- vargranger can be used to test causality.
- Numbers < . < .a < … < .z
- about tells you the version of the stat, a bit like the winver command in windows
- use verinst to check whether the installation is okay
- set maxvar/matsize/mem #, permanently
- type drop _all ahead of set mem because you can't have any variables in the memory before setting the memory; I guess a clear command is sufficient, or equivalent to drop _all
- F2 is equivalent to #review; so that the previously used commands would be recalled
- Program the hot keys using the statement such as global F2 "" and you can put such statements inside a profile.do file in the initialization folder
- Press the pgup key to retrieve the previous command and pgdn for the next one, click on ESC to clear the command window
- There is a nice variable name completion feature in stata, where you can type only partial variable name and press Tab key to let stata finish the variable name
- If a command doesn't require a variable name, it is typically assumed to use _all, i.e., all the variables inside the memory
- If a dataset is not already sorted, then we can add a sort option behind the by statement like by var, sort:
- The in range statement is specified as in #1/#2, where the first and the last observations are abbreviated as f/l
- The in range cannot be used in place of a by var: qualifier because the in ranger refers the absolute position, not relative ones
- Once you have typed the varlist for the command, the option can appear anywhere, separated by a comma, but we can't put an option in the middle of a varlist
- 1(2)10 refers to the list of numbers 1 3 5 7 9, note that 10 is not part of the list because of the increment being 2
- with a mistaken clear command, you can lose hours of work; with a mistaken replace command, you can lose weeks of work
- the tilde ~ can be used to mean zero or more characters go here.
- ~ is the same thing as *, except that ~ makes one further restriction that there will be only one such variable be identified.
- ? stands for one character only
- you may place a dash between two variables to include all variables stored in between these two variables; to see what variables are stored in between, see the variables window.
- We can put a varname:varlabel to make the equivalent of two commands: one to name the variable, and the other to label the variable
- The format of time series operator is operator#.varname. L is the lag operator, F is the forward operator, S is the seasonal adjustment operator, and D is the difference operator
- S4.var refers to a four period difference, or var(t)-var(t-4)
- Stata also understands operator(numlist). For example, L(1/3).c is equivalent to L1.c L2.c L3.c we can also apply the operators to variables in the same parenthesis, for example, L(1/3).(gnp cons) refers toL1.gnp L2.gnp L3.gnp L1.cons L2.cons L3.cons

- We need to tsset the time series before the use of time series operator, and we can also tsset cvar time to set a cross-sectional-time-series data series
- If we want to perform the operation on only the first three variables of each by group, then we can write by varlist: command if _n<=3
- by varlist1 (varlist2): command implies that STATA needs to verify if the data are sorted by varlist1 and then varlist2. If no, yields an error message and quit; otherwise, execute the command by only varlist1.
- To detect missing values, we use x >= .; to exclude missing value, we use if x<.
- string() and real() are two functions that can make the conversion between character and numeric values possible.
- A numeric format starts with a % sign, indicating the beginning of a format. We can follow % with a dash to make it left justified. We can also add a zero after % to indicate filling zeros in front of the number. %w.dx, where w is a numeric digit for the number width, and d means the number of decimal places, where x can be g (for general), e (for exponential), c (for comma format) or f (for fixed format). %d signifies a date format, although we can still add a dash after the % to indicate left justification
- %t is a time series format
- %ws stands for the string format, where w indicates the width of the string
- label data "a string" assigns a string to the data set, like a title for the data set; label variable "a string" assigns a label for the variable; label define sexlabel 0 "male" 1 "female"; label values sex sexlabel assigns a new value for the variables with assigned label value that were defined beforehand.
- encode sex, generate(gender) makes a new variable gender that creates a new variable that is essentially identical to sex, except that the value labels were assigned to gender and we can use STATA to perform statistical commands on the new-numeric variable gender, whereas the original string variable sex is not recognized.
- We can type note : then a list of strings to signify some notes for the dataset. We can later display the notes by type the command notes
- When we add two string variables together using +, it effectively acts as a string concatenation operator. We can also use substr, string and upper functions for the string functions
- ~ is equivalent to ! for negation
- [eqno]_b[varname] is the same as [eqno]_coef[varname], referring to the most recently estimated coefficients
- [eqno]_se[varname] accesses the standard error of the most recently estimated model
- _n refers to the current observation whereas _N refers to the total number of observations of the dataset being used
- _cons is a constant 1 when directly used, or the intercept term when indirectly used such as _b[_cons]
- _pi contains π
- _rc is the return code from the most recent capture command
- if a categorical variable is part of the explanatory variables, then we have to use _b[varname[cvalue]] or _se[varname[cvalue]] to access the coefficient or standard error. We may use [eqno]_b[varname[cvalue]] as an abbreviation for [eqno]_b[varname[cvalue]], or even [eqno]varname. There are two ways of referring to the equation number eqno:

an absolute form or an indirect way. We may use [#1] to refer the first equation in the absolute form. In the case of multinomial logit model, we can even use the outcome value to refer the equation number, without putting a # sign in front of it. That is, [1]varname is fine.

- Most stata results are returned in the r-class so that we can use r(name) to access the value or return list to access the list of r-class returns from the most recent command.
- Most estimation results are returned in the e-class so that we can use e(name) to access the value or ereturn list to access the list of e-class returns in the most recent model.
- There are some results returned in the s-class that we can use s(name) to access the value or sreturn list to access the list of s-class returns in the most recent model.
- When by varlist: is used, _n ad _N refer to the position within each by group. Suppose that yesno is a label value with 1 for yes and 0 for no. Suppose that there is a numeric variable answer with label value yesno. We can select all yes answers with if answer ==1 in the usual way because answer is a numeric variable. Alternatively, we can also use if answer ==“yes”:yesno because the yesno label value would replace “yes” automatically with 1 even though answer is not a string variable so strictly speaking, we can’t use answer == “yes”.
- Be aware that the valuation of y == 1.1 may not return the correct answer because y may be stored as a double-precision variable, yet the comparison basis 1.1 is stored as single-precision. One way of bypassing this problem is to use y == float(1.1).
- We use matrix list matname to list the values of a matrix. We use matrix rownames matname = varlist to set the row names for the matrix, and we use matrix colnames matname = varlist to set the column names.
- equation_name:ts_operator.sub_name is the naming convention for matrix
- When using matrix input matname = (#,#\#,#), we don’t have any length limitation on the matrix size, but we can’t use functional operators inside the input elements. When using matrix matname = (#, #\#,#), we have a size limitation but we may use functional operators inside the input elements.
- We can use translate filename.smcl filename.log, replace to translate the log file
- If you are interested in running a job without seeing any output to the screen or log file, then use the command run dofile
- A program is defined by program progname … end
- Before using the program, we have to load the program using the do progname command, then we can call the progname anywhere we want.
- It is also quite common to type the progname right below the program … end statements so that the do progname would load the program and execute it.
- Once a program is defined, stata doesn’t allow you to redefine it.
- If we want to redefine a program, we have to type program drop progname before the new definition. But if we put a program drop progname in front of the new program definitions of the same do file, then we have a problem if the progname is not defined before and then stata would return an error. A proper workaround is to add a capture command in front of the program drop progname so that stata doesn’t care if the progname is found or not.

- Define a local macro variable by local macroname “a string” and we can call the local macro just defined by ‘macroname’, with single quotes. **IMPORTANT**: the left single quote is the character under the tilde ~, so we should be using `’ effectively.
- We may stack two macro variable together to get a new string such as ‘macro1’‘macro2’
- Global macro can be defined in much the same way as local macro. But the way of calling global macro is different from calling local macro. We use a dollar sign in front of the macroname to call a global, but use a pair of single quotes to call local macro.
- We may use local macroname content to store the content directly to the macro. Alternatively, we can us local macroname = expression to store the results of the expression to the local macro. Note that the expression can be numeric or string.
- While we can input the same string to the macro variable either using the content directly or using the expression evaluation method, the evaluation method has a string limitation much more severe than the content copying method.
- The following two statements are equivalent: local i = ‘i’+1 vs. local ++i; similarly, the following two statements are also equivalent: local i = ‘i’-1 vs. local --i;
- If we need to evaluate the content of the macro to see if it is the same as another string, then we might encounter some strange looking expressions such as “ ‘macroname’ ” == “base”
- Compounded double quotes, ‘ “” ’, is necessary if the content inside double quotes may also contain double quotes.
- ‘i++’ expands the local macro ‘i’ first and then increment the local macro ‘i’. on the other hand, ‘++i’ increment the local macro ‘i’ first and then expand the incremented ‘i’.
- the usage of ‘=exp’ is equivalent to assign a new local macro by the expression and then pass the string output of the expression by calling the newly defined local macro.
- Stata does’ mind if we refer to a local macro that doesn’t exist, it simply puts nothing in its place. We can use the macro vector by putting layers of ‘’ for the local macro’s. we can eliminate a macro by defining nothing to the macroname, such as define macroname ;
- We can built layers of global macro by xi or ${x$i}. We can also mix global and local macros such as ${x‘i’}
- We can use \ in front of a macro name to postpone the substitution of the macroname.
- Because \ was used to delay the parsing, don’t use \ to access file folders under windows. Instead, use the forward slash /.
- When passing arguments for programs, just type the variable names right after the progname. Program arguments are passed to program via local macros, where ‘0’ refers to the entire string followed the progname, ‘1’, ‘2’, etc., refer to the first, second word of the input string, ‘*’ is similar to but different from ‘0’ because any odd spacing and double quotes are gone.
- Since ‘1’, ‘2’, …, is hard to read and use, we prefer putting an args argnames right below the program line so that we can refer to the arguments by names.
- set obs n is the command we use to make a set of n observations with no particular variables.
- It’s a way of counting the number of arguments passed to the program.
  ```
  local k = 1
  while “ ‘ ‘k’ ’ ” != “” {
  ```

```
                local ++k
}
local ++k
```

- while is the all-purpose loop command in stata, although there are foreach and forvalues available too. The following two segments are equivalent:
  while 'i' <= 'k' { code segments; local ++i }
  forvalues i = 1(1)'k' { code segments }
- when the preserve command is issued, stat holds a copy of the data in the memory and restore the data when program exists
- assign temporary variables using tempvar varlist before using the generate command, so that these variables will be automatically dropped when stat exits the program. Refer these variables by single quotes like local macros.
- We can use tempname, just like tempvar, to create temporary scalars and matrices. Remember to call them as local macros using single quotes
- We may create temporary files by issuing tempfile filelist in conjunction with preserve and restore. Remember to refer these filenames like local macros using single quotes.
- To return results from programs, add the appropriate option in the program line, such as rclass, eclass or sclass, and finish the program (before the end line) with return/ereturn/sreturn plus the return type (scalar, local, or matrix).
- You can also put the return statement right after the results have been generated, not merely on the block right before the end statement. If you do this, however, you have to use return(varname) inside the program afterwards to re-use the results you just returned.
- A single ado-file can contain more than one program, but if it does, the other programs defined in the ado-file are assumed to be subroutines of the main program.
- The syntax command needs to be revisited soon.
- After an estimation is done or restored, we can use if e(sample) to specify the particular subset, or the entire set if relevant, for further analysis.
- We can always type the command vce to access the variance-covariance matrix of the estimators in active memory.
- Use egen newvname = group(list_of_by_var) to create a numeric list of identifiers for the specified group of sorting variables.