

Revised September 2015

A quick introduction to STATA:

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

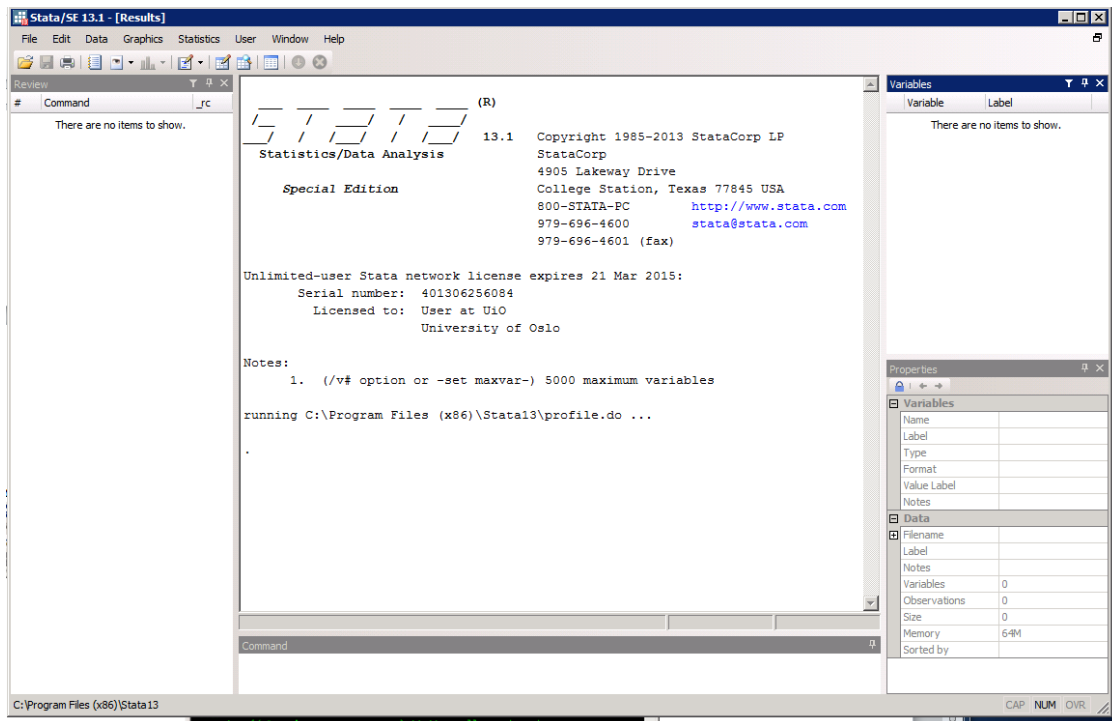
1. How to access STATA from the pc's at the computer lab and elsewhere

Log in at <https://kiosk.uio.no/> and find STATA (SE) under *Analyse*.

2. The windows:

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

When opening Stata you get



Exercise 1; 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*.

To look at the data, click, e.g., the browse icon to get the browse window. In the browse window you cannot change anything (it is read-only). If you want to edit the data, click on the edit icon instead.

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 working directory (i.e., *M:\stata\dat* if that is the name).

You can save a copy of the data set (or any other data set) in your own working 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 STATA 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 working directory. This is especially 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, after having changed the working directory, you can simply give the command *use auto*, and the data are reloaded into STATA as you left them last time.

4. The spreadsheet:

If you write *edit* or *browse* in the command box (or click the edit or browse icons), a spreadsheet window will pop up. If you used the browse command, you can only view and not edit the spreadsheet.

The screenshot shows the STATA Data Editor window with a spreadsheet of car data. The spreadsheet has columns for 'make', 'price', 'mpg', 'rep78', 'headroom', 'trunk', 'weight', 'length', 'turn', 'displacement', and 'gear'. The 'make' column contains car models like 'AMC Concord', 'AMC Pacer', 'AMC Spirit', etc. The 'price' column contains values like 4,099, 4,749, 3,799, etc. The 'mpg' column contains values like 22, 17, 22, etc. The 'rep78' column contains values like 3, 3, ., etc. The 'headroom' column contains values like 2.5, 3.0, 3.0, etc. The 'trunk' column contains values like 11, 11, 12, etc. The 'weight' column contains values like 2,930, 3,350, 2,640, etc. The 'length' column contains values like 186, 173, 168, etc. The 'turn' column contains values like 40, 40, 35, etc. The 'displacement' column contains values like 121, 258, 121, etc. The 'gear' column contains values like 4, 4, 4, etc.

The Variables panel on the right shows a list of variables with their labels and formats. The 'make' variable is highlighted, showing its label as 'Make and Model' and its format as 'str 18'. The Properties panel shows the variable 'make' with its label 'Make and Model', type 'str 18', and format '%-18s'. The Data panel shows the filename 'auto.dta', label '1978 Automobile Data', and other statistics like 'Variables: 12', 'Observations: 74', 'Size: 3.11K', and 'Memory: 64M'.

Given that you are in the edit window, in the windows on the right you can edit the name, the variable labels, or even the format of the variable. In the spreadsheet you have opened (the Edit window), click the variable “make” in the upper right window. Then you will see (in the lower right window) that the label is “Make and model” and the format is “%-18s” – which can be edited there. 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.

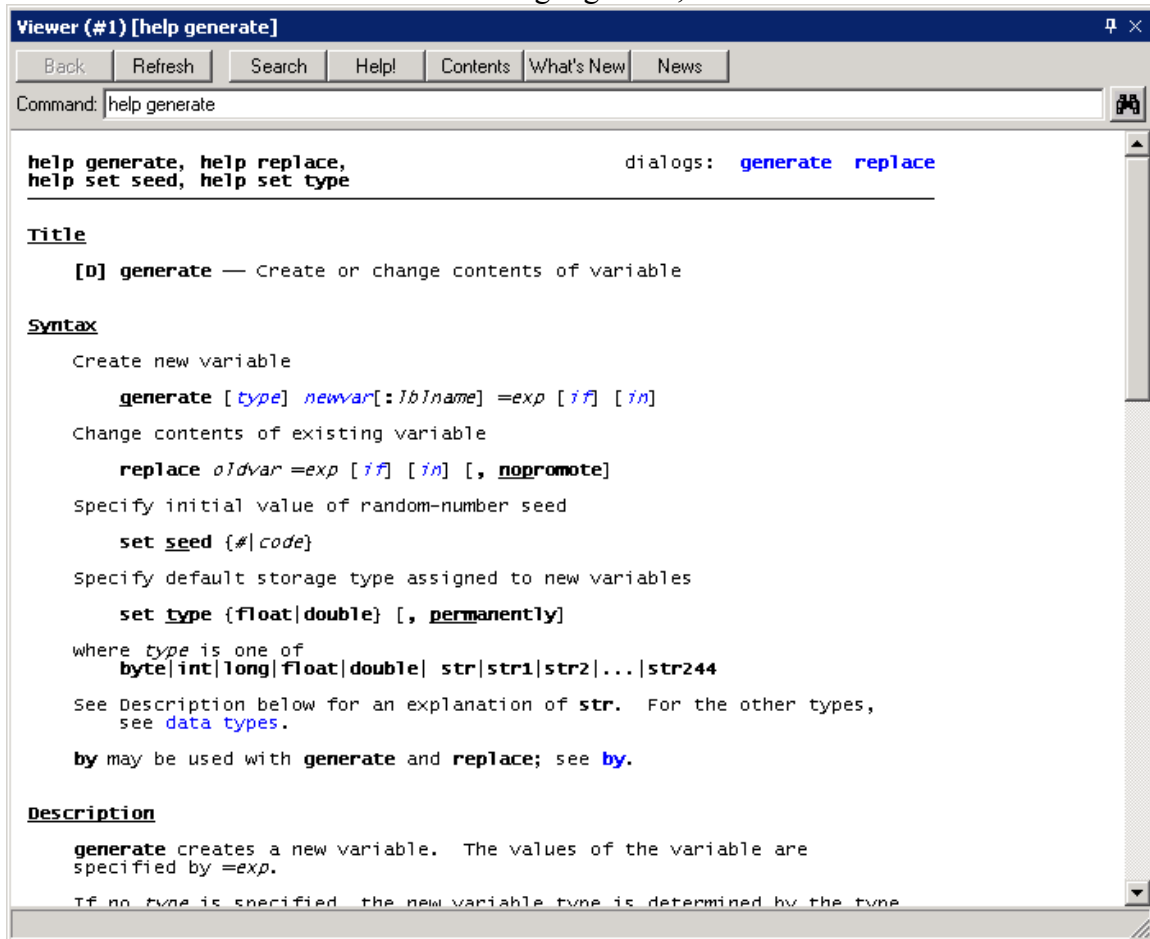
A note on formats; number variables can indeed be stored as string variables. This will often be the case when the data that is loaded is not originally in STATA format. When such data is loaded, it is therefore good practice to check whether the number variables are stored correctly. *Note that numbers stored as strings get the color red in the data window.*

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” as in exercise 2.

Exercise 2; 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, 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. (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 screenshot shows a window titled "Viewer (#1) [help generate]". The command entered is "help generate". The help text is as follows:

```

help generate, help replace,          dialogs:  generate  replace
help set seed, help set type

Title
  [0] generate — Create or change contents of variable

Syntax
  Create new variable
    generate [type] newvar[:lbname] =exp [if] [in]
  Change contents of existing variable
    replace oldvar =exp [if] [in] [, nopromote]
  Specify initial value of random-number seed
    set seed {#|code}
  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 data types.
  by may be used with generate and replace; see 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
  
```

The information you get from the help function is somewhat condense and sometimes hard to understand when you are a beginner. In *Help -> PDF Documentation* on the menu you will find complete documentation written in understandable terms with lots of

examples and descriptions of methods used. Much can be learned about statistics/econometrics by reading in that documentation. There are also a lot of stuff about Stata on the web (can be accessed from the Stata –help menu) – examples, programs, data, etc.

6. *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*. A comma in a command always means that what follows is a list of options to the command!

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- or short: gen*), 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 3; You can use the help function to establish what the following commands does; (these are must-to-know STATA commands). Don't do all that now, but try, as an example, e.g. "list". A more complete and readable description you can find in the PDF-Documentation mentioned above.

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

7. 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*.

8. 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 1: 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*.

Note 2: In the *review window* called Commands you find a list of earlier commands you have used. If you wish to rerun one of those, just click on the command and it will show up in the command window. There it can be easily edited in the usual way if you wish.

Note 3: Another useful feature when writing commands is the *variables window* which lists all variables used in the session. When you need the name of a variable in a command, just click on the name in the variable list and the name will show up in the command at the place where the cursor is.

9. 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 (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.

Note that in *help twoway* you can learn how to make overlaid graphs (i.e., several graphs put on the top of each other), for example you could draw a density function on the top of an histogram to assess graphically the goodness of fit of a probability model.

To make a graph of a function, $y = f(x)$, use¹ the command `twoway function`. E.g. make a graph of the pdf in the $N(3, 4)$ distribution (i.e. where $\mu = 3$ and $\sigma^2 = 4$), with pdf

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2 \cdot 4}(x-3)^2} \quad \text{for } -\infty < x < \infty$$

A plot of this (do it!) in the range (-4, 10) is achieved by (note that the constant Pi is specified as `_pi` in Stata)

```
twoway function y=exp(-(x-3)^2/8)/sqrt(2*_pi), range(-4 10)
```

¹ You get more information in: `help twoway function`

10. 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` and type:

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

Exercise 4; Interpret the estimated model². 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.]

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?

11. Using log-files:

This facility allows you to print or save all commands you used for a session with STATA. To start logging a session, type *log using sessionname* (or open the log via *file..* on the menu), 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* (or use *view...* on the file menu).

Note. “smcl” is the program STATA uses to produce output files, also called “review 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.,

² 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:

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

where the constants a_0, a_1, \dots, 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*.

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

As a shortcut use the menu, *File -> Log -> Begin*, to open a new log-file.

12. Make patterned/random data

Input the following lines and figure out what they do.

Command	Notes
<code>clear</code>	To empty STATA of all data and variables before a new session.
<code>browse</code> (or click the browse icon)	Close the browse window to get back to the command level.
<code>Set obs 100</code>	
<code>browse</code>	
<code>egen year = fill(1900 1901)</code>	"egen" is an extended version of "generate" that we need for defining new variables, e.g. consisting of patterned data and other types.
<code>browse</code>	
<code>egen trend = fill(0.1(0.1)10)</code>	
<code>browse</code>	
<code>generate a = sin(trend)</code>	
<code>browse</code>	
<code>generate cycle=trend+a</code>	
<code>twoway (line trend cycle year)</code>	
<code>set seed ?</code>	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.
<code>drawnorm u</code>	
<code>generate gdp = cycle+u</code>	
<code>twoway (line trend cycle gdp year)</code>	

Exercise 5; 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 string to real numbers. You can then type (don't do this now unless you 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.

To test out a loop, try the following command in a do-file (for the *di* command see below).

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

Note1: This small program you can also enter 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 (the end brace) 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.)

Note 2: Sometimes after a session you would like to save the commands you have used for a later session. You can do this directly by right-click on the review window for commands and go to save... You can also load the commands you have used into the do-file editor: First right-click the review-window and *select all*. Then right-click again on the marked area and go to *Send to do-file editor*. There you can edit the commands – e.g., removing commands you don't want – and save the do-file for later use.

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:

```
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 or from an Excel-file:

- 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.
- If you have data organized as columns in an Excel-file, it is easy to get them into Stata. (1) If the data set is not too large, you can simply use copy of the Excel-data (mark all columns with data including the variable names in the first row) and paste them into the edit data window by pointing at one of the columns. (2) If the data set is large, you can also use the import function in Stata from the menu: *File -> import -> ODBC data source*.
- You can also easily copy data from the Stata data window and paste them into Excel in the usual manner.

Exercise 6³

- 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.]. For example, *di gammap(2, 1.5)* gives .4421746, which is the probability $P(T \leq 1.5)$ where $T \sim \Gamma(2, 1)$ (i.e. $\alpha = 2$).

Now, make a table of $P(T \leq t+1 | T > t)$ 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

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

Hence

³ This exercise is about the gamma distribution. If you have not seen this before, you may jump to exercise 7 and 8, and do exercise 6 later when you have learned about the gamma distribution.

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

[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. 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 (in the edit data base) marking the variable name in the upper right window and edit the name in the lower right window.

After thus having entered the numbers for t , generate a new column named e.g. y1 with $P(T \leq t+1 | T > t)$ 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))
```

[Note. If you copy this command and paste it into the command window, the command will not work. This is because the two minus-signs are not copied correctly from a pdf file. So: first copy and paste the command in the command window. Then remove the two minus-signs and replace them with minus-signs from the keyboard. Then the command should work.]

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 $P(T \leq 4)$ 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

$$P(T \leq 4) = P(3.2T \leq (3.2)4) = P(Y \leq 12.8)$$

Then you can use *gammap*. (Confirm that the answer is 0.9999619).

Exercise 7

- a. Stata is good at numerical integration (using the *integ* command). Suppose you need the

value of the integral $\int_1^3 f(x) dx = \int_1^3 \frac{e^{-(x-2)^2}}{x} dx$, which is a difficult (transcendent) integral.

Numerical integration consists of dividing the integration interval ((1, 3) here) into many equally spaced points, calculate all the $y = f(x)$ values, joining the points (x, y) by simple functions, and calculating the area under the approximate curve. Reproduce the following:

Command	Notes
<code>clear</code>	To empty STATA of all data and variables before a new session.
<code>Range x 1 3 200</code>	Divide the interval (1, 3) into 200 points. (Inspect the result in the browse window.)
<code>gen y =exp(-(x-2)^2)/x</code>	Calculate y for each x (inspect the result)
<code>line y x</code>	Plot y against x with lines between the points
<code>integ y x</code>	Calculate the integral

Is $f(x)$ a pdf over (1, 3)? (defining $f(x) = 0$ for x outside the interval)

We may also want the cumulative integral, $F(x) = \int_1^x f(u)du$ for $1 \leq x \leq 3$. This is easily obtained by adding the option, `gen(cy)`, to the `integ` command (where I have called the variable with the cumulative values for `cy`). I.e., do the following

`integ y x, gen(cy)` (and inspect the result)

Now, line-plot both y and cy against x in the same diagram by

`line y cy x`

b. Try out the Stata numerical integral in a case we know the answer :

$\int_1^3 e^{-x} dx = e^{-1} - e^{-3} = 0.318092\dots$. Calculate the integral by `integ` as above and compare with the exact result.

c. The function, $f(x)$, in **a.** may be turned into a pdf, by dividing the function by the integral-value. That is most easily done by using that, just after running `integ`, the result is stored in the constant, `r(integral)`. (Check that by the command, `disp r(integral)`, just after having run `integ`. So, to turn the function into a pdf, run the following command

`replace y =y/r(integral)`

Then check the effect by running `integ y x` again.

Exercise 8

One famous approximation in the history of mathematics is Stirling's formula⁴ (discovered around 1730 by *Abraham de Moivre* and *James Stirling*) which provides an approximation to factorials:

⁴ See, e.g., Sydsæther II, exercise 3 to section 6.3

$$(1) \quad n! \approx \sqrt{2\pi} \cdot n^{n+\frac{1}{2}} e^{-n}$$

where the approximation gets better as n increases. The question is in what sense the approximation improves when n gets larger. It actually means that the ratio of the left side of (1) divided by the right side tends to 1 as $n \rightarrow \infty$ (as proven by de Moivre and Sterling). What about the difference, i.e., the left side minus the right side?.

Let us illustrate this by Stata:

Command	Notes
clear	Empty Stata
set obs 20	
egen n=fill(1 2)	
click the browse icon to see the result	
gen nfac=round(exp(lnfactorial(n)),1)	Generates $n!$. Stata calculates first $\ln(n!)$ by the function <i>lnfactorial</i> . To get $n!$, we therefore calculate $\exp(\ln(n!))$. This may create some decimal points which are removed by the <i>round</i> -function.
open the browse window to see the result	
gen nfacappr=sqrt(2*_pi)*n^(n+.5)*exp(-n)	Generate the approximate value of $n!$.
gen lerror= nfac- nfacappr	Generate the absolute error
gen frerror= nfac/ nfacappr	Generate the relative error
open the browse window to see the result	
format nfac %25.0fc	To change the look of $n!$ in the browse window.
open the browse window to see the result	

Note that the absolute error seems to increase with n while it is the relative error that seems to converge to 1.

It was this approximation that enabled de Moivre to derive the normal distribution approximation to binomial probabilities – which seems to be the first known appearance of the normal distribution in the history of mathematics.