,640	16
4	Buick Century	4,816	20	3	4.5	16	3,250	19
5	Buick Electra	7,827	15	4	4.0	20	4,080	22
6	Buick LeSabre	5,788	18	3	4.0	21	3,670	21
7	Buick Opel	4,453	26	.	3.0	10	2,230	17
8	Buick Regal	5,189	20	3	2.0	16	3,280	20
9	Buick Riviera	10,372	16	3	3.5	17	3,880	20
10	Buick Skylark	4,082	19	3	3.5	13	3,400	20
11	Cad. Deville	11,385	14	3	4.0	20	4,330	22
12	Cad. Eldorado	14,500	14	2	3.5	16	3,900	20
13	Cad. Seville	15,906	21	3	3.0	13	4,290	20
14	Chev. Chevette	3,299	29	3	2.5	9	2,110	16
15	Chev. Impala	5,705	16	4	4.0	20	3,690	21
16	Chev. Malibu	4,504	22	3	3.5	17	3,180	19
17	Chev. Monte Carlo	5,104	22	2	2.0	16	3,220	20
18	Chev. Monza	3,667	24	2	2.0	7	2,750	17
19	Chev. Nova	3,955	19	3	3.5	13	3,430	19
20	Dodge Colt	3,984	30	5	2.0	8	2,120	16
21	Dodge Diplomat	4,010	18	2	4.0	17	3,600	20
22	Dodge Magnum	5,886	16	2	4.0	17	3,600	20
23	Dodge St. Regis	6,342	17	2	4.5	21	3,740	22
24	Ford Fiesta	4,389	28	4	1.5	9	1,800	16
25	Ford Mustang	4,187	21	3	2.0	10	2,650	17
26	Linc. Continental	11,497	12	3	3.5	22	4,840	22
27	Linc. Mark V	13,594	12	3	2.5	18	4,720	22
28	Linc. Versailles	13,466	14	3	3.5	15	3,830	20

The spreadsheet editor also includes a "Review" window showing the command "use 'C:\Stata8\auto.dta', cl edit" and a "Variables" window listing the variables: make, price, mpg, rep78, headroom, trunk, weight, length, turn, displacement, and gear_ratio.

If you double click the variable names you can edit the name, or the variable labels. 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 is 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 is loaded is not originally in STATA format. When such data is 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”. **Exercise;** Write the commands *list make in 2*, *list weight in 1/7*. What is returned in the results window?

The help facility:

Suppose you want to use the *generate* command, and cannot quite remember how it is used. You can then type *help generate* in the command window:

```

help for generate, replace, seed          manual: [R] generate
                                       dialogs: generate replace

Create or change contents of variable

generate [type] newvar(=lblname) = exp [if exp] [in range]
replace oldvar = exp [if exp] [in range] [, nopromote]

set seed < # | code >
set type < float | double >

where type is one of
byte | int | long | float | double | str | str1 | str2 | ... | str#0

See Description below for an explanation of str. For the other types, see help
datatypes.

by ... : may be used with generate and replace; see help by.

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 of
result returned by = exp. A float variable (or a double according to set type)
is created if the result is numeric and a string variable is created if the
result is a string. In the latter case, a str# variable is created where # is
the smallest size string that will hold the result.

If a type is specified, then the result returned by = exp must be string or
numeric according to whether type is string or numeric. If str is specified, a
str# variable is created where # is the smallest size string that will hold the
result.

See help egen for extensions to generate.

```

By clicking on –more- or just hitting the space bar, you will scroll down the windows. Alternatively you can type in *generate* in the help menu dialog box, or type *view help generate* in the command window. In any case you will see the help information in a separate window which is called the *view editor*. 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 command syntax:

The command syntax is almost always on the general form:

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

Where:

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.

The brackets indicate that specification is optional. The [by *varlist*:] formulation is optional 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*. Writing `help generate` in the command window returns the following syntax from the results 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*) 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* if 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 *:lblname* formulation is optional and allows you to specify a variable label that describes the content of the new variable.

To generate the new variable we type

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

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

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

Exercise; Use the help function to establish what the following commands does; (these are must-to-know STATA commands). Try e.g. “list”.

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

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*.

Logical expressions:

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

== equals to
 ~= 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: Note (in *browse*) 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*.

Graphics

The graphics facility in STATA is quite well developed and allows numerous variations. 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 (*histogram price, bin(10)*). Compare box-plots of foreign and domestic cars (*graph box price, medtype(line) over(price)*). 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.

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 $x = \text{mpg}/\text{weight}$ and type:

```
regress price mpg weight x foreign
```

Exercise; Interpret the estimated model. 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. Can you use the *scatter* command, or the *graphics-easygraph-scatterplot* 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?

Using log-files:

This facility allows you to print or save all commands you used 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. When the session is completed, 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*.

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 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 100781. 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
```

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 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 string to real numbers. You can then type

```
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.

To test out a loop, try the following command (you can use the command window for this one-liner)

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

Note: *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:

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

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

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

Loading data in ASCII format:

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