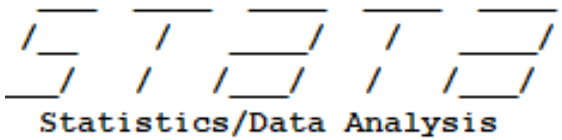
Introduction to Stata

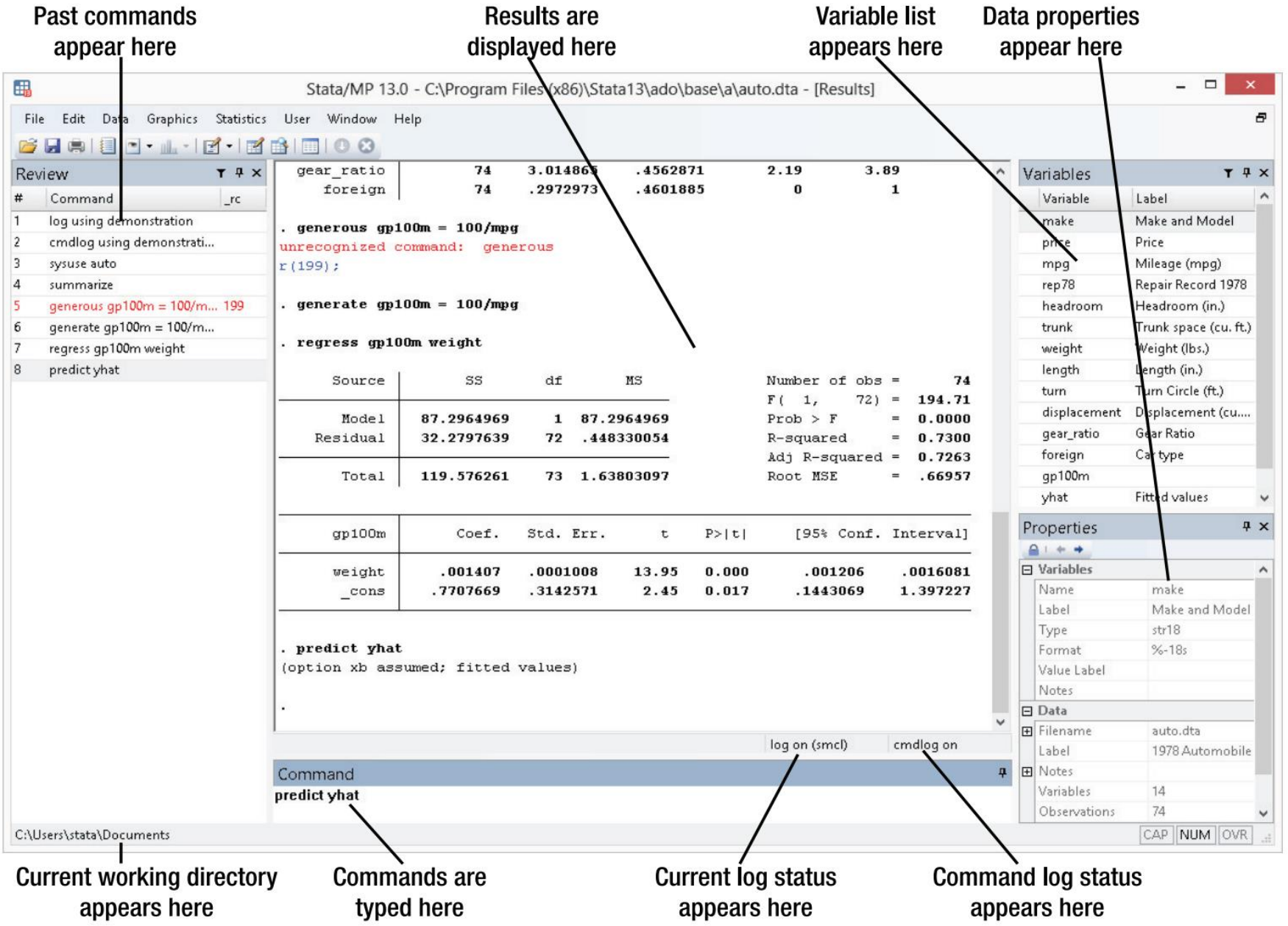


Spring 2019

Francisca Alba

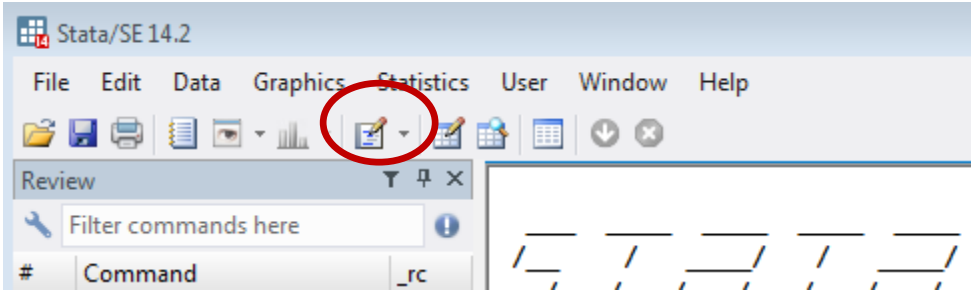
The Stata Screen

- **Variables** window (right side) - lists all variables along with their label (if it exists).
  - Tip: The properties box of the variables window gives you the number of observations in your dataset.
- **Command** window (at bottom) – a place to write commands (but remember to use a .do file so that you can save them).
  - Tip: the command box can be useful if you want to run a quick tabulation.
  - BUT: never use the command window to make CHANGES to your data.
- **Review** window (left side) - accumulates all commands run in a session.
  - Tip: Use the page up button on your keyboard to run through your older commands.
- **Results** window (center) shows all results as produced.
- General commands (file, edit, etc.) at very top-left of screen. Also allows you to generate commands.
  - Use these buttons as a backup if you can't remember the syntax of a command. However, using syntax, rather than this point- and-click methods is considered better programming.


Past commands appear here
Results are displayed here
Variable list appears here
Data properties appear here
Stata/MP 13.0 - C:\Program Files (x86)\Stata13\ado\base\a\auto.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Review
Command _rc
1 log using demonstration
2 cmdlog using demonstrati...
3 sysuse auto
4 summarize
5 generous gp100m = 100/m... 199
6 generate gp100m = 100/m...
7 regress gp100m weight
8 predict yhat
gear_ratio 74 3.014865 .4562871 2.19 3.89
foreign 74 .2972973 .4601885 0 1
. generous gp100m = 100/mpg
unrecognized command: generous
r(199);
. generate gp100m = 100/mpg
. regress gp100m weight
Source SS df MS
Model 87.2964969 1 87.2964969
Residual 32.2797639 72 .448330054
Total 119.576261 73 1.63803097
Number of obs = 74
F(1, 72) = 194.71
Prob > F = 0.0000
R-squared = 0.7300
Adj R-squared = 0.7263
Root MSE = .66957
gp100m Coef. Std. Err. t P>|t| [95% Conf. Interval]
weight .001407 .0001008 13.95 0.000 .001206 .0016081
_cons .7707669 .3142571 2.45 0.017 .1443069 1.397227
. predict yhat
(option xb assumed; fitted values)
.
log on (smcl)
cmdlog on
Command
predict yhat
C:\Users\stata\Documents
Variables
Variable Label
make Make and Model
price Price
mpg Mileage (mpg)
rep78 Repair Record 1978
headroom Headroom (in.)
trunk Trunk space (cu. ft.)
weight Weight (lbs.)
length Length (in.)
turn Turn Circle (ft.)
displacement Displacement (cu....
gear_ratio Gear Ratio
foreign Car type
gp100m
yhat Fitted values
Properties
Variables
Name make
Label Make and Model
Type str18
Format %-18s
Value Label
Notes
Data
Filename auto.dta
Label 1978 Automobile
Notes
Variables 14
Observations 74
CAP NUM OVR
Current working directory appears here
Commands are typed here
Current log status appears here
Command log status appears here

The Do-file

- You can type all the same commands in a Do-file that you would type in the command window.
- But, the Do-file allows you to SAVE all your commands.
- To open a Do file, go to File → Do OR click here:



- Do-files are a saved chronological list of everything you have done to your data.
  - Allows you to go back a re-run commands, analyses, and make modifications to your commands.
- USE A DO-FILE!!
  - For your final project, you will have to turn in your Do-file, so important to use one anyway.

Stata File Extensions

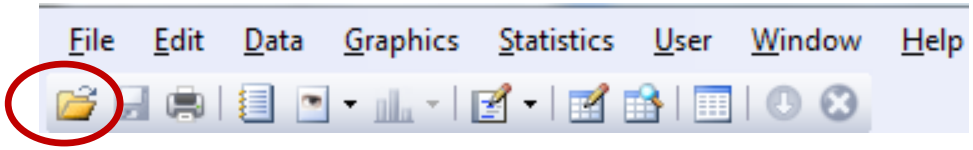
- Like other programs, Stata has file extensions.
- For example, Word documents are .doc/.docx and Excel files are .csv/.xls/.xlxs.
- (Some) of Stata's extensions are:
  - .dta, for a dataset
  - .do, for a Do-file
  - .dct, for a Stata data dictionary (used for infiling)
- Once you have imported in your data, remember to save as a .dta file! That way you can access the dataset the next time without importing.

Importing Data into Stata

- It is highly likely that your final project dataset (or part of it) will not be in .dta format.
- Stata has various commands that allow you to import other file types into Stata.

Importing Data into Stata I

- Three options using a point-and-click approach:
  1. In the main Stata screen: File → Open
  2. Use the folder button:



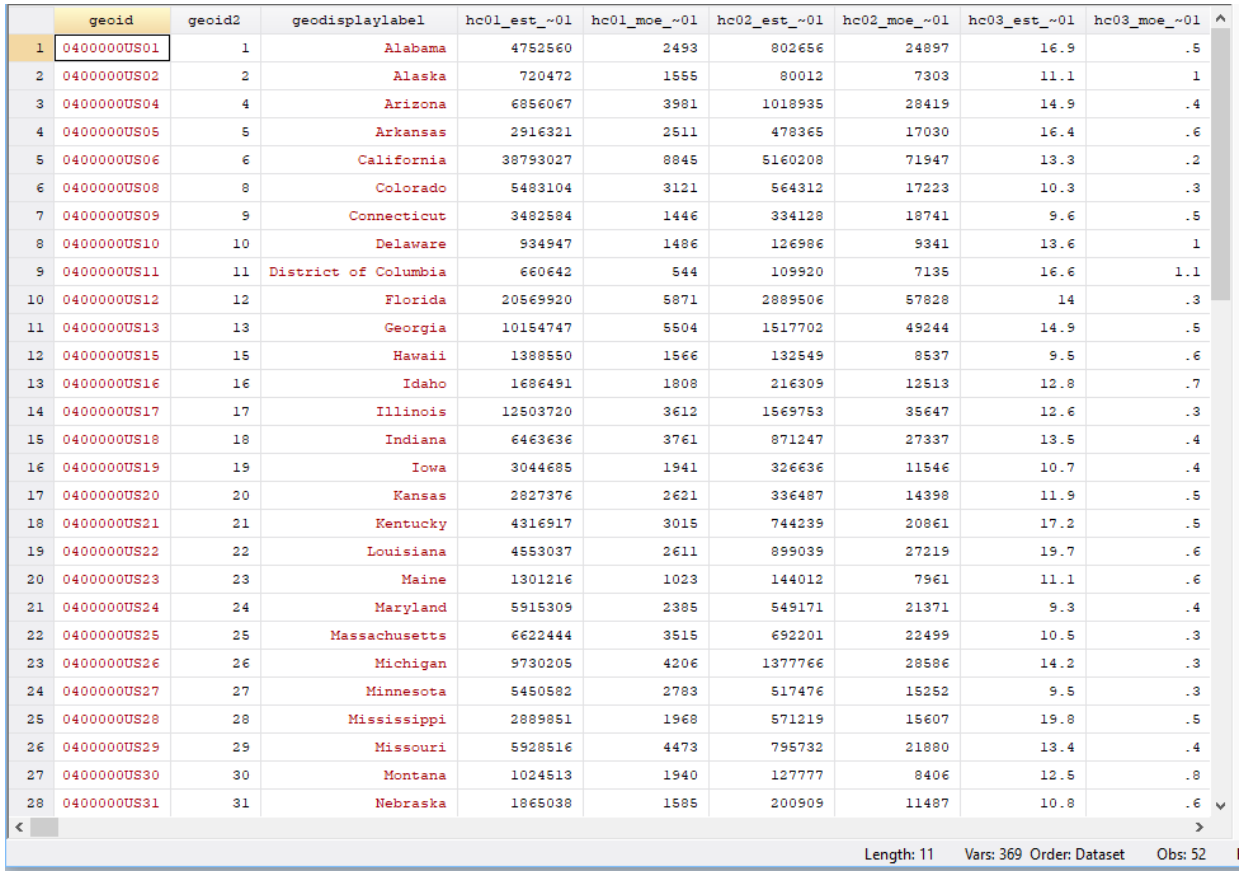
  3. Drag the .dta file into the program
- **But, always better to USE CODE**

Importing Data into Stata II

```
*******************************************************************************
** 0. SET ENVIRONMENT
*******************************************************************************

** Set working directory
**   A working directory is the folder on your computer that you would like to work out of.
**   It can be where your files are stored, or where you would like to store your files.
**   It allows you to call and save data without programming the entire path.

cd "C:\Users\FAlba\EconometricsII"

/*******************************************************************************
** 1. IMPORT
*******************************************************************************

** In .dta format
use "filename.dta", clear

** In .csv format
import delimited using "filename.csv", varnames(1) clear

** In .xls/.xlsx format
import excel using "filename.xlsx", firstrow clear

** In .xpt (SAS) format
import sasxport "filename.xpt", clear

** Unformatted text data
infile var1 var2 using "filename.fileextension", clear

** Text data in fixed format with a dictionary
infile using "datadictionaryname.dct", using("filename.fileextension") clear

** Why "clear?"
**  One limitation of Stata is that it can only hold one dataset in memory.
*   (unlike other stat programs such as R and Python).
**  The "clear" option removes the current dataset from memory so that you can
*   load a new one.

/* FYI: the "varnames(1)"/"firstrow" options tells Stata the first row in your
   excel spreadsheet contains the variable names.*/
```

String vs. Numeric Variables

- If a variable contains any non-numeric characters, the variable imports as a string.
- Stata will give you an error if you try to perform any calculations on string variables.
- You can tell whether Stata is reading your variable as a string if the observations show up in red.
  - Observations in black are non-numeric.
  - Observations in blue are numeric with value labels.



	geoid	geoid2	geodisplaylabel	hc01_est_~01	hc01_moe_~01	hc02_est_~01	hc02_moe_~01	hc03_est_~01	hc03_moe_~01
1	0400000US01	1	Alabama	4752560	2493	802656	24897	16.9	.5
2	0400000US02	2	Alaska	720472	1555	80012	7303	11.1	1
3	0400000US04	4	Arizona	6856067	3981	1018935	28419	14.9	.4
4	0400000US05	5	Arkansas	2916321	2511	478365	17030	16.4	.6
5	0400000US06	6	California	38793027	8845	5160208	71947	13.3	.2
6	0400000US08	8	Colorado	5483104	3121	564312	17223	10.3	.3
7	0400000US09	9	Connecticut	3482584	1446	334128	18741	9.6	.5
8	0400000US10	10	Delaware	934947	1486	126986	9341	13.6	1
9	0400000US11	11	District of Columbia	660642	544	109920	7135	16.6	1.1
10	0400000US12	12	Florida	20569920	5871	2889506	57828	14	.3
11	0400000US13	13	Georgia	10154747	5504	1517702	49244	14.9	.5
12	0400000US15	15	Hawaii	1388550	1566	132549	8537	9.5	.6
13	0400000US16	16	Idaho	1686491	1808	216309	12513	12.8	.7
14	0400000US17	17	Illinois	12503720	3612	1569753	35647	12.6	.3
15	0400000US18	18	Indiana	6463636	3761	871247	27337	13.5	.4
16	0400000US19	19	Iowa	3044685	1941	326636	11546	10.7	.4
17	0400000US20	20	Kansas	2827376	2621	336487	14398	11.9	.5
18	0400000US21	21	Kentucky	4316917	3015	744239	20861	17.2	.5
19	0400000US22	22	Louisiana	4553037	2611	899039	27219	19.7	.6
20	0400000US23	23	Maine	1301216	1023	144012	7961	11.1	.6
21	0400000US24	24	Maryland	5915309	2385	549171	21371	9.3	.4
22	0400000US25	25	Massachusetts	6622444	3515	692201	22499	10.5	.3
23	0400000US26	26	Michigan	9730205	4206	1377766	28586	14.2	.3
24	0400000US27	27	Minnesota	5450582	2783	517476	15252	9.5	.3
25	0400000US28	28	Mississippi	2889851	1968	571219	15607	19.8	.5
26	0400000US29	29	Missouri	5928516	4473	795732	21880	13.4	.4
27	0400000US30	30	Montana	1024513	1940	127777	8406	12.5	.8
28	0400000US31	31	Nebraska	1865038	1585	200909	11487	10.8	.6

Length: 11 Vars: 369 Order: Dataset Obs: 52

Handling String Variables

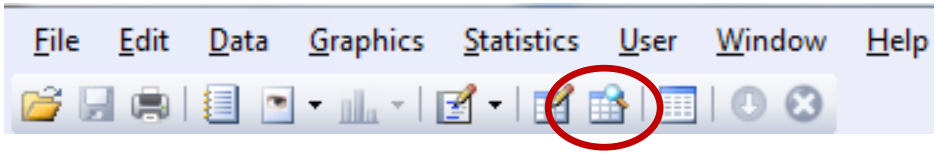
```
/*******************************************************************************
** 2. HANDLING STRING VARIABLES
*******************************************************************************

** Import
import excel using "filename.xlsx", firstrow clear

** If your variable has imported as a string you have two options:

  ** 1. Destring and replace the variable
  destring varname, replace


  ** 2. Destring and generate a new variable. This means you will have two (one string)
  **    and one numeric) variables in your dataset containing the same information.
  **    Some people prefer this option as it preserves the original variable.
  destring varname, generate(newvarname)

/* Note: if the variable has nonnumeric characters you need to specify the
   "force" option. This will convert any observations with a non-numeric character
   to a missing value in stata */

  destring varname, replace force
  destring varname, generate(newvarname) force
```

Looking At Your Data I

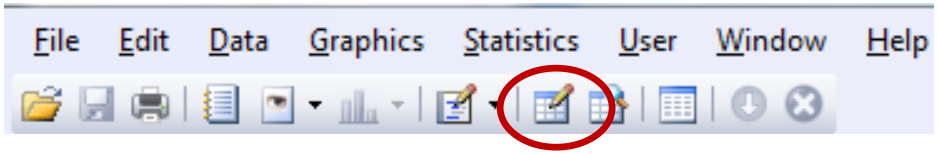
```
/*******************************************************************************
** 3. LOOKING AT YOUR DATA
*******************************************************************************

/* Once you have imported, it is good practice to explore your dataset
   There are various commands that can help you achieve this */

** Browse
browse // ** Browse all variables
browse var1 var2 var3 ... // Browse some variables
browse var1 var2 var3 if var4 == 1 // Browse with conditions

** Describe
/* gives you information on varnames, storage types, display format,
   value labels, and variable labels */

describe // this describes all variables
describe var1 var2 ... // this describes two variables (var1 & var2).

** Tabulate
tab var1 // Tabulate one variable
tab var1 var2 // Tabulate two variables (cross tab)
tabl var1 var2 // Tab multiple variables (individually, rater than cross)

** Table
** Similar to tab but allows for a multiple way ("super variable") tab
table var1 var2 var3

** Summarize
sum var1 // gives you # of obs, mean, std. dev., min, and max
sum var1, detail // gives above and percentiles, variance, and more
```

Looking At Your Data II

- You can also click here to browse your dataset:



- Or, click here to make edits to your dataset. Like, editing the variable name, variable label, or making edits directly into a cell. Remember, all these edits can also be done with code!



Looking At Your Data III

```
/*******************************************************************************
** 4. LOOKING AT YOUR DATA III
*******************************************************************************/

/* The table function mentioned above is a good command to know
   For example, this table provides the sample size (n) and mean
   Of two variables: age and num of family members in the household,
   by a third variable: sex. I exclude negative values of age as in
   this specific dataset negative values correspond to
   outside of the universe of the var. The second line is the same
   table but I include a weight since this is survey data*/

use "nov18.dta", clear

table pesex if prtage >= 0, c(n prtage mean prtage n hrnumhou mean hrnumhou)
table pesex if prtage >= 0 [iw=pwsswgt], c(n prtage mean prtage n hrnumhou mean hrnumhou)
```

```
. table pesex if prtage >= 0, c(n prtage mean prtage n hrnumhou mean hrnumhou)

----------------------------------------------------------------------
      SEX |      N(prtage)     mean(prtage)     N(hrnumhou)  mean(hrnumhou)
----------+-----------------------------------------------------------
     MALE |          59,838   39.1374206543          59,838         3.28218
   FEMALE |          62,906   41.16051101685          62,906         3.20575
----------------------------------------------------------------------

. table pesex if prtage >= 0 [iw=pwsswgt], c(n prtage mean prtage n hrnumhou mean hrnumhou)

----------------------------------------------------------------------
      SEX |      N(prtage)     mean(prtage)     N(hrnumhou)  mean(hrnumhou)
----------+-----------------------------------------------------------
     MALE |        1.58e+08   37.77444076538        1.58e+08         3.34502
   FEMALE |        1.65e+08   39.61488723755        1.65e+08         3.27849
----------------------------------------------------------------------
```

Dictionary of (Some) Symbols

- Stata code uses a combination of pre-existing commands with logical statements:

Stata Syntax	Meaning
=	Equals
>	Greater than
<	Less than
!	Does not
&	And
|	Or

- To reference a value, use combinations of these:

Stata Syntax	Meaning
==	Does equal
!=	Does not equal
>=	Greater than or equal
<=	Less than or equal
&	And
|	Or

- Note: Parentheses work as they do in math
  - i.e., (P & Q) | R is different than P & Q | R

General Tips for Programming in Stata

- Syntax matters
  - Any code that isn't exactly right won't work (at least not the way you want).
- Capitalization matters
  - Commands: use lowercase
  - Variables: Stata recognizes City, city, and CITY as unique variables
    - Thus, it is best to stick with a consistent naming method for your variables (e.g., use lowercase for everything)
- Stata defaults each command to one line, unless you tell it otherwise
  - Tell it otherwise by adding /// to the end of a line (led by a space "_///")
- Annotate your program by adding commented-out text
  - To comment out a line, start it with *
  - To comment out multiple lines, start with /* and end with */
  - Your Do-Files should have comments. This keeps your code more organized, helps to explain what you are doing (and why you are doing it).

Creating & Manipulating Variables – Strings

```
/*******************************************************************************************************
** 5. CREATING & MANIPULATING VARIABLES - STRING VARIABLES
*******************************************************************************************************/
import excel using "povrate.xlsx", firstrow clear

** Creating a string variable: include quotations around the value
gen year = "2017" // this creates a string variable that equals 2017 for each observation
gen month = "01" // this creates a string variable that equals 01 for each observation
gen state = "Alabama" if Geoid2 == 1
/* this creates a string variable that equals Alabama when a second variable, Geoid2 is equal to 1*/
/* Note: that if the Geoid2 variable was a string, we would have to put quotes around the 1 for
   the code to run */

** Create a string variable that is multiple character vars together
gen monthyear = year + month // The resulting variable here is 201701

** Taking out a character from a string
replace povrate = subinstr(povrate, "A", "0", .)

/*this replaces the variable povrate with the variable povrate but takes out the character A and
  replaces it with 0 in all occurances of A */
```

- See here for a list more string functions (there are many!).

Creating & Manipulating Variables – Numeric

```
/*******************************************************************************
** 6. CREATING & MANIPULATING VARIABLES - NUMERIC VARIABLES
*******************************************************************************

/* Often, you will want to create a variable or change
   the coding of a variable that already exists. */

** Gen var equal to 1 for each observation in your dataset
gen all              = 1

** Gen var equal to 1 for each male in your dataset
gen male             = 1 if pesex == 1

/* Once a varaible is generated, you can alter it
   by replacing values. For example: */

** Gen exclusive race variable
/* I create a variable racecat equal to 5, then
   I replace the values to be 4 if another variable
   prtdtrace is equal to 5 and another variable
   prdthsp is equal to -1 and so on. Then I define
   labels for the variable and label the values. */
gen racecat          = 5
replace racecat      = 4 if ptdtrace == 05 & prdthsp == -1
replace racecat      = 3 if inrange(prdthsp, 1,8)
replace racecat      = 2 if ptdtrace == 2 & prdthsp == -1
replace racecat      = 1 if ptdtrace == 1 & prdthsp == -1

label define racecat 1 "White, nh" 2 "Black, nh" 3 "Hispanic" 4 "Asian, nh" 5 "Other"
label values racecat racecat
```

Creating & Manipulating Variables – Boolean Expressions

```
/*****************************************************************
** 7. CREATING & MANIPULATING VARIABLES - BOOLEAN EXPRESSI
*****************************************************************

/* Often, you can use "boolean expressions" to shorten
   your code. For example, I could create a variable
   equal to 0 and then replace it when another variable
   pemlr is equal to 1 or 2.*/

gen emp              = 0
replace emp          = 1 if pemlr == 1 | pemlr == 2

/* Alternatively, I could use a boolean expression, which
   Stata evaluates as a 1 or a 0. This makes my code
   shorter. E.g. only 1 line of code rather than 2.
   These two lines below are both booleans, and do
   the same thing */

gen emp1             = inrange(pemlr, 1, 2)
gen emp2             = (pemlr == 1 | pemlr == 2)

/* These three variables are all the same. I can
   check this by using the browse command and the
   tab command */

br emp emp1 emp2
tab emp, m
tab emp1, m
tab emp2, m
```

Creating & Manipulating Variables – Boolean Expressions

```
. tab emp, m

        emp |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     84,091       58.78       58.78
          1 |     58,959       41.22      100.00
------------+-----------------------------------
      Total |    143,050      100.00

. tab emp1, m

       emp1 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     84,091       58.78       58.78
          1 |     58,959       41.22      100.00
------------+-----------------------------------
      Total |    143,050      100.00

. tab emp2, m

       emp2 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |     84,091       58.78       58.78
          1 |     58,959       41.22      100.00
------------+-----------------------------------
      Total |    143,050      100.00
```

	emp	emp1	emp2
1	1	1	1
2	1	1	1
3	0	0	0
4	0	0	0
5	1	1	1
6	1	1	1
7	1	1	1

Creating & Manipulating Variables - Egen

```
/*****************************************************************
** 8. CREATING AND MANIPULATING VARIABLES - EGEN
*****************************************************************

/* The egen command will handle many other more
   complicated variable creations. For example: */

egen avnmchld = mean(prnmchld)

/* The above command generates a variable 'avnmchld',
   which is the mean value of the total number of
   children under the age of 18 across all observations
   (will be the same value for each respondent)*/

bysort gestfips: egen stavnmchld = mean(prnmchld)

/* The above command generates a variable 'stavnmchld'
   which is the mean value of the number of children
   under the age of 18 across observations in a state
   (will be the same value for each observation within
   the same state, different across states)*/

/* See this page for more egen commands:
   https://www.stata.com/manuals13/degen.pdf*/
```

- Type “help egen” for a full list of functions or go to this page.

Missing Values

- Stata treats missing values as really large numbers
  - Thus, referencing really large numbers will also reference missing values

```
For example, I have a variable that indicates a
peron's labor force status. If the person is outside
of the universe in this dataset, they are given a
missing value for this variable. */

tab pemlr
```

```
. tab pemlr

 MONTHLY LABOR FORCE RECODE |      Freq.     Percent        Cum.
----------------------------+-----------------------------------
           EMPLOYED-AT WORK |     57,492       57.26       57.26
            EMPLOYED-ABSENT |      1,755        1.75       59.01
      UNEMPLOYED-ON LAYOFF  |        192        0.19       59.20
        UNEMPLOYED-LOOKING  |      1,901        1.89       61.09
 NOT IN LABOR FORCE-RETIRED |     19,844       19.76       80.85
NOT IN LABOR FORCE-DISABLED |      5,284        5.26       86.11
   NOT IN LABOR FORCE-OTHER |     13,942       13.89      100.00
----------------------------+-----------------------------------
                      Total |    100,410      100.00
```

Missing Values

```
/* I decide to create an indicator for everyone who is not in
   the labor force, which corresponds to values 5,6, and 7.
   So I run this code: */

gen nilf              = 1 if pemlr >= 5
```

```
. tab nilf

      nilf |      Freq.     Percent        Cum.
-----------+-----------------------------------
         1 |     82,157      100.00      100.00
-----------+-----------------------------------
     Total |     82,157      100.00

. tab pemlr, m

 MONTHLY LABOR FORCE RECODE |      Freq.     Percent        Cum.
----------------------------+-----------------------------------
           EMPLOYED-AT WORK |     57,492       40.06       40.06
            EMPLOYED-ABSENT |      1,755        1.22       41.29
      UNEMPLOYED-ON LAYOFF |        192        0.13       41.42
        UNEMPLOYED-LOOKING |      1,901        1.32       42.75
 NOT IN LABOR FORCE-RETIRED |     19,844       13.83       56.58
NOT IN LABOR FORCE-DISABLED |      5,284        3.68       60.26
   NOT IN LABOR FORCE-OTHER |     13,942        9.72       69.97
                          . |     43,087       30.03      100.00
----------------------------+-----------------------------------
                      Total |    143,497      100.00
```

Missing Values

```
/* What has happened here is that I have coded everyone who is
   missing the pemlr variable as not in the labor force. This
   is not correct! So, what I should have done was: */

gen nilf            = 1 if pemlr >= 5 & pemlr < .
```

```
. tab nilf

       nilf |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |     39,070      100.00      100.00
------------+-----------------------------------
      Total |     39,070      100.00
```

Appending vs. Merging data

- Appending two datasets will stack datasets on top of each other
  - If dataset A has 20 observations, and dataset B has 15 observations, the appended dataset will have 20+15=35 observations
  - Typically use this function if you have multiple cross-sections of the same variable(s). Like each state's poverty rate in different years.
- Merging two data sets will bring two sets of data together BY the variables you want
  - If data set C has 30 observations, and data set D has 25 observations, and data sets C and D share 18 cities, merging by city will give you data set with 18+(30-18)+(25-18)=37 observations
  - Typically do this when data sets share the same unit of observation but have different variables (e.g., one dataset contains the poverty rate by state and another dataset contains the unemployment rate by state).

Appending Data

```
/*****************************************************************
** 10. APPENDING
*****************************************************************

/* I have two cross-sections of the Current Population Survey:
   October 2018 and November 2018. I want to do an analysis on
   both of these months. They have the SAME variables. So, I
   use this code: */

** Load first dataset (if you have not done so already)
use "Oct18.dta", clear

** Append November on to the first dataset
append using "Nov18.dta"

** Remember to save!
save "cps_nov_oct.dta", replace
```

Appending data

- Original dataset (dataset A). In this case, Oct18.dta, which has 143,497 observations:

	pepdemp1	ptnmemp1	pepdemp2	ptnmemp2	year	month
143485	-1	-1	-1	-1	2018	oct
143486	-1	-1	-1	-1	2018	oct
143487	-1	-1	-1	-1	2018	oct
143488	-1	-1	-1	-1	2018	oct
143489	-1	-1	-1	-1	2018	oct
143490	-1	-1	-1	-1	2018	oct
143491	-1	-1	-1	-1	2018	oct
143492	-1	-1	-1	-1	2018	oct
143493	-1	-1	-1	-1	2018	oct
143494	-1	-1	-1	-1	2018	oct
143495	-1	-1	-1	-1	2018	oct
143496	-1	-1	-1	-1	2018	oct
143497	-1	-1	-1	-1	2018	oct

Appending Data

	pepdemp1	ptnmemp1	pepdemp2	ptnmemp2	year	month
143490	-1	-1	-1	-1	2018	oct
143491	-1	-1	-1	-1	2018	oct
143492	-1	-1	-1	-1	2018	oct
143493	-1	-1	-1	-1	2018	oct
143494	-1	-1	-1	-1	2018	oct
143495	-1	-1	-1	-1	2018	oct
143496	-1	-1	-1	-1	2018	oct
143497	-1	-1	-1	-1	2018	oct
143498	-1	-1	-1	-1	2018	nov
143499	-1	-1	-1	-1	2018	nov
143500	-1	-1	-1	-1	2018	nov
143501	-1	-1	-1	-1	2018	nov
143502	-1	-1	-1	-1	2018	nov
143503	-1	-1	-1	-1	2018	nov

- I will now have the same number of variables but way more observations!
- Keep in mind that if my variables were not named the same, you can still append, but both variables will show up in your appended version.
  - If I had a variable year in Oct18.dta and a variable year1 in Nov18, the variable year would show up as missing for all observations in Nov18.dta and the variable year1 will show up as missing for all observations in Oct18.dta

Merging data

```
/*****************************************************************
** 11. MERGING
*****************************************************************

/* To merge data, you need to merge BY the correct
   variables.

   How do we know what the correct variables are? We merge by:
   * Whatever makes the row unique.
   * This may be one ID variable (e.g. respondent ID), or it
     may be an ID variable and a year variable, or an ID var,
     year var, and month var...
   * Note: you can merge datasets with different "levels", but
     you can only merge by variables you have in each dataset.
     For example, you may have a person-level dataset, and
     want to merge in the State-level unemployment rate as
     your author holds this variable constant. As long as you
     have the state where each person lives in your person-
     level dataset, you can merge.

   Lets say that I have two different variables. The poverty
   rate and the unemployment rate, both by State in 2017 in
   two separate Stata files. I want put these into Stata
   together to do an analysis. To do this, I use this code: */

** First I import both files and rename the variables
** Note: the variable that you would like to merge by needs to
*  be named the same in both datasets!
```

Merging Data

```
** First I import both files and rename the variables
** Note: the variable that you would like to merge by needs to
*  be named the same in both datasets!

** Import poverty rate
import delim "povrate1.csv", clear varnames(1)
rename (geoid2 geodisplaylabel hc03_est_vc01) (statefips state povrate)
save "povrate.dta", replace

** Import unemployment rate
import delim "unemp.csv", clear varnames(1)
rename (geoid2 geodisplaylabel hc04_est_vc01) (statefips state unemprate)
save "unemp.dta", replace
```

	geoid	statefips	state	povrate
1	0400000US01	1	Alabama	16.9
2	0400000US02	2	Alaska	11.1
3	0400000US04	4	Arizona	14.9
4	0400000US05	5	Arkansas	16.4
5	0400000US06	6	California	13.3
6	0400000US08	8	Colorado	10.3
7	0400000US09	9	Connecticut	9.6
8	0400000US10	10	Delaware	13.6
9	0400000US11	11	District of Columbia	16.6
10	0400000US12	12	Florida	14
11	0400000US13	13	Georgia	14.9

	geoid	statefips	state	unemprate
1	0400000US01	1	Alabama	5.8
2	0400000US02	2	Alaska	7.6
3	0400000US04	4	Arizona	5.8
4	0400000US05	5	Arkansas	5.6
5	0400000US06	6	California	5.9
6	0400000US08	8	Colorado	4.2
7	0400000US09	9	Connecticut	6.1
8	0400000US10	10	Delaware	5.3
9	0400000US11	11	District of Columbia	6.6
10	0400000US12	12	Florida	5.5
11	0400000US13	13	Georgia	5.8

Merging Data

```
** Merge
use "povrate.dta", clear
merge 1:1 statefips using "unemp.dta"
save "pov_unemp.dta", replace
```

	geoid	statefips	state	povrate	unemprate	_merge
1	0400000US01	1	Alabama	16.9	5.8	matched (3)
2	0400000US02	2	Alaska	11.1	7.6	matched (3)
3	0400000US04	4	Arizona	14.9	5.8	matched (3)
4	0400000US05	5	Arkansas	16.4	5.6	matched (3)
5	0400000US06	6	California	13.3	5.9	matched (3)
6	0400000US08	8	Colorado	10.3	4.2	matched (3)
7	0400000US09	9	Connecticut	9.6	6.1	matched (3)
8	0400000US10	10	Delaware	13.6	5.3	matched (3)
9	0400000US11	11	District of Columbia	16.6	6.6	matched (3)
10	0400000US12	12	Florida	14	5.5	matched (3)
11	0400000US13	13	Georgia	14.9	5.8	matched (3)

```
. merge 1:1 statefips using "unemp.dta"

    Result                           # of obs.
    -----------------------------------------
    not matched                             0
    matched                                52   (_merge==3)
    -----------------------------------------
```

Merging Data

- There are a variety of ways of merging, depending on the level of each data set
  - One-to-one merges (1:1) – most common, you link one unique row in data set A to one unique row in data set B
    - Example – merging a student-level test score data set to a student-level demographics data set
  - One-to-many merges (1:m, or m:1) – where you link one unique row in data set A to multiple rows in data set B (or vice versa)
    - Examples – merging a student-level data set with a state-level educational dataset. Like, student-level test score data with state-level educational spending data.
  - Many-to-many merge (m:m) – there is rarely ever a reason for you to do this. In fact, this is exactly what you are usually trying to avoid!

Merging Data – Understanding Stata Output

```
. merge 1:1 teacherid using "course_test_scores_nodup.dta"

    Result                           # of obs.
    -----------------------------------------
    not matched                             3
        from master                         1  (_merge==1)
        from using                          2  (_merge==2)

    matched                                48  (_merge==3)
    -----------------------------------------
```

- Note: this output tells you that there are three records that didn't merge. One from the master dataset (the dataset you have in working memory in Stata) and two in the using dataset (the dataset you merge in).
- See this page for help on merging data.

Merging Data - Errors

```
. sort teacherid

. merge 1:1 teacherid using "course_test_scores.dta"
variable teacherid does not uniquely identify observations in the using data
r(459);
```

- This error is telling you that there is at least one instance where two rows have the same teacher ID
- Check for duplicates:
  - Save the data you are working on
  - Open the new data, and tag duplicates records:

    duplicates tag teacherid, gen(dup)

    Code creates a variable that flags the duplicate records
  - Then you could potentially tabulate the dup variable and browse the data to see if the records are, in fact, complete duplicates. If so, you could drop one.
  - Note that most of the time this error is telling you that you have not uniquely identified your observations. This means you should check the codebook to figure out what other variables you need to merge by to uniquely identify each observation.

Collapsing Data

- Data can be transposed, reshaped, or collapsed to create aggregated data sets
- The collapse command helps us with this:

collapse [statistic] [varlist], by(variable_category)

```
   For example: */

collapse (mean) prtage hrnumhou if prtage >= 0, by(month pesex)

** This is the same except we use a weight as it is survey data
collapse (mean) prtage hrnumhou if prtage >= 0 [iw=pwsswgt], by(month pesex)
```

	pesex	month	prtage	hrnumhou
1	MALE	nov	39.1374206543	3.28218
2	FEMALE	nov	41.16051101685	3.20575
3	MALE	oct	38.89236831665	3.29156
4	FEMALE	oct	40.99266815186	3.20441

Collapsing Data

```
/* You may want to save your collapsed data, or use your
   collapsed dataset to create a table that you can copy
   into Excel or some other program. Once you run the
   collapse command your dataset changes, and you might
   want to use your original dataset again and Stata has
   cleared it from memory. The solution is to*/

* Use preserve and restore
preserve
collapse (mean) prtage hrnumhou if prtage >= 0, by(month pesex)
restore

/* Note that if you run this a line at a time from your
   Do-file Stata will "forget" about the preserve. Thus,
   run the preserve from your command window, but don't
   forget to put it into your Do-file anyway. */
```

Stata Resources

- Stata manual: type "help" + the command in command window.
- UCLA Stata Page
- Google Stata command/question
- Official Stata Support Page
- Stata Youtube Channel

The End!

Feel free to email me with any programming or econometric questions!

falba@gwu.edu

Stata is an amazing tool. I hope you learn to love it as much as I do.