INTRODUCTION TO STATA

PURPOSE OF THE WORKSHOP

- This workshop introduces the usage of Stata for data analysis
- Topics include
  - Stata as a data analysis software package
  - Navigating Stata
  - Data import
  - Exploring data
  - Data visualization
  - Data management
  - Basic statistical analysis

STATA

WHAT IS STATA?

- Stata is an easy to use but powerful data analysis software package that features strong capabilities for:
  - Statistical analysis
  - Data management and manipulation
  - Data visualization
- Stata offers a wide array of statistical tools that include both standard methods and newer, advanced methods, as new releases of Stata are distributed annually

STATA: ADVANTAGES

- Command syntax is very compact, saving time
- Syntax is consistent across commands, so easier to learn
- Competitive with other software regarding variety of statistical tools
- Excellent documentation
- Exceptionally strong support for
  - Econometric models and methods
  - Complex survey data analysis tools

STATA: DISADVANTAGES

- Limited to one dataset in memory at a time
  - Must open another instance of Stata to open another dataset
  - This won't be a problem for most users
- Community is smaller than R (and maybe SAS)
  - less online help
  - fewer user-written extensions

ACQUIRING AND USING STATA

- https://www.stata.com/
- Which Stata is right for me?
- Flavors of Stata are IC, SE and MP
  - IC ≤ SE ≤ MP, regarding size of dataset allowed, number of processors used, and cost

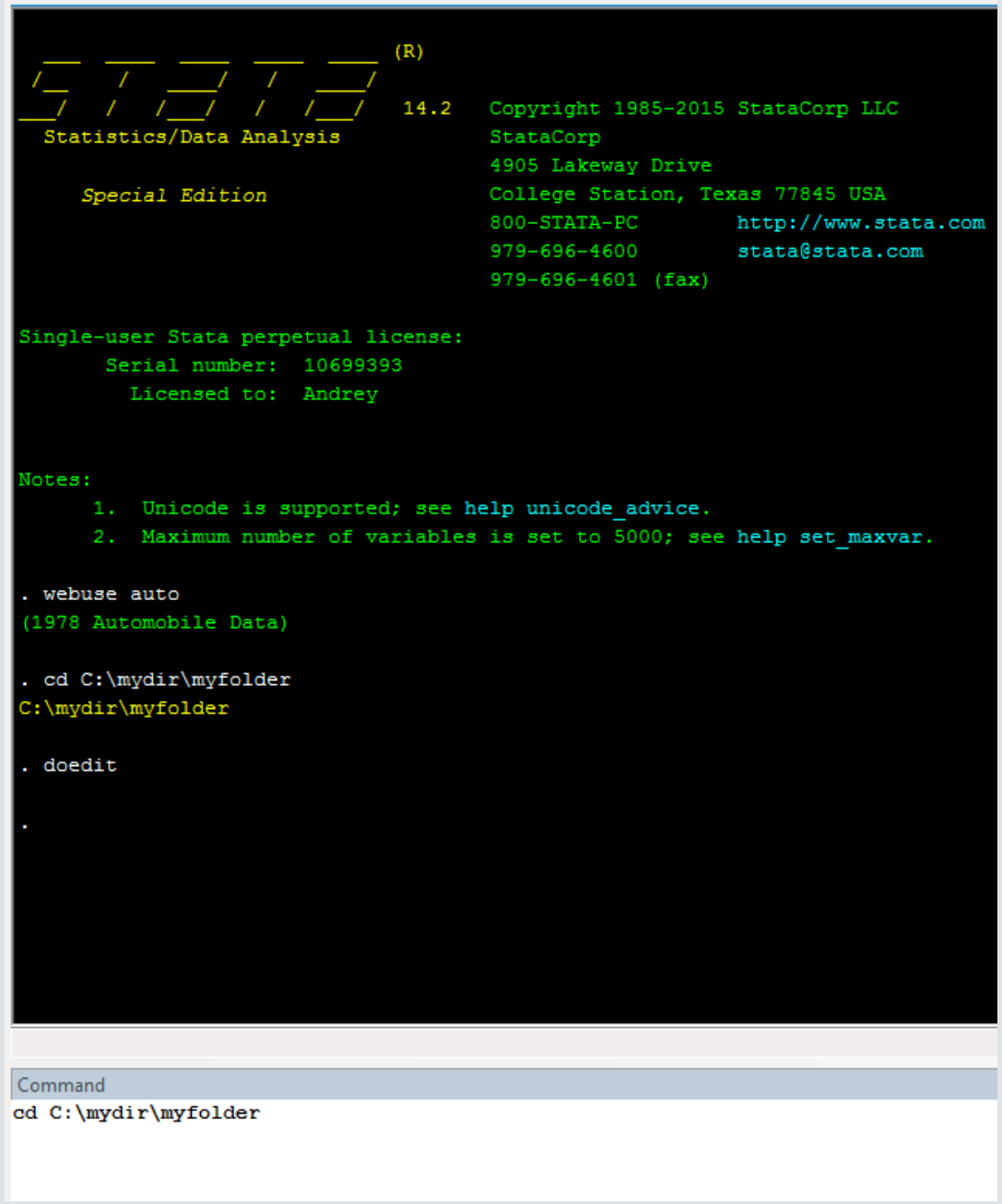
Product features	Stata/BE (Basic Edition)	Stata/SE (Standard Edition)	Stata/MP 2-core	Stata/MP 4-core	Stata/MP 6+
Maximum number of variables					
Up to 2,048 variables	✔	✔	✔	✔	✔
Up to 32,767 variables	-	✔	✔	✔	✔
Up to 120,000 variables	-	-	✔	✔	✔
Maximum number of observations					
Up to 2.14 billion	✔	✔	✔	✔	✔
Up to 20 billion	-	-	✔	✔	✔
Speed comparisons					
Fast	✔	✔	✔	✔	✔
Twice as fast	-	-	✔	✔	✔
Almost four times as fast	-	-	-	✔	✔
Even faster	-	-	-	-	✔
Time to run logistic regression with 10 million observations and 20 covariates					
20 seconds	✔	✔	✔	✔	✔
10 seconds	-	-	✔	✔	✔
5.2 seconds	-	-	-	✔	✔
< 5.2 seconds	-	-	-	-	✔
Maximum number of independent variables					
798	✔	✔	✔	✔	✔
10,998	-	✔	✔	✔	✔
65,532	-	-	✔	✔	✔

NAVIGATING STATA'S INTERFACE

```
cd        change working directory
```

CHANGE WORKING DIRECTORY

Change working directory in Stata for Windows to C:\mydir\myfolder

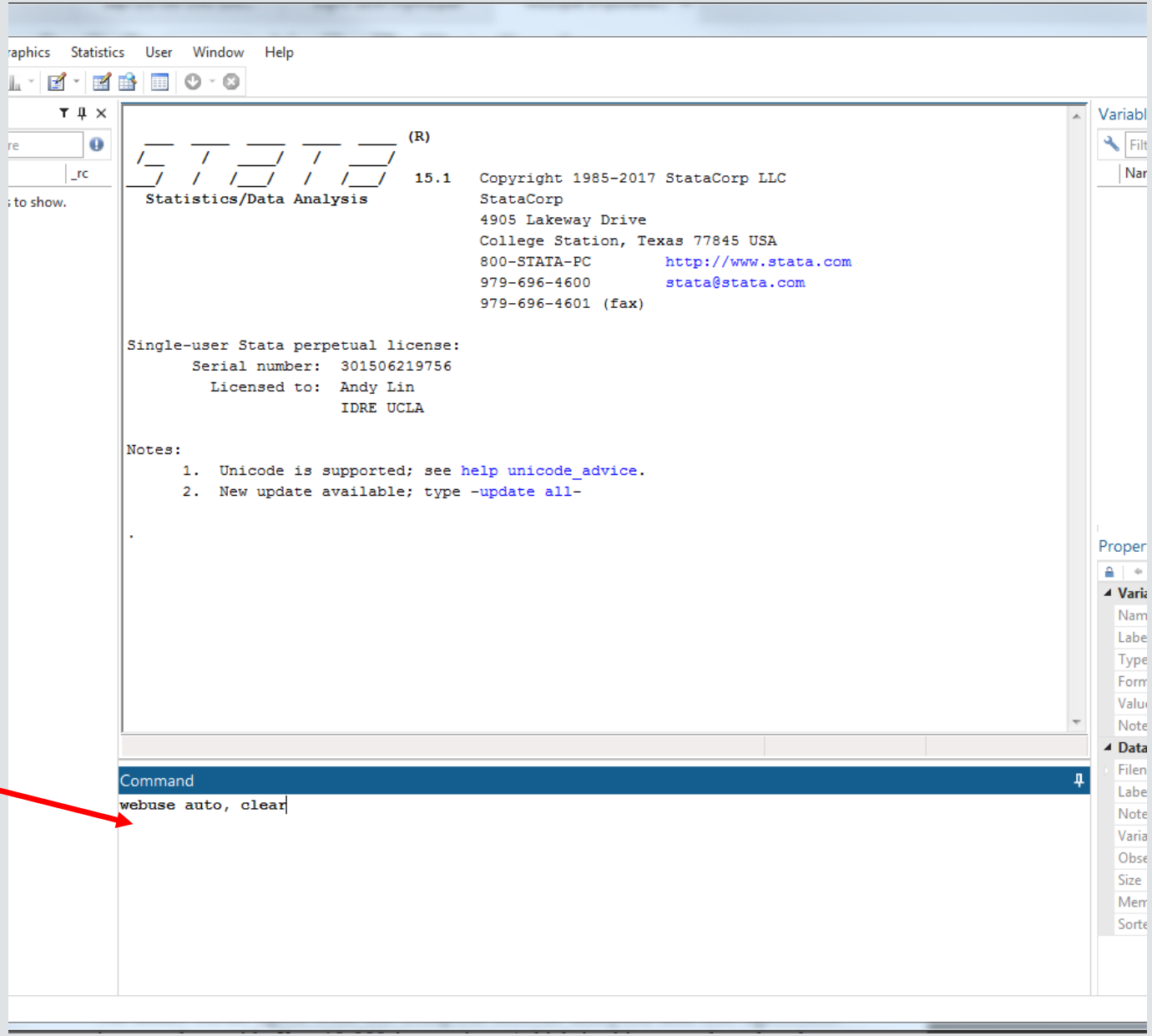


COMMAND WINDOW

You can enter commands directly into the Command window

This command (**webuse**) will load a Stata dataset over the internet

Go ahead and enter the command

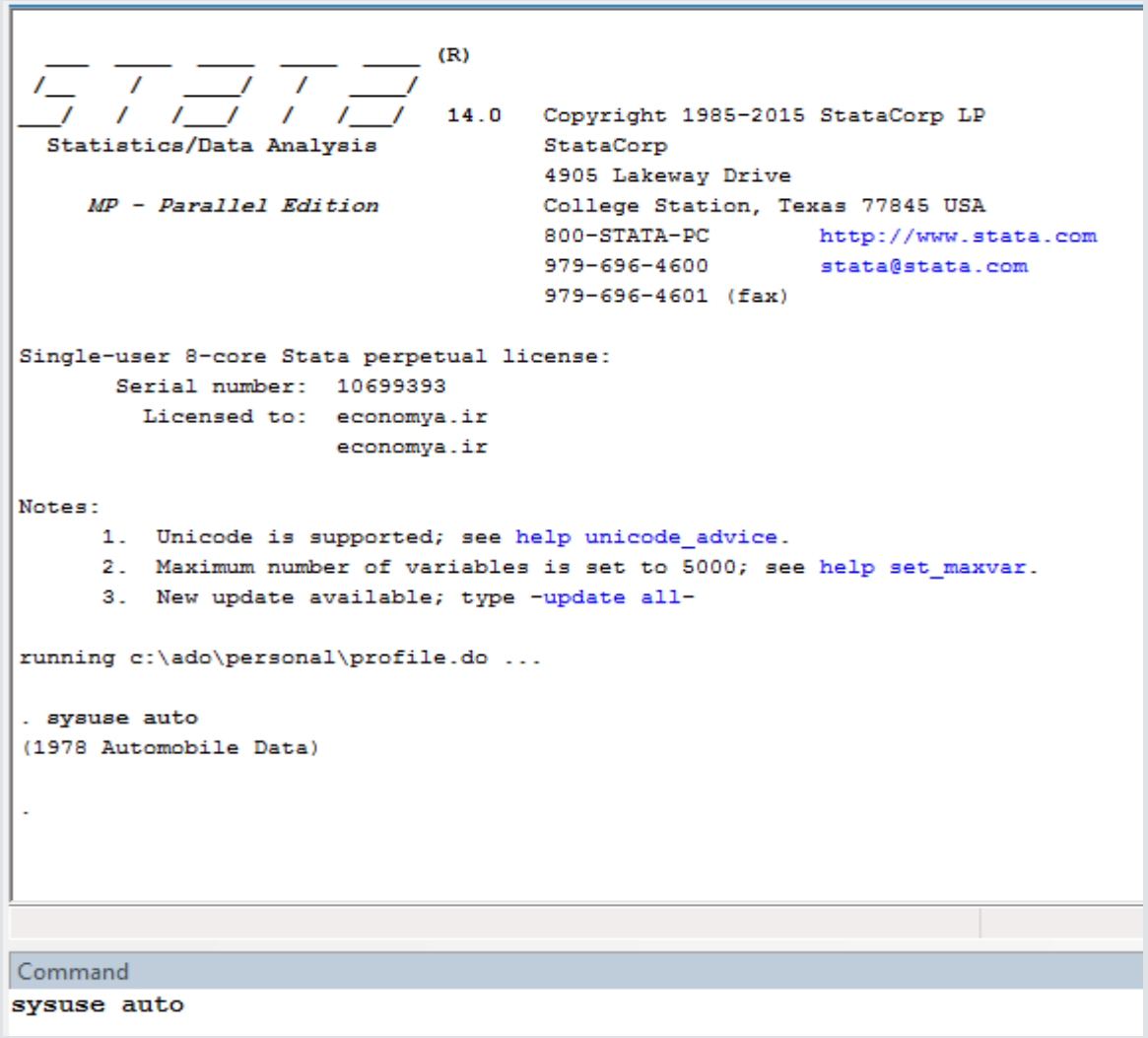


COMMAND WINDOW

You can enter commands directly into the Command window

This command (**sysuse**) will load a Stata dataset over the your system

Go ahead and enter the command

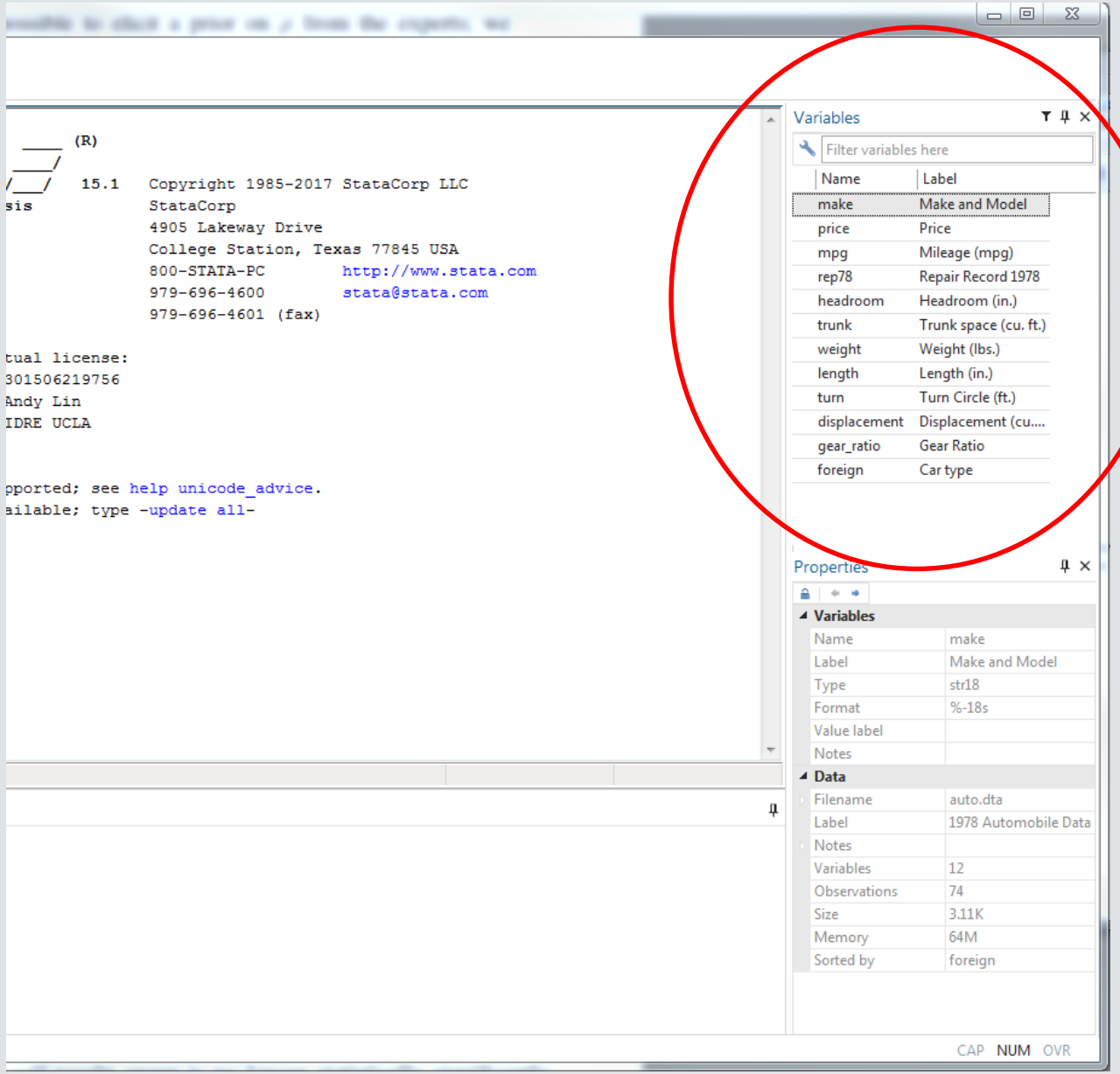


VARIABLES WINDOW

Once you have data loaded, variables in the dataset will be listed with their labels in the order they appear on the dataset

Clicking on a variable name will cause its description to appear in the Properties Window

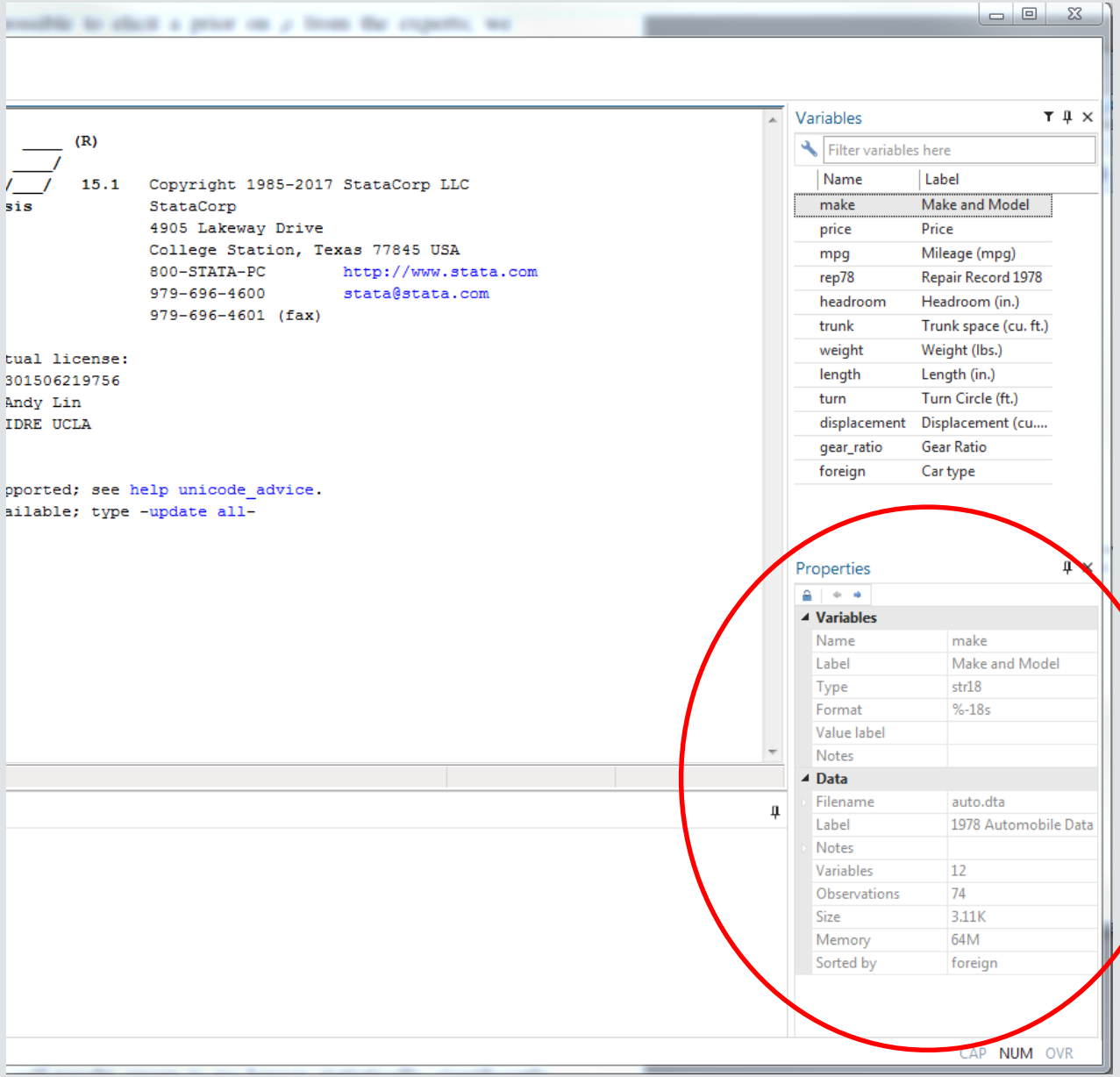
Double-clicking on a variable name will cause it to appear in the Command Window



PROPERTIES WINDOW

The **Variables** section lists information about selected variable

The **Data** section lists information about the entire dataset



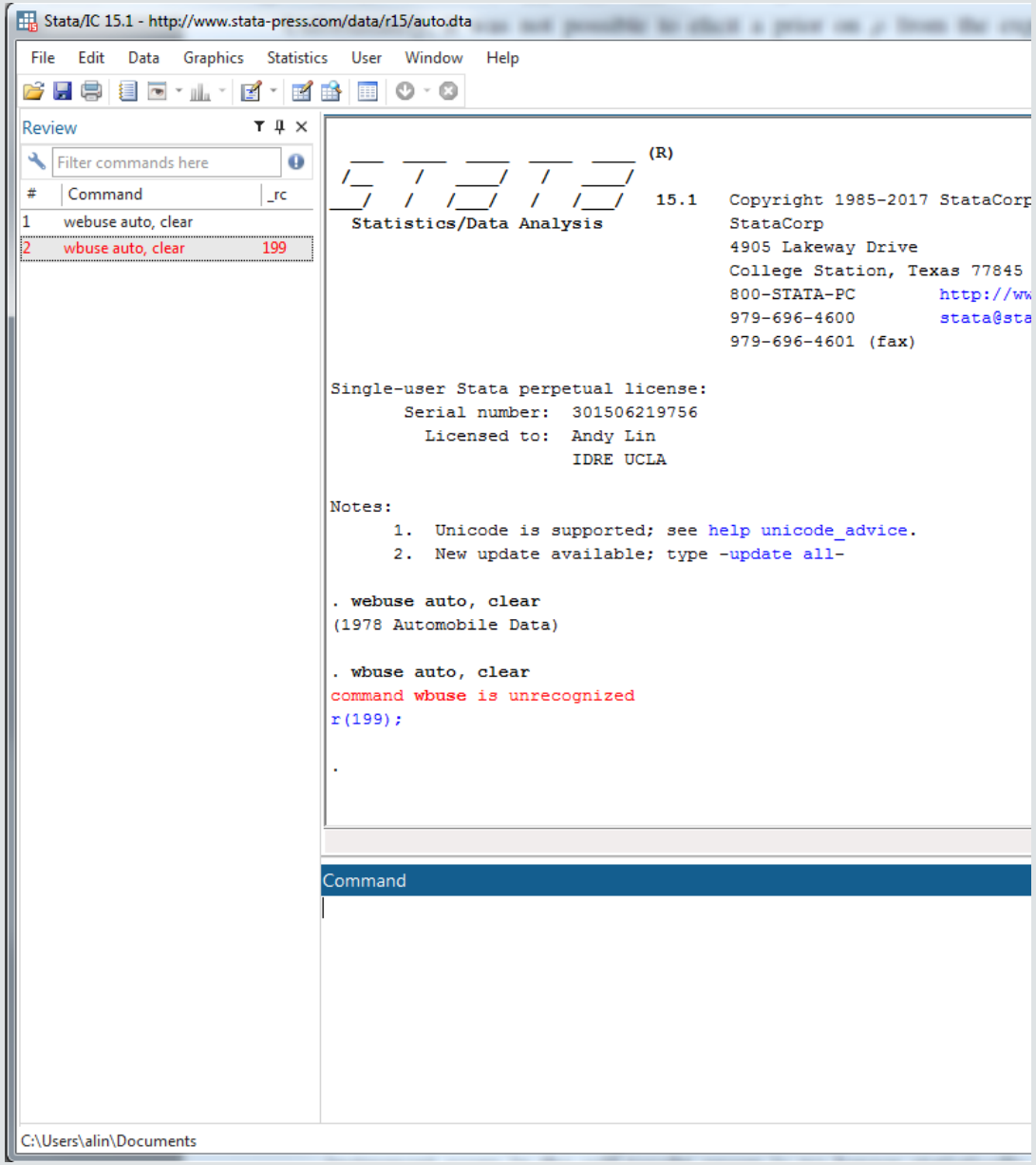
REVIEW WINDOW

The Review window lists previously issued commands

Successful commands will appear black

Unsuccessful commands will appear red

Double-click a command to run it again

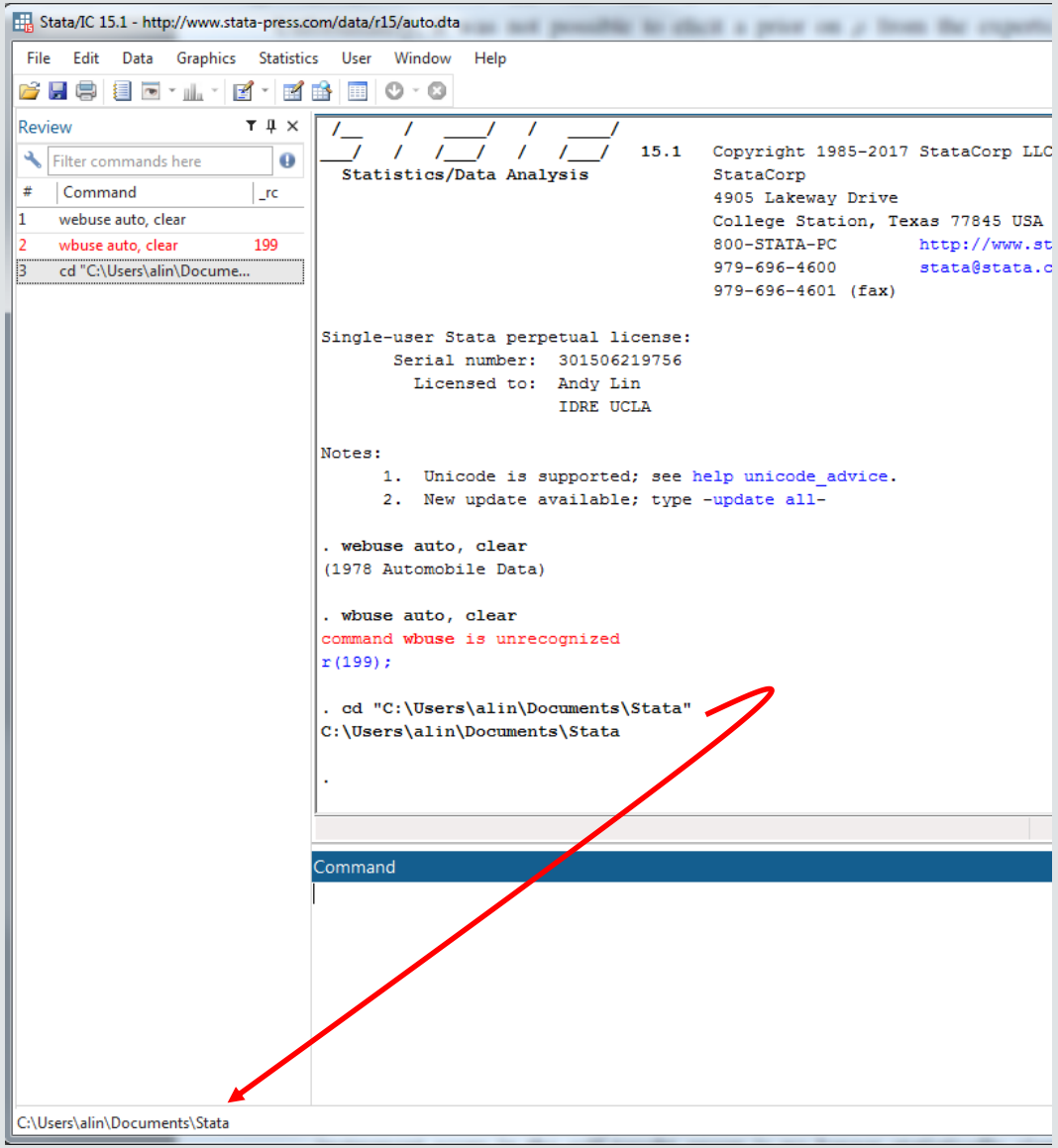


WORKING DIRECTORY

At the bottom left of the Stata window is the address of the working directory

Stata will load from and save files to here, unless another directory is specified

Use the command *cd* to change the working directory

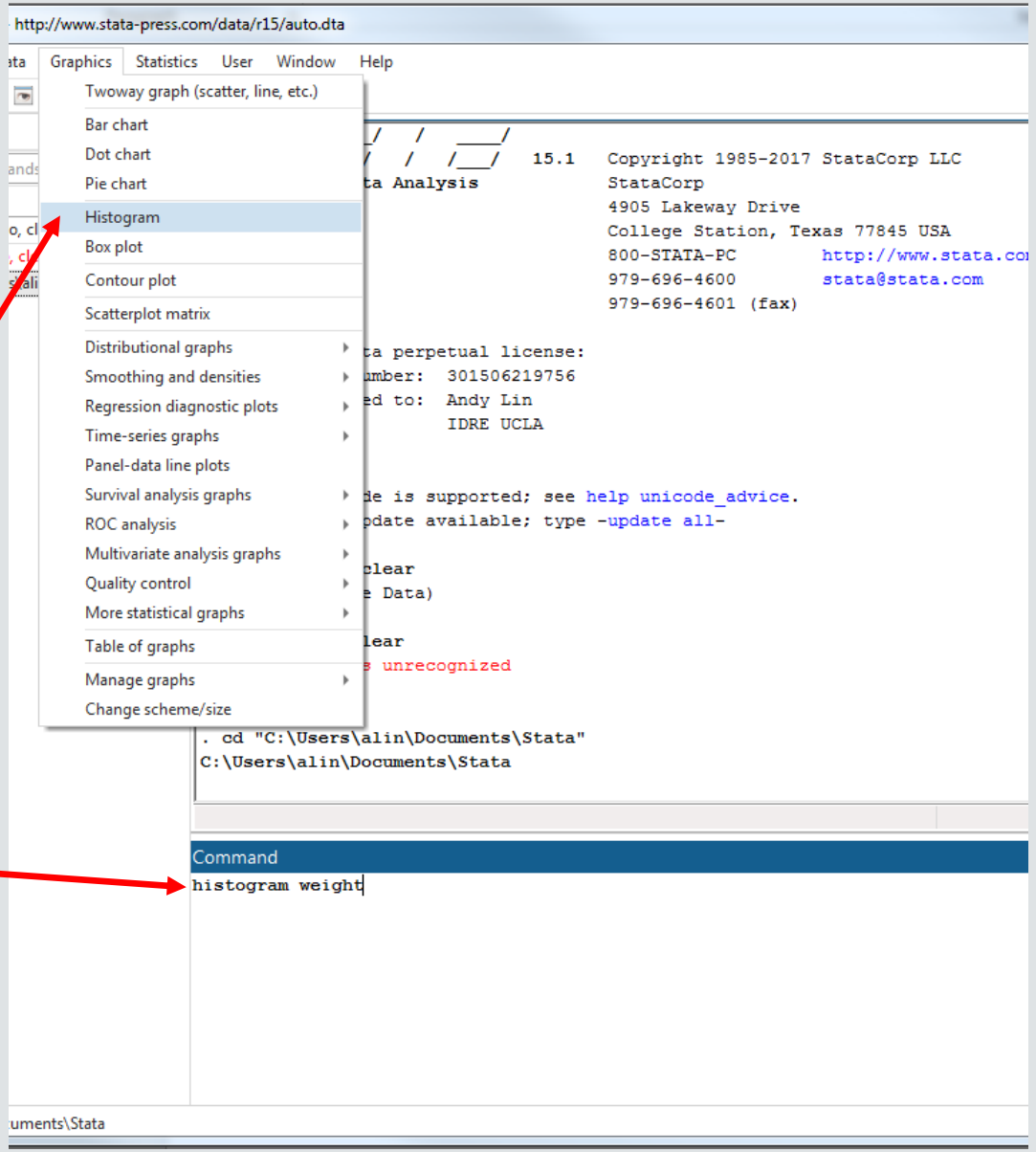


STATA MENUS

Almost all Stata users use syntax to run commands rather than point-and-click menus

Nevertheless, Stata provides menus to run *most* of its data management, graphical, and statistical commands

Example: two ways to create a histogram



DO-FILES

```
doedit  open do-file editor
```

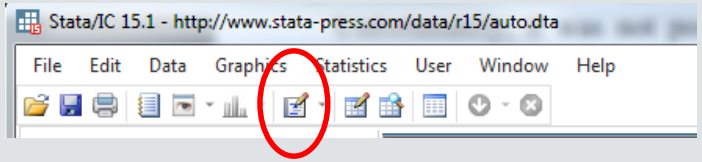
DO-FILES ARE SCRIPTS OF COMMANDS

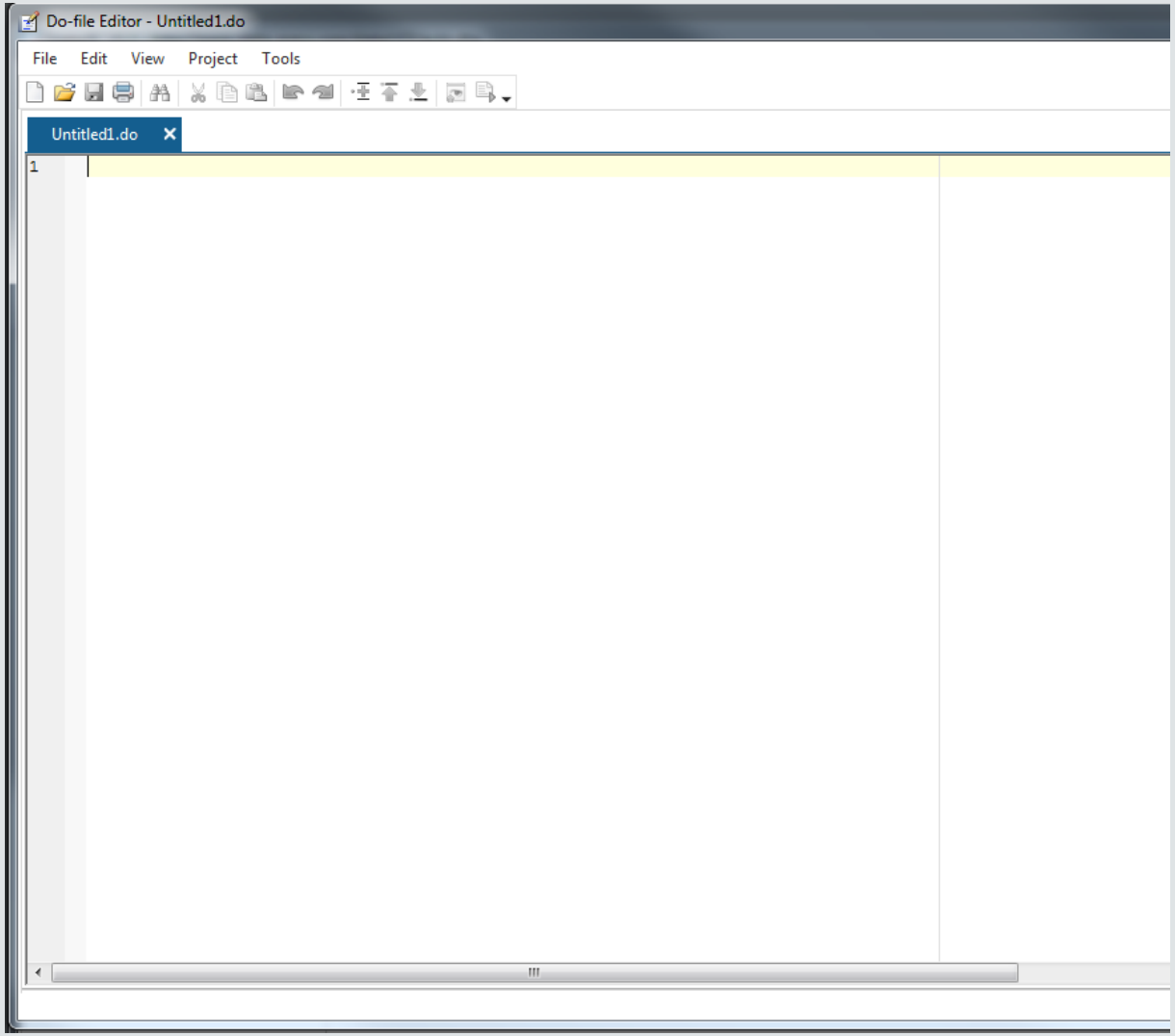
- Stata do-files are text files where users can store and run their commands for reuse, rather than retyping the commands into the Command window
  - Reproducibility
  - Easier debugging and changing commands
- We recommend *always* using a do-file when using Stata
- The file extension .do is used for do-files

OPENING THE DO-FILE EDITOR

Use the command *doedit* to open the do-file editor

Or click on the pencil and paper icon on the toolbar





The do-file editor is a text file editor specialized for Stata

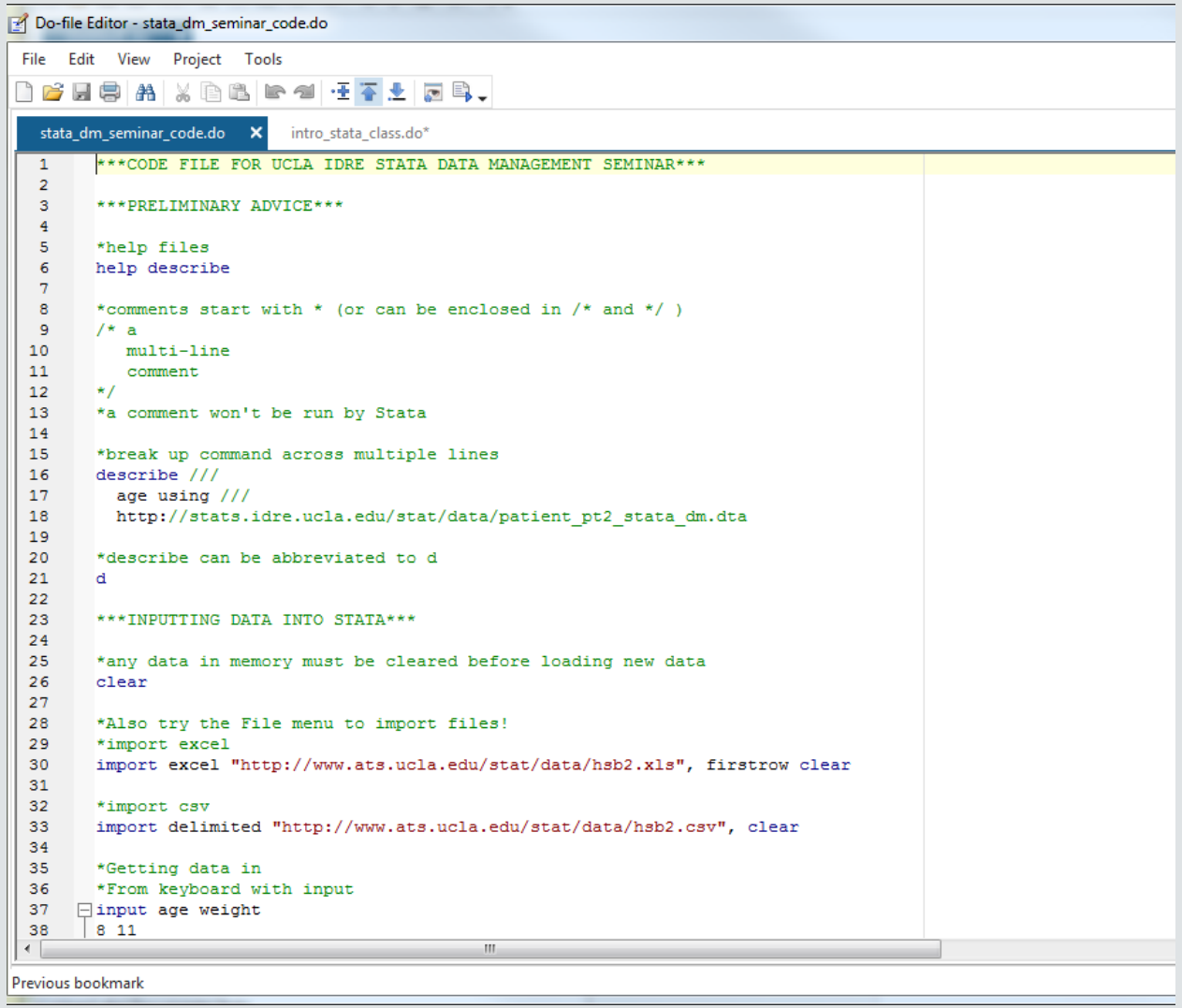
SYNTAX HIGHLIGHTING

The do-file editor colors Stata commands blue

Comments, which are not executed, are usually preceded by * and are colored green

Words in quotes (file names, string values) are colored "red"

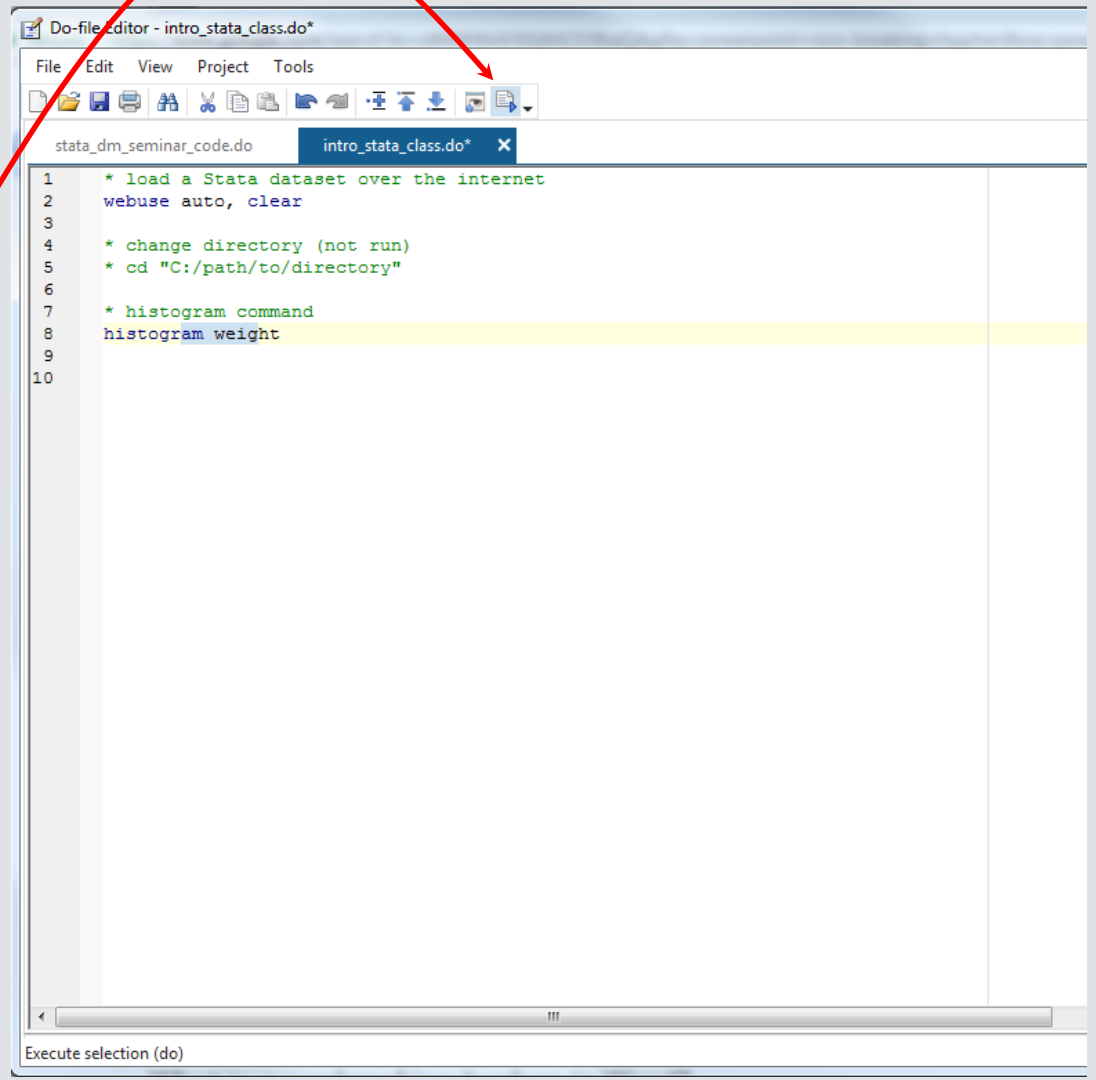
Stata 16 features an enhanced editor that features tab auto-completion for Stata commands and previously typed words



RUNNING COMMANDS FROM THE DO-FILE

To run a command from the do-file, highlight part or all of the command, and then hit Ctrl-D or the "Execute(do)" icon, the rightmost icon on the do-file editor toolbar

Multiple commands can be selected and executed

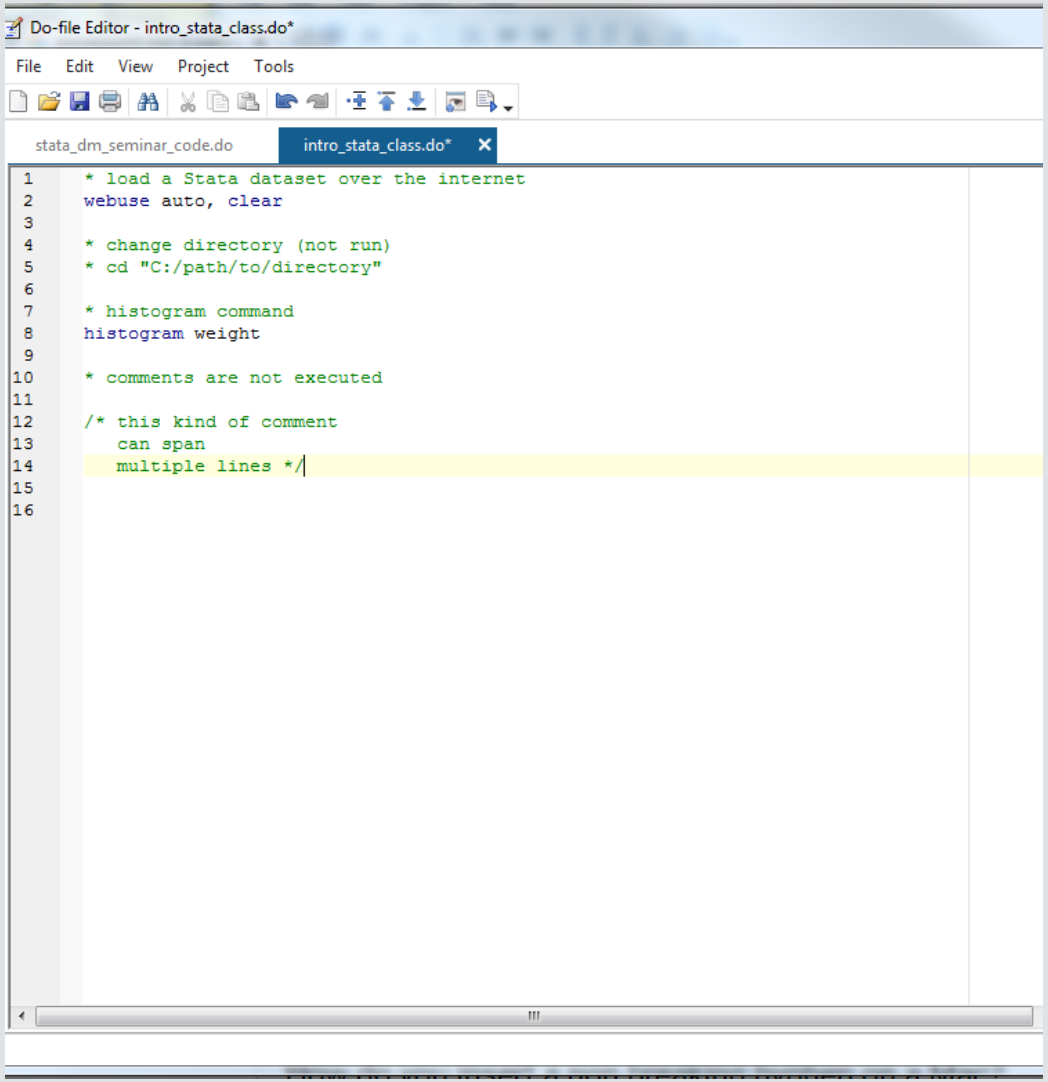


COMMENTS

Comments are not executed, so provide a way to document the do-file

Comments are either preceded by * or surrounded by /* and */

Comments will appear in green in the do-file editor



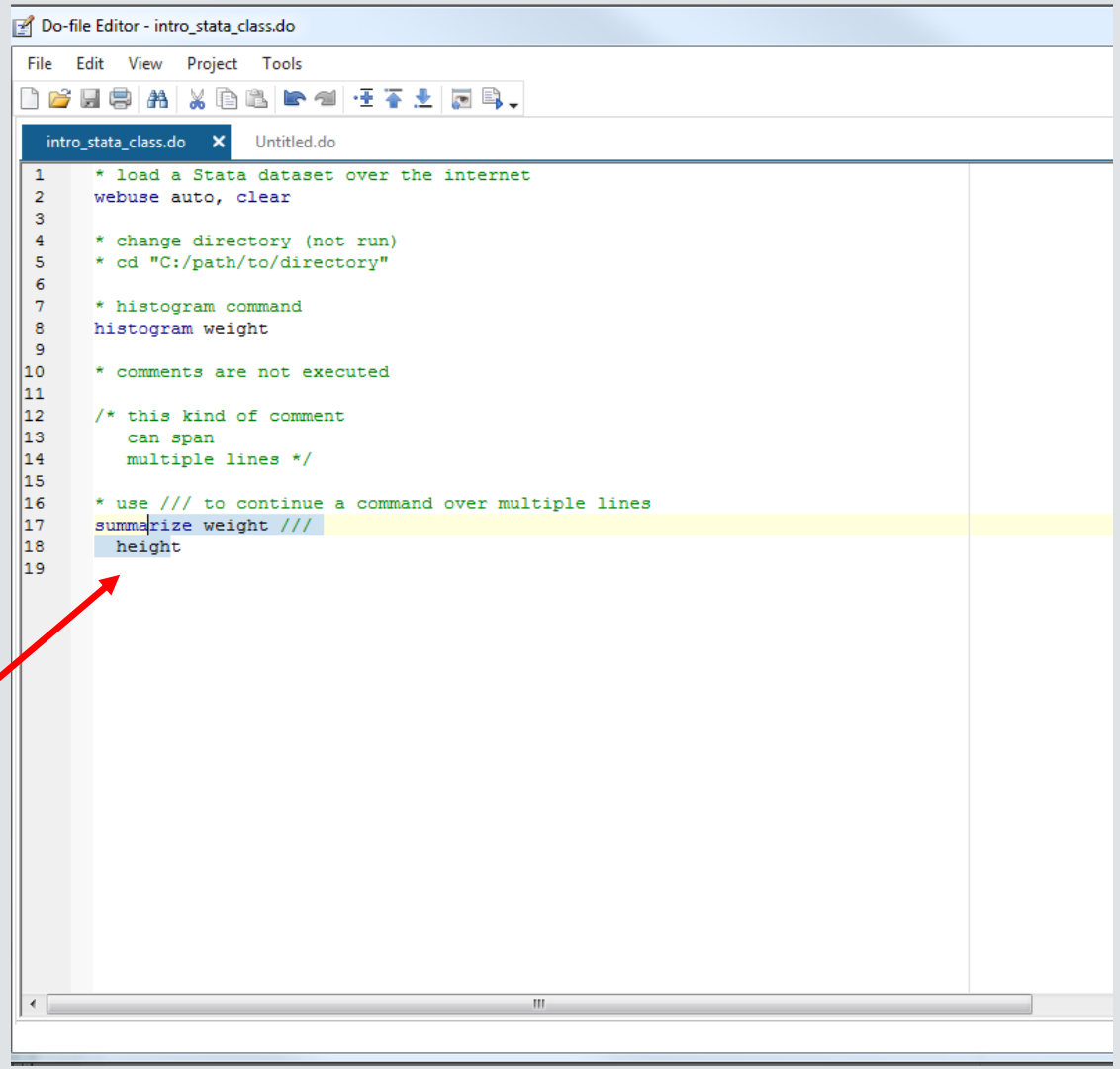
LONG LINES IN DO-FILES

Stata will normally assume that a newline signifies the end of a command

You can extend commands over multiple lines by placing /// at the end of each line except for the last

Make sure to put a space before ///

When executing, highlight each line in the command(s)



IMPORTING DATA

use	*load Stata dataset*
save	*save Stata dataset*
clear	*clear dataset from memory*
import excel	*import Excel dataset*
import delimited	*import delimited data (csv)*

STATA .dta FILES

- Data files stored in Stata's format are known as .dta files
  - Remember that coding files are "do-files" and usually have a .do extension
- Double clicking on a .dta file in Windows will open up a the data in a *new* instance of Stata (not in the current instance)
  - Be careful of having many Statas open

LOADING AND SAVING .dta FILES

- The command *use* loads Stata .dta files
  - Usually these will be stored on a hard drive, but .dta files can also be loaded over the internet (using a web address)
- Use the command *save* to save data in Stata's .dta format
  - The *replace* option will overwrite an existing file with the same name (without *replace*, Stata won't save if the file exists)
- The extension .dta can be omitted when using *use* and *save*

```
* read from hard drive; do not execute
use "C:/path/to/myfile.dta"

* load data over internet
use https://stats.idre.ucla.edu/stat/data/hs0

* save data, replace if it exists
save hs0, replace
```

CLEARING MEMORY

- Because Stata will only hold one data set in memory at a time, memory must be cleared before new data can be loaded
- The *clear* command removes the dataset from memory
- Data import commands like *use* will often have a *clear* option which clears memory before loading the new dataset

```
* clear data from memory
clear

* load data but clear memory first
use https://stats.idre.ucla.edu/stat/data/hs0, clear
```

IMPORTING EXCEL DATA SETS

- Stata can read in data sets stored in many other formats
- The command *import excel* is used to import Excel data
  - An Excel filename is required (with path, if not located in working directory) after the keyword *using*
- Use the *sheet()* option to open a particular sheet
- Use the *firstrow* option if variable names are on the first row of the Excel sheet

```
* import excel file; change path below before executing
import excel using "C:\path\myfile.xlsx", sheet("Sheet1") firstrow clear

import excel using "C:\mydir\myfolder\data.xlsx", sheet("Sheet1") firstrow
clear
```

IMPORTING .csv DATA SETS

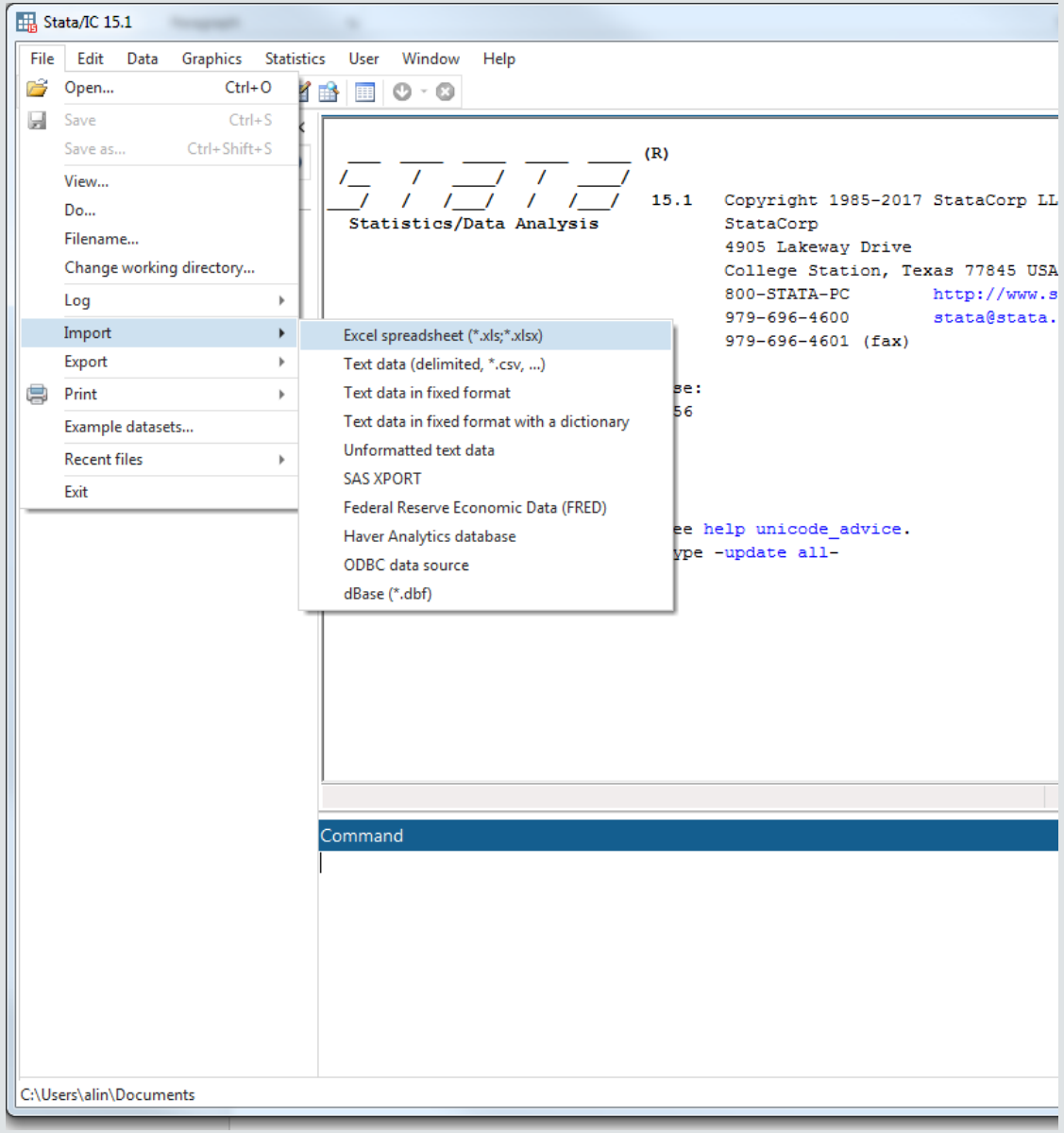
- Comma-separated values files are also commonly used to store data
- Use *import delimited* to read in .csv files (and files delimited by other characters such as tab or space)
- The syntax and options are very similar to *import excel*
  - But no need for *sheet()* or *firstrow* options (first row is assumed to be variable names in .csv files)

```
* import csv file; change path below before executing
import delimited using "C:\path\myfile.csv", clear
```

USING THE MENU TO IMPORT EXCEL AND .CSV DATA

Because path names can be very long and many options are often needed, menus are often used to import data

Select File -> Import and then either
"Excel spreadsheet" or
"Text data(delimited,*.csv, …)"



PREPARING DATA FOR IMPORT

- To get data into Stata cleanly, make sure the data in your Excel file or .csv file have the following properties
  - Rectangular
    - Each column (variable) should have the same number of rows (observations)
    - No graphs, sums, or averages in the file
  - Missing data should be left as blank fields
    - Missing data codes like -999 are ok too (see command *mvdecode*)
  - Variable names should contain only alphanumeric characters or _ or .
  - Make as many variables numeric as possible
    - Many Stata commands will only accept numeric variables

HELP FILES AND STATA SYNTAX

```
help command    open help page for command
```

HELP FILES

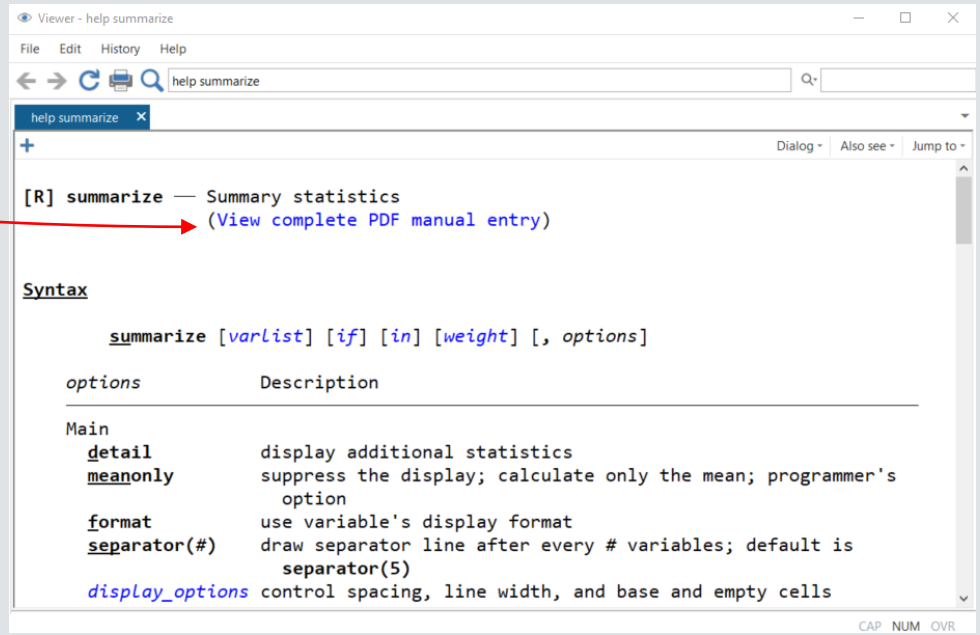
- Precede a command name (and certain topic names) with *help* to access its help file.

- Let's take a look at the help file for the *summarize* command.

```
*open help file for command summarize
help summarize
```

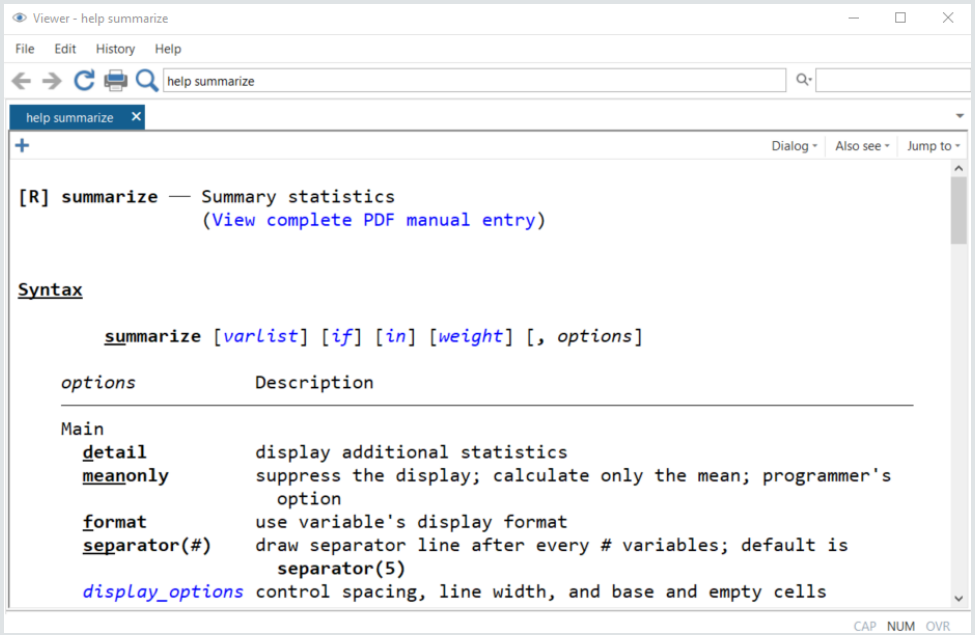
HELP FILE: TITLE SECTION

- command name and a brief description
- link to a .pdf of the Stata manual entry for *summarize*
  - manual entries include details about methods and formulas used for estimation commands, and thoroughly explained examples.



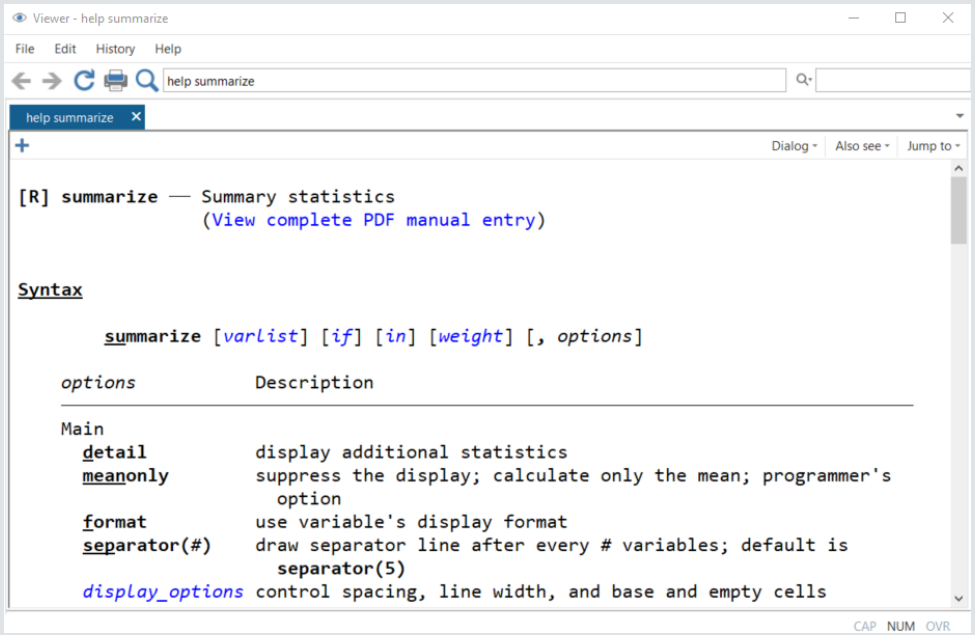
HELP FILE: SYNTAX SECTION

- various uses of command and how to specify them
- **bolded** words are required
- the underlined part of the command name is the minimal abbreviation of the command required for Stata to understand it
  - We can use `su` for `summarize`
- *italicized* words are to be substituted by the user
  - e.g. *varlist* is a list of one or more variables
- [Bracketed] words are optional (don't type the brackets)
- a comma , is almost always used to initiate the list of options



HELP FILE: OPTIONS SECTION

- Under the syntax section, we find the list of *options* and their description
- Most Stata commands come with a variety of options that alter how they process the data or how they output
- Options will typically follow a comma
- Options can also be abbreviated



HELP FILE: SYNTAX SECTION

- Summary statistics for all variables

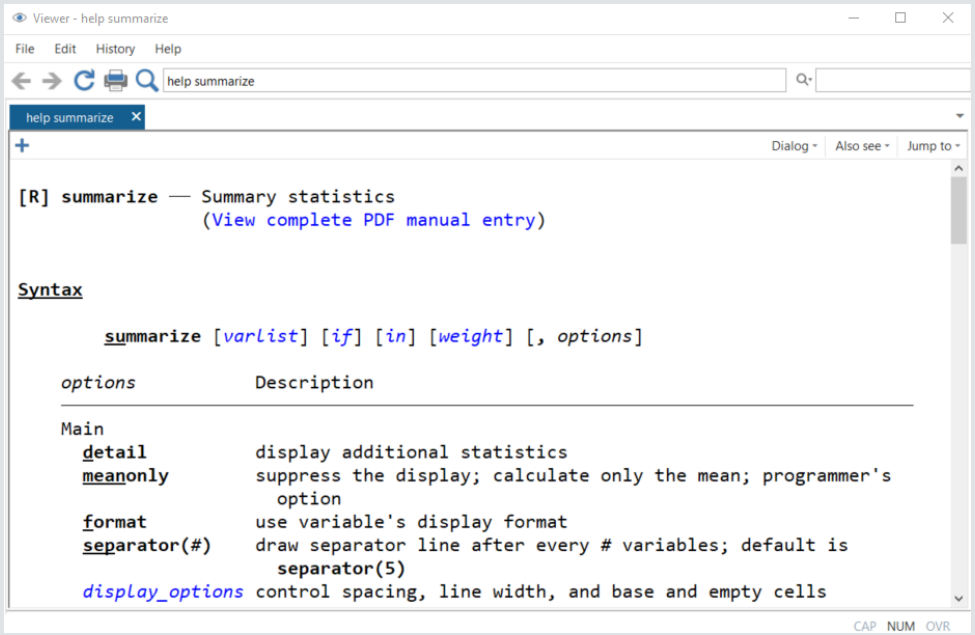
```
summarize
```

- Summary statistics for just variables read and write (using abbreviated command)

```
summ read write
```

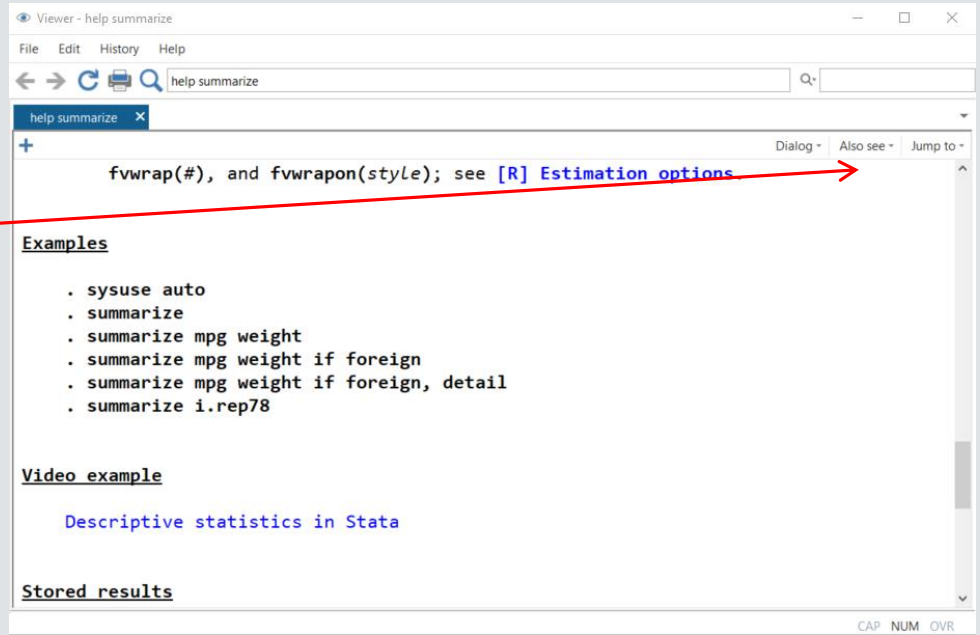
- Provide additional statistics for variable read

```
summ read, detail
```



HELP FILE: THE REST

- Below *options* are **Examples** of using the command, including video examples! (occasionally)
- Click on "Also see" to open help files of related commands



GETTING TO KNOW YOUR DATA

VIEWING DATA

```
browse   open spreadsheet of data

list     print data to Stata console
```

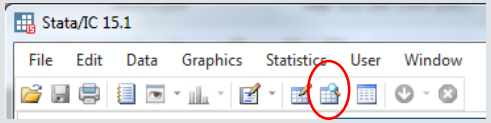
WORKSHOP DATASET

- We will use a dataset consisting of 200 observations (rows) and 13 variables (columns)
- Each observation is a student
- Variables
  - Demographics – gender(1=male, 2=female), race, ses(low, middle, high), etc
  - Academic test scores
    - read, write, math, science, socst
- Go ahead and load the dataset!

```
* Workshop dataset
use https://stats.idre.ucla.edu/stat/data/hs0, clear
```

BROWSING THE DATASET

- Once the data are loaded, we can view the dataset as a spreadsheet using the command *browse*
- The magnifying glass with spreadsheet icon also browses the dataset



- Black columns are numeric, red columns are strings, and blue columns are numeric with string labels

Data Editor (Browse) - [hs0.dta]

File Edit View Data Tools

gender[1] 1

	gender	id	race	ses	schtyp	prgtype	read	write	math
1	1	70	white	low	1	general	57	52	41
2	2	121	white	middle	1	vocati	68	59	53
3	1	86	white	high	1	general	44	33	54
4	1	141	white	high	1	vocati	63	44	47
5	1	172	white	middle	1	academic	47	52	57
6	1	113	white	middle	1	academic	44	52	51
7	1	50	african-amer	middle	1	general	50	59	42
8	1	11	hispanic	middle	1	academic	34	46	45
9	1	84	white	middle	1	general	63	57	54
10	1	48	african-amer	middle	1	academic	57	55	52
11	1	75	white	middle	1	vocati	60	46	51
12	1	60	5	middle	1	academic	57	65	51
13	1	95	white	high	1	academic	73	60	71
14	1	104	white	high	1	academic	54	63	57
15	1	38	african-amer	low	1	academic	45	57	50
16	1	115	white	low	1	general	42	49	43
17	1	76	white	high	1	academic	47	52	51
18	1	195	white	middle	2	general	57	57	60
19	1	114	white	high	1	academic	68	65	62
20	1	85	white	middle	1	general	55	39	57
21	1	167	white	middle	1	general	63	49	35

LISTING OBSERVATIONS

- The *list* command prints observation to the Stata console
- Simply issuing "*list*" will list all observations and variables
  - Not usually recommended except for small datasets
- Specify variable names to list only those variables
- We will soon see how to restrict to certain observations

```
* list read and write for first 5 observations
li read write in 1/5
```

```
     +--------------+
     | read   write |
     |--------------|
  1. |   57      52 |
  2. |   68      59 |
  3. |   44      33 |
  4. |   63      44 |
  5. |   47      52 |
     +--------------+
```

SELECTING OBSERVATIONS

```
in      select by observation number

if      select by condition
```

SELECTING BY OBSERVATION NUMBER WITH *in*

- Many commands are run on a subset of the data set observations
- *in* selects by observation (row) number
- Syntax
  - *in firstobs/lastobs*
    - *30/100* – observations 30 through 100
  - Negative numbers count from the end
  - "*L*" means last observation
    - *-10/L* – tenth observation from the last through last observation

```
* list science for last 3 observations
li science in -3/L
```

```
     +---------+
     | science |
     |---------|
198. |      55 |
199. |      58 |
200. |      53 |
     +---------+
```

SELECTING BY CONDITION WITH *if*

- *if* selects observations that meet a certain condition
  - gender == 1 (male)
  - math > 50
- *if* clause usually placed after the command specification, but before the comma that precedes the list of options

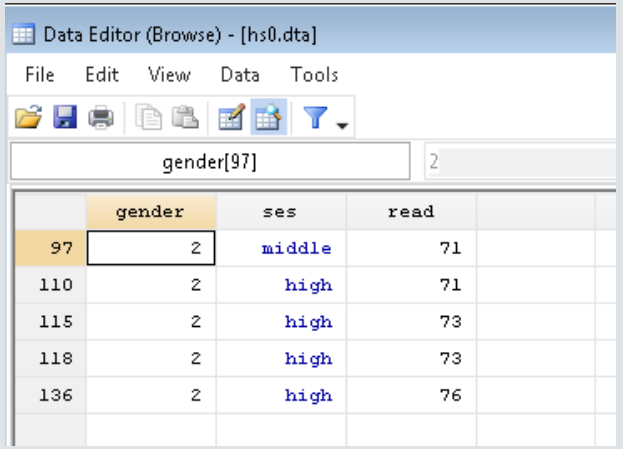
```
* list gender, ses, and math if math > 70
* with clean output
li gender ses math if math > 70, clean
```

	gender	ses	math
13.	1	high	71
22.	1	middle	75
37.	1	middle	75
55.	1	middle	73
73.	1	middle	71
83.	1	middle	71
97.	2	middle	72
98.	2	high	71
132.	2	low	72
164.	2	low	72

STATA LOGICAL AND RELATIONAL OPERATORS

- == equal to
  - double equals used to check for equality
- <, >, <=, >= greater than, greater than or equal to, less than, less than or equal to
- ! not
  - != not equal
- & and
- | or

```
* browse gender, ses, and read
*  for females (gender=2) who have read > 70
browse gender ses read if gender == 2 & read > 70
```



	gender	ses	read
97	2	middle	71
110	2	high	71
115	2	high	73
118	2	high	73
136	2	high	76

EXERCISE 1

- Use the `browse` command to examine the ses values for students with write score greater than 65
- Then, use the help file for the `browse` command to rewrite the command to examine the ses values *without labels*.

- Answers to exercises are at the bottom of the workshop do-file

EXPLORING DATA

codebook	*inspect variable values*
summarize	*summarize distribution*
tabulate	*tabulate frequencies*

EXPLORE YOUR DATA BEFORE ANALYSIS

- Take the time to explore your data set before embarking on analysis
- Get to know your sample with quick summaries of variables
  - Demographics of subjects
  - Distributions of key variables
- Look for possible errors in variables

USE *codebook* TO INSPECT VARIABLE VALUES

For more detailed information about the values of each variable, use codebook, which provides the following:

- For all variables
  - number of unique and missing values
- For numeric variables
  - range, quantiles, means and standard deviation for continuous variables
  - frequencies for discrete variables
- For string variables
  - frequencies
  - warnings about leading and trailing blanks

```
* inspect values of variables read gender and prgtype
codebook read gender prgtype
```

```
-------------------------------------------------------------------------------------------------------------------
read                                                                                                reading score
-------------------------------------------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [28,76]                      units:  1
        unique values:  30                        missing .:  0/200

                  mean:    52.23
             std. dev:   10.2529

           percentiles:        10%       25%       50%       75%       90%
                                39        44        50        60        67

-------------------------------------------------------------------------------------------------------------------
gender                                                                                               (unlabeled)
-------------------------------------------------------------------------------------------------------------------

                  type:  numeric (float)

                 range:  [1,2]                        units:  1
        unique values:  2                         missing .:  0/200

            tabulation:  Freq.  Value
                          91  1
                         109  2

-------------------------------------------------------------------------------------------------------------------
prgtype                                                                                              (unlabeled)
-------------------------------------------------------------------------------------------------------------------

                  type:  string (str8)

        unique values:  3                         missing "":  0/200

            tabulation:  Freq.  Value
                         105  "academic"
                          45  "general"
                          50  "vocati"
```

SUMMARIZING CONTINUOUS VARIABLES

- The *summarize* command calculates a variable's:
  - number of non-missing observations
  - mean
  - standard deviation
  - min and max

```
* summarize continuous variables
summarize read math
```

Variable	Obs	Mean	Std. Dev.	Min	Max
read	200	52.23	10.25294	28	76
math	200	52.645	9.368448	33	75

```
* summarize read and math for females
summarize read math if gender == 2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
read	109	51.73394	10.05783	28	76
math	109	52.3945	9.151015	33	72

DETAILED SUMMARIES

- Use the *detail* option with *summary* to get more estimates that characterize the distribution, such as:
  - percentiles (including the median at 50th percentile)
  - variance
  - skewness
  - kurtosis

```
* detailed summary of read for females
summarize read if gender == 2, detail
```

```
                        reading score
-------------------------------------------------------------
      Percentiles      Smallest
 1%           34            28
 5%           36            34
10%           39            34       Obs                 109
25%           44            35       Sum of Wgt.         109

50%           50                     Mean           51.73394
                      Largest       Std. Dev.      10.05783
75%           57            71
90%           68            73       Variance        101.16
95%           68            73       Skewness       .3234174
99%           73            76       Kurtosis       2.500028
```

TABULATING FREQUENCIES OF CATEGORICAL VARIABLES

- *tabulate* (often shortened to *tab*) displays counts of each value of a variable
  - useful for variables with a limited number of levels
- For variables with labeled values, use the *nolabel* option to display the underlying numeric values

```
* tabulate frequencies of ses
tabulate ses
```

```
        ses |      Freq.     Percent        Cum.
------------+-----------------------------------
        low |         47       23.50       23.50
     middle |         95       47.50       71.00
       high |         58       29.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

```
* remove labels
tab ses, nolabel
```

```
        ses |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         47       23.50       23.50
          2 |         95       47.50       71.00
          3 |         58       29.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

TWO-WAY TABULATIONS

- `tabulate` can also calculate the joint frequencies of two variables
- Use the `row` and `col` options to display row and column percentages
- We may have found an error in a race value (5?)

```
* with row percentages
tab race ses, row
```

race	ses: low	ses: middle	ses: high	Total
hispanic	9	11	4	24
	37.50	45.83	16.67	100.00
asian	3	5	3	11
	27.27	45.45	27.27	100.00
african-amer	11	6	3	20
	55.00	30.00	15.00	100.00
white	24	71	48	143
	16.78	49.65	33.57	100.00
5	0	2	0	2
	0.00	100.00	0.00	100.00
Total	47	95	58	200
	23.50	47.50	29.00	100.00

EXERCISE 2

- Use the *tab* command to determine the numeric code for "Asians" in the race variable
- Then use *summarize* to estimate the mean of the variable science for Asians

DATA VISUALIZATION

histogram	*histogram*
graph box	*boxplot*
scatter	*scatter plot*
graph bar	*bar plots*
twoway	*layered graphics*

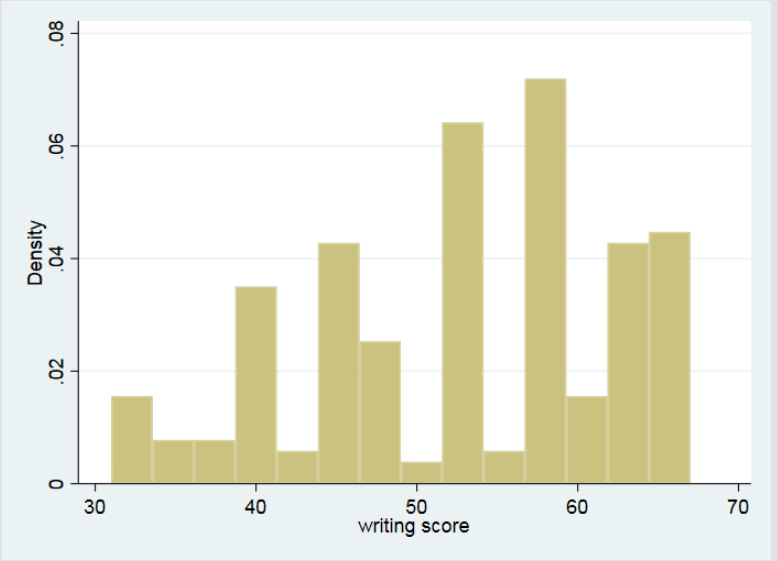
DATA VISUALIZATION

- Data visualization is the representation of data in visual formats such as graphs
  - Graphs help us to gain information about the distributions of variables and relationships among variables quickly through visual inspection
- Graphs can be used to explore your data, to familiarize yourself with distributions and associations in your data
- Graphs can also be used to present the results of statistical analysis

HISTOGRAMS

- Histograms plot distributions of variables by displaying counts of values that fall into various intervals of the variable

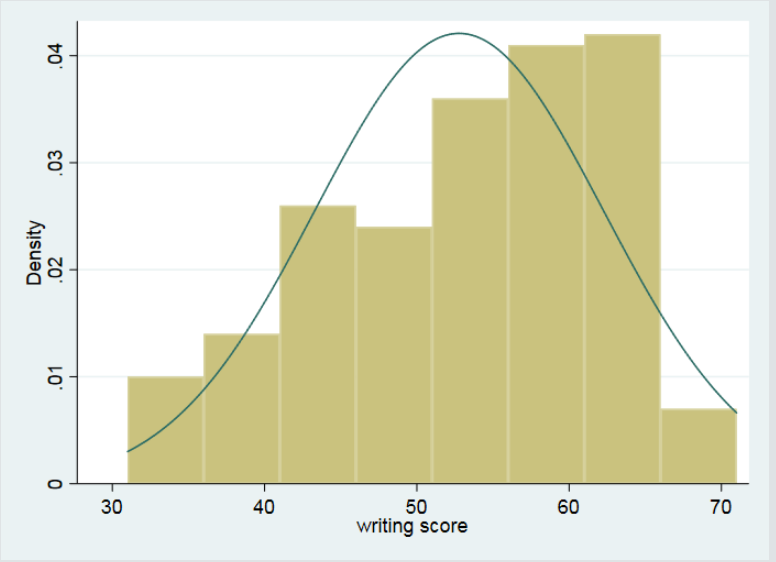
```
*histogram of write
histogram write
```



histogram OPTIONS *

- Use the option *normal* with *histogram* to overlay a theoretical normal density
- Use the *width()* option to specify interval width

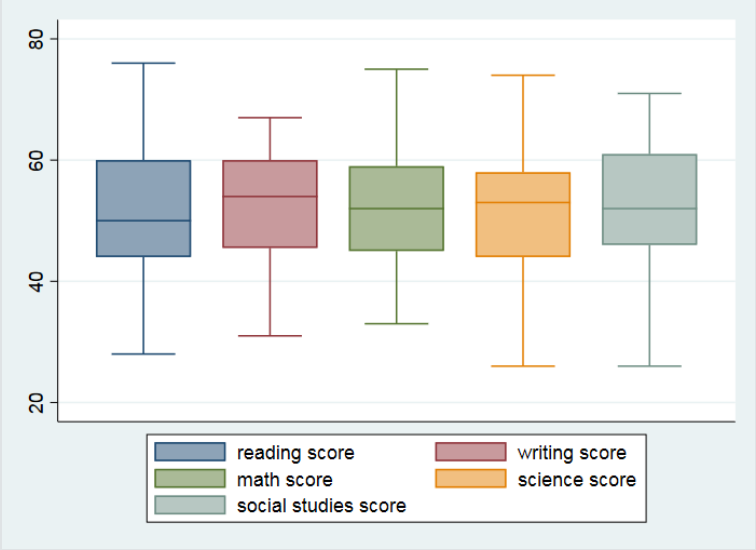
```
* histogram of write with normal density
*  and intervals of length 5
hist write, normal width(5)
```



BOXPLOTS *

- Boxplots are another popular option for displaying distributions of continuous variables
- They display the median, the interquartile range, (IQR) and outliers (beyond 1.5*IQR)
- You can request boxplots for multiple variables on the same plot

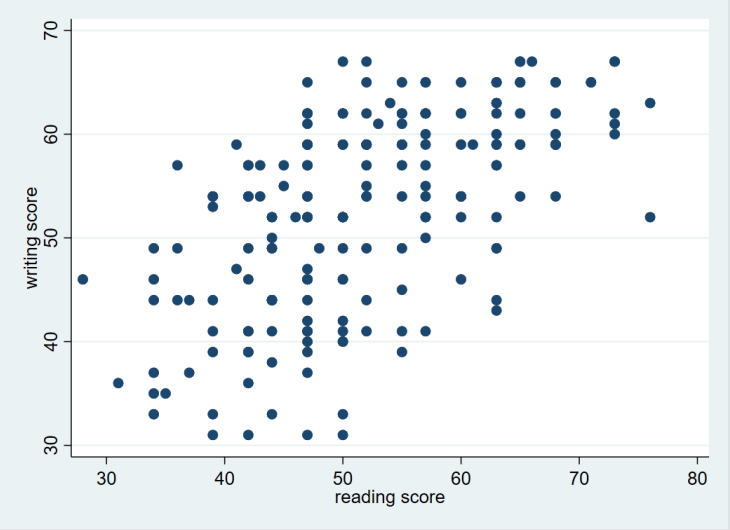
```
* boxplot of all test scores
graph box read write math science socst
```



SCATTER PLOTS

- Explore the relationship between 2 continuous variables with a scatter plot
- The syntax `scatter var1 var2` will create a scatter plot with `var1` on the y-axis and `var2` on the x-axis

```
* scatter plot of write vs read
scatter write read
```

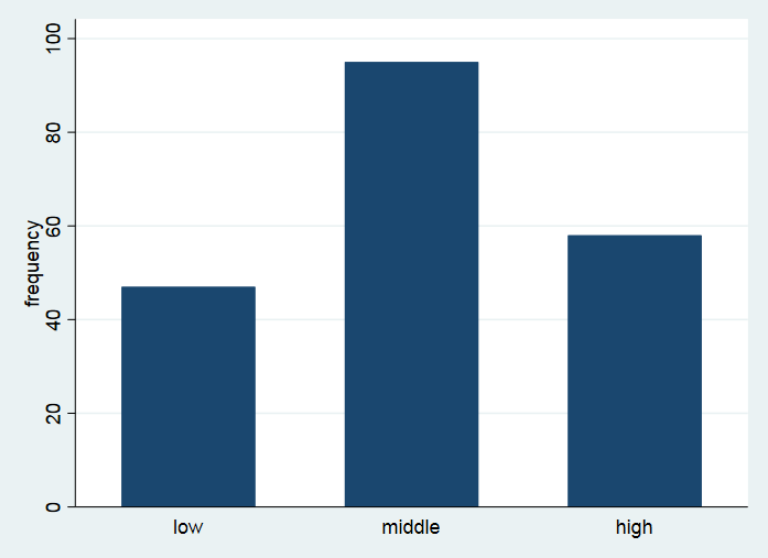


BAR GRAPHS TO VISUALIZE FREQUENCIES

- Bar graphs are often used to visualize frequencies
- `graph bar` produces bar graphs in Stata
  - its syntax is a bit tricky to understand
- For displays of frequencies (counts) of each level of a *variable*, use this syntax:

```
graph bar (count), over(variable)
```

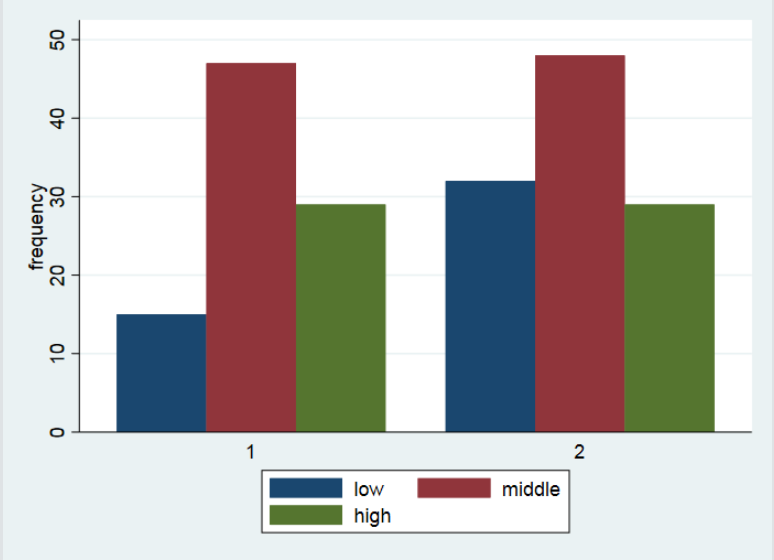
```
* bar graph of count of ses
graph bar (count), over(ses)
```



TWO-WAY BAR GRAPHS

- Multiple *over(variable)* options can be specified
- The option *asyvars* will color the bars by the first *over()* variable

```
* frequencies of gender by ses
*   asyvars colors bars by ses
graph bar (count), over(ses) over(gender) asyvars
```



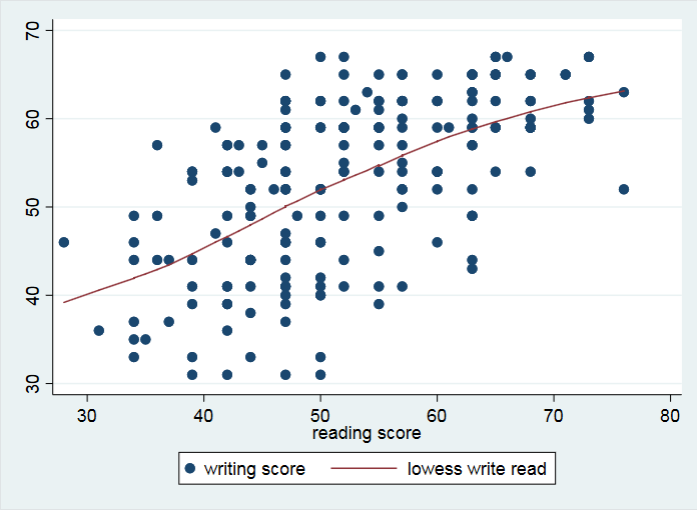
TWO-WAY, LAYERED GRAPHICS

- The Stata graphing command *twoway* produces layered graphics, where multiple plots can be overlayed on the same graph
- Each plot should involve a y-variable and an x-variable that appear on the y-axis and x-axis, respectively
  - Syntax (generally): *twoway (plottype1 yvar xvar) (plottype2 yvar xvar)…*
  - *plottype* is one of several types of plots available to *twoway*, and *yvar* and *xvar* are the variables to appear on the y-axis and x-axis
  - See *help twoway* for a list of the many *plottypes* available

LAYERED GRAPH EXAMPLE 1

- Layered graph of scatter plot and lowess plot (best fit curve)

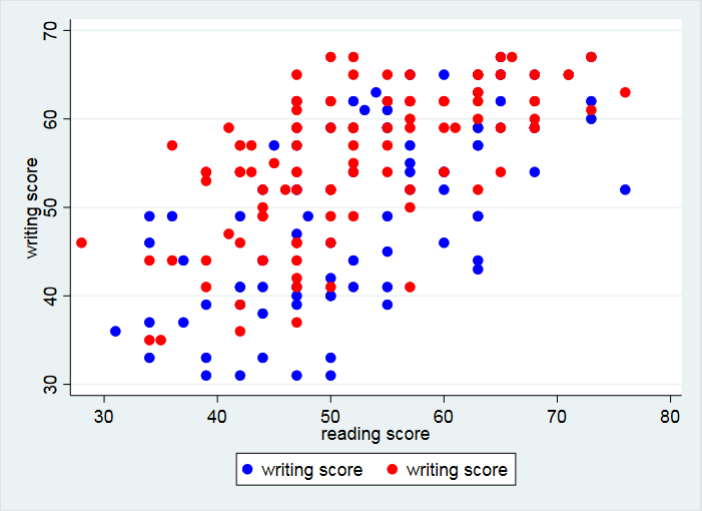
```
* layered graph of scatter plot and lowess curve
twoway (scatter write read) (lowess write read)
```



LAYERED GRAPH EXAMPLE 2

- You can also overlay separate plots by group to the same graph with different colors
  - Use *if* to select groups
  - the *mcolor()* option controls the color of the markers

```
* layered scatter plots of write and read
*   colored by gender
twoway (scatter write read if gender == 1, mcolor(blue)) ///
(scatter write read if gender == 2, mcolor(red))
twoway (scatter write read if female == 1, mcolor(blue))
(scatter write read if female == 0, mcolor(red))
```



EXERCISE 3

- Use the `scatter` command to create a scatter plot of math on the x-axis vs write on the y-axis
- Use the help file for `scatter` to change the **shape of the markers** to triangles.

DATA MANAGEMENT

CREATING, TRANSFORMING, AND LABELING VARIABLES

generate	*create variable*
replace	*replace values of variable*
egen	*extended variable generation*
rename	*rename variable*
recode	*recode variable values*
label variable	*give variable description*
label define	*generate value label set*
label value	*apply value labels to variable*
encode	*convert string variable to numeric*

GENERATING VARIABLES

- Variables often do not arrive in the form that we need
- Use *generate* (often abbreviated *gen* or *g*) to create variables, usually from operations on existing variables
  - sums/differences/products/means of variables
  - squares of variables
- If an input value to a generated variable is missing, the result will be missing

```
* generate a sum of 3 variables
generate total = math + science + socst
(5 missing values generated)

* it seems 5 missing values were generated
*  let's look at variables
summarize total math science socst
```

Variable	Obs	Mean	Std. Dev.	Min	Max
total	195	156.4564	24.63553	96	213
math	200	52.645	9.368448	33	75
science	195	51.66154	9.866026	26	74
socst	200	52.405	10.73579	26	71

MISSING VALUES IN STATA

- Missing numeric values in Stata are represented by .
- Missing string values in Stata are represented by “” (empty quotes)
- You can check for missing by testing for equality to . (or “” for string variables)
  - You can also use the missing() function
- When using estimation commands, generally, observations with missing on any variable used in the command will be dropped from the analysis

```
* list variables when science is missing
li math science socst if science == .

* same as above, using missing() function
li math science socst if missing(science)
```

```
     +-----------------------+
     | math   science   socst |
     |-----------------------|
  9. |   54         .      51 |
 18. |   60         .      56 |
 37. |   75         .      66 |
 55. |   73         .      66 |
 76. |   43         .      31 |
     +-----------------------+
```

REPLACING VALUES

- Use *replace* to replace values of existing variables
  - Often used with *if* to replace values for a subset of observations

```
* replace total with just (math+socst)
*  if science is missing
replace total = math + socst if science == .

* no missing totals now
summarize total
```

```
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       total |        200      155.42    25.47565        74        213
```

EXTENDED GENERATION OF VARIABLES

- *egen* (extended generate) creates variables using a wide array of functions, which include:
  - statistical functions that accept multiple variables as arguments
    - e.g. means across several variables
  - functions that accept a single variable, but do not involve simple arithmetic operations
    - e.g. standardizing a variable (subtract mean and divide by standard deviation)
- See the help file for *egen* to see a full list of available functions

```
* egen with function rowmean generates variable that
* is mean of all non-missing values of those
* variables
egen meantest = rowmean(read math science socst)
summarize meantest read math science socst
```

Variable	Obs	Mean	Std. Dev.	Min	Max
meantest	200	52.28042	8.400239	32.5	70.66666
read	200	52.23	10.25294	28	76
math	200	52.645	9.368448	33	75
science	195	51.66154	9.866026	26	74
socst	200	52.405	10.73579	26	71

```
* standardize read
egen zread = std(read)
summarize zread
```

Variable	Obs	Mean	Std. Dev.	Min	Max
zread	200	-1.84e-09	1	-2.363225	2.31836

g mean2= (read +math +science +socst)/4

RENAMING AND RECODING VARIABLES

- *rename* changes the name of a variable
  - Syntax: *rename old_name new_name*
- *recode* changes the values of a variable to another set of values
  - Syntax: *recode (old=new) (old=new)…*
- Here we will change the gender variable (1=male, 2=female) to "female" and will recode its values to (0=male, 1=female)
  - Thus, it will be clear what the coding of female signifies

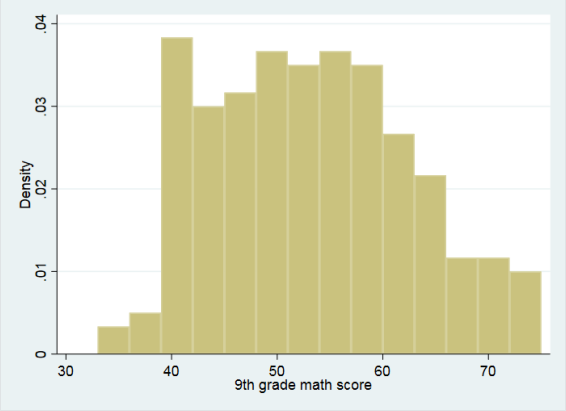
```
* renaming variables
rename gender female
* recode values to 0,1
recode female (1=0)(2=1)
tab female
```

```
     female |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         91       45.50       45.50
          1 |        109       54.50      100.00
------------+-----------------------------------
      Total |        200      100.00
```

LABELING VARIABLES (1)

- Short variable names make coding more efficient but can obscure the variable's meaning
- Use *label variable* to give the variable a longer description
- The variable label will sometimes be used in output and often in graphs

```
* labeling variables (description)
label variable math "9th grade math score"
label variable schtyp "public/private school"
* the variable label will be used in some output
histogram math
tab schtyp
```



LABELING VARIABLES (1)

- Short variable names make coding more efficient but can obscure the variable's meaning
- Use `label variable` to give the variable a longer description
- The variable label will sometimes be used in output and often in graphs

```
* labeling variables (description)
label variable math "9th grade math score"
label variable schtyp "public/private school"
* the variable label will be used in some output
histogram math
tab schtyp
```

```
public/priv |
 ate school |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        168       84.00       84.00
          2 |         32       16.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

LABELING VALUES

- Value labels give text descriptions to the numerical values of a variable.
- To create a new set of value labels use `label define`
  - Syntax: `label define labelname # label…,` where `labelname` is the name of the value label set, and `(# label…)` is a list of numbers, each followed by its label.
- Then, to apply the labels to variables, use `label values`
  - Syntax: `label values varlist labelname`, where `varlist` is one or more variables, and `labelname` is the value label set name

```
* schtyp before labeling values
tab schtyp
```

```
public/priv |
 ate school |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        168       84.00       84.00
          2 |         32       16.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

```
* create and apply labels for schtyp
label define pubpri 1 public 2 private
label values schtyp pubpri
tab schtyp
```

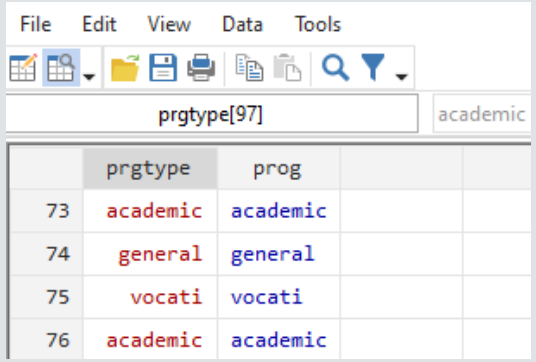
```
public/priv |
 ate school |      Freq.     Percent        Cum.
------------+-----------------------------------
     public |        168       84.00       84.00
    private |         32       16.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

ENCODING STRING VARIABLES INTO NUMERIC (1)

- *encode* converts a string variable into a numeric variable
  - remember that some Stata commands require numeric variables
  - *encode* will use alphabetical order to order the numeric codes
  - *encode* will convert the original string values into a set of value labels
  - *encode* will create a new numeric variable, which must be specified in option *gen(varname)*

```
* encoding string prgtype into
*  numeric variable prog
encode prgtype, gen(prog)

* we see that prog is a numeric with labels (blue)
*  while the old variable prog is string (red)
browse prog prgtype
```



	prgtype	prog
73	academic	academic
74	general	general
75	vocati	vocati
76	academic	academic

ENCODING STRING VARIABLES INTO NUMERIC (2)

- remember to use the option *nolabel* to remove value labels from *tabulate* output
- Notice that numbering begins at 1

```
* we see labels by default in tab
tab prog
```

```
      prog |      Freq.     Percent        Cum.
-----------+-----------------------------------
  academic |        105       52.50       52.50
   general |         45       22.50       75.00
    vocati |         50       25.00      100.00
-----------+-----------------------------------
     Total |        200      100.00
```

```
* use option nolabel to remove the labels
tab prog, nolabel
```

```
      prog |      Freq.     Percent        Cum.
-----------+-----------------------------------
         1 |        105       52.50       52.50
         2 |         45       22.50       75.00
         3 |         50       25.00      100.00
-----------+-----------------------------------
     Total |        200      100.00
```

EXERCISE 4

- Use the *generate* and *replace* commands to create a variable called "highmath" that takes on the value 1 if math is greater than 60, and 0 otherwise
- Then use the *label define* command to create a set of value labels called "mathlabel", which labels the value 1 "high" and the value 0 "low"
- Finally, use the *label values* command to apply the "mathlabel" labels to the newly generated variable highmath. Use the *tab* command on highmath to check your results.

DATASET OPERATIONS

keep	*keep variables, drop others*
drop	*drop variables, keep others*
keep if	*keep observations, drop others*
drop if	*drop observations, keep others*
sort	*sort by variables, ascending*
gsort	*ascending and descending sort*

SAVE YOUR DATA BEFORE MAKING BIG CHANGES

- We are about to make changes to the dataset that cannot easily be reversed, so we should save the data before continuing

```
* save dataset, overwrite existing file
save hs1, replace
```

KEEPING AND DROPPING VARIABLES

- *keep* preserves the selected variables and drops the rest
  - Use *keep* if you want to remove most of the variables but keep a select few
- *drop* removes the selected variables and keeps the rest
  - Use *drop* if you want to remove a few variables but keep most of them

```
* drop variable prgtype from dataset
drop prgtype

* keep just id read and math
keep id read math
```

KEEPING AND DROPPING OBSERVATIONS

- Specify *if* after *keep* or *drop* to preserve or remove observations by condition
- To be clear, *keep if* and *drop if* select observations, while *keep* and *drop* select variables

```
* keep observation if reading > 40
keep if read > 40
summ read
```

Variable	Obs	Mean	Std. Dev.	Min	Max
read	178	54.23596	8.96323	41	76

```
* now drop if math outside range [30,70]
drop if math < 30 | math > 70
summ math
```

Variable	Obs	Mean	Std. Dev.	Min	Max
math	168	52.68452	8.118243	35	70

SORTING DATA (1)

- Use *sort* to order the observations by one or more variables
  - *sort var1 var2 var3*, for example, will sort first by *var1*, then by *var2*, then by *var3*, all in ascending order

```
* sorting
* first look at unsorted
li in 1/5
```

```
     +-------------------+
     |  id   read   math |
     |-------------------|
  1. |  70     57     41 |
  2. | 121     68     53 |
  3. |  86     44     54 |
  4. | 141     63     47 |
  5. | 172     47     57 |
     +-------------------+
```

SORTING DATA (2)

- Use *sort* to order the observations by one or more variables
  - *sort var1 var2 var3*, for example, will sort first by *var1*, then by *var2*, then by *var3*, all in ascending order

```
* now sort by read and then math
sort read math
li in 1/5
```

```
     +------------------+
     |  id   read   math |
     |------------------|
  1. |  37     41     40 |
  2. |  30     41     42 |
  3. | 145     42     38 |
  4. |  22     42     39 |
  5. | 124     42     41 |
     +------------------+
```

SORTING DATA (3) *

- Use *gsort* with *+* or *−* before each variable to specify ascending and descending order, respectively

```
* sort descending read then ascending math
gsort -read +math
li in 1/5
```

```
     +------------------+
     |  id   read   math |
     |------------------|
  1. |  61     76     60 |
  2. | 103     76     64 |
  3. |  34     73     57 |
  4. |  93     73     62 |
  5. |  95     73     71 |
     +------------------+
```

EXERCISE 5

- Reload the hs0 data set fresh using the following command:

```
use https://stats.idre.ucla.edu/stat/data/hs0, clear
```

- Subset the dataset to observations with write score greater than or equal to 60. Then remove all variables except for id and write. Save this as a Stata dataset called "highwrite"
- Reload the hs0 dataset, subset to observations with write score less than 60, remove all variables except id and write, and save this dataset as "lowwrite"
- Reload the hs0 dataset. Drop the write variable. Save this dataset as "nowrite".

COMBINING DATASETS

append	*add more observations*
merge	*add more variables, join by matching variable*

APPENDING DATASETS

- Datasets are not always complete when we receive them
  - multiple data collectors
  - multiple waves of data
- The *append* command combines datasets by stacking them row-wise, adding more observations of the same variables

APPENDING DATASETS

- Let's *append* together two of the datasets we just created in the previous exercise
- Begin with one of the datasets in memory
  - First load the "highwrite" dataset
- Then *append* the "lowwrite" dataset
  - Syntax: *append using dtaname*
    - *dtaname* is the name of the Stata data file to append
- Variables that appear in only one file will be filled with missing in observations from the other file

```
* first load highwrite
use highwrite, clear

* append lowwrite
append using lowwrite

* summarize write shows 200 observations and
write scores above and below 70
summ write
```

Variable	Obs	Mean	Std. Dev.	Min	Max
write	200	52.775	9.478586	31	67

MERGING DATASETS (1)

- To add a dataset of columns of variables to another dataset, we merge them
- In Stata terms, the dataset in memory is termed the master dataset
  - the dataset to be merged in is called the "using" dataset
- Observations in each dataset to be merged should be linked by an id variable
  - the id variable should uniquely identify observations in at least one of the datasets
  - If the id variable uniquely identifies observations in both datasets, Stata calls this a 1:1 merge
  - If the id variable uniquely identifies observations in only one dataset, Stata calls this a 1:m (or m:1) merge

MERGING DATASETS (2)

- Let's merge our dataset of id and write with the dataset "nowrite" using id as the merge variable
- merge syntax:
  - 1-to-1: `merge 1:1 idvar using dtaname`
  - 1-to-many: `merge 1:m idvar using dtaname`
  - many-to-1: `merge m:1 idvar using dtaname`
  - Note that `idvar` can be multiple variables used to match
- Let's try this 1-to-1 merge
- Stata will output how many observations were successfully and unsuccessfully merged

```
* merge in nowrite dataset using id to link
merge 1:1 id using nowrite
```

```
Result                           # of obs.
-----------------------------------------
not matched                             0
matched                               200  (_merge==3)
-----------------------------------------
```

BASIC STATISTICAL ANALYSIS

ANALYSIS OF CONTINUOUS, NORMALLY DISTRIBUTED OUTCOMES

mean	*means and confidence intervals*
ttest	*t-tests*
correlate	*correlation matrices*
regress	*linear regression*
predict	*model predictions*
test	*test of linear combinations of coefficients*

LOAD DATASET

- Please load the dataset hs1, which is dataset hs0 altered by our data management commands, using the following syntax:

```
use https://stats.idre.ucla.edu/stat/data/hs1, clear
```

MEANS AND CONFIDENCE INTERVALS (1)

- Confidence intervals express a range of plausible values for a population statistic, such as the mean of a variable, consistent with the sample data
- The *mean* command provides a 95% confidence interval, as do many other commands

```
* many commands provide 95% CI
mean read

Mean estimation                   Number of obs    =       200

--------------------------------------------------------------
             |       Mean   Std. Err.      [95% Conf. Interval]
-------------+------------------------------------------------
        read |      52.23   .7249921       50.80035    53.65965
--------------------------------------------------------------
```

PREVALENCE AND CONFIDENCE INTERVALS (1)

- Confidence intervals express a range of plausible values for a population statistic, such as the mean of a variable, consistent with the sample data
- The `proportion` command provides a 95% confidence interval, as do many other commands

```
proportion ses

Proportion estimation                 Number of obs    =       200
```

	Proportion	Std. Err.	[95% Conf.	Interval]
ses				
low	.235	.0300565	.1809417	.299307
middle	.475	.0353997	.4061244	.54484
high	.29	.0321663	.2308622	.3572503

T-TESTS TEST WHETHER THE MEANS ARE DIFFERENT BETWEEN 2 GROUPS

- t-tests test whether the mean of a variable is different between 2 groups
- The t-test assumes that the variable is normally distributed
- The independent samples t-test assumes that the two groups are independent (uncorrelated)
- Syntax for independent samples t-test:
  - `ttest var, by(groupvar)`, where `var` is the variable whose mean will be tested for differences between levels of `groupvar`
- The `ttest` command can also perform a paired-samples t-test, using slightly different syntax
- Let's peform a t-test to see if the means of write are different between the 2 genders

INDEPENDENT SAMPLES T-TEST EXAMPLE

```
* independent samples t-test
ttest read, by(female)
Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       0 |      91    52.82418    1.101403    10.50671    50.63605     55.0123
       1 |     109    51.73394    .9633659    10.05783    49.82439     53.6435
---------+--------------------------------------------------------------------
combined |     200        52.23    .7249921    10.25294    50.80035    53.65965
---------+--------------------------------------------------------------------
    diff |            1.090231    1.457507               -1.783998    3.964459
------------------------------------------------------------------------------
    diff = mean(0) - mean(1)                                      t =   0.7480
Ho: diff = 0                     degrees of freedom =      198

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.7723         Pr(|T| > |t|) = 0.4553          Pr(T > t) = 0.2277
```

INDEPENDENT SAMPLES T-TEST EXAMPLE

```
* independent samples t-test
ttest read, by(female)
Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       0 |      91    52.82418    1.101403    10.50671    50.63605     55.0123
       1 |     109    51.73394    .9633659    10.05783    49.82439     53.6435
---------+--------------------------------------------------------------------
combined |     200       52.23    .7249921    10.25294    50.80035    53.65965
---------+--------------------------------------------------------------------
    diff |            1.090231    1.457507               -1.783998    3.964459
------------------------------------------------------------------------------
    diff = mean(0) - mean(1)                                      t =   0.7480
Ho: diff = 0                     degrees of freedom =      198

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.7723         Pr(|T| > |t|) = 0.4553          Pr(T > t) = 0.2277
```

CORRELATION

- A correlation coefficient quantifies the linear relationship between two (continuous) variables on a scale between -1 and 1
- Syntax: *correlate varlist*
- The output will be a correlation matrix that shows the pairwise correlation between each pair of variables
- If you need p-values for correlations, use the command *pwcorr*

```
* correlation matrix of 5 variables
corr read write math science socst

(obs=195)

             |     read    write     math  science    socst
-------------+------------------------------------------------
        read |   1.0000
       write |   0.5960   1.0000
        math |   0.6492   0.6203   1.0000
     science |   0.6171   0.5671   0.6166   1.0000
       socst |   0.6175   0.5996   0.5299   0.4529   1.0000
```

MODEL ESTIMATION COMMAND SYNTAX

- Most model estimation commands in Stata use a standard syntax:

  model_command depvar indepvarlist, options

- Where
  - *model_command* is the name of a model estimation command
  - *depvar* is the name of the dependent variable (outcome)
  - *indepvarlist* is a list of independent variables (predictors)
  - *options* are options specific to that *command*

LINEAR REGRESSION

- Linear regression, or ordinary least squares regression, models the effects of one or more predictors, which can be continuous or categorical, on a normally-distributed outcome
- Syntax: *`regress depvar indepvarlist`*, where *`depvar`* is the name of the dependent variable, and *`indepvarlist`* is a list of independent variables
  - To be safe, precede independent variables names with *`i.`* to denote categorical predictors and *`c.`* to denote continuous predictors
  - For categorical predictors with the *`i.`* prefix, Stata will automatically create dummy 0/1 indicator variables and enter all but one (the first, by default) into the regression
- Let's run a linear regression of the dependent variable write predicted by independent variables math (continuous) and ses (categorical)

LINEAR REGRESSION EXAMPLE

```
* linear regression of write on continuous
*  predictor math and categorical predictor ses
regress write c.math i.ses
```

```
      Source |       SS           df       MS      Number of obs   =       200
-------------+----------------------------------   F(3, 196)       =     41.07
       Model |  6901.40673         3  2300.46891   Prob > F        =    0.0000
    Residual |  10977.4683       196  56.0074912   R-squared       =    0.3860
-------------+----------------------------------   Adj R-squared   =    0.3766
       Total |   17878.875       199   89.843593   Root MSE        =    7.4838

------------------------------------------------------------------------------
       write |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        math |   .6115218   .0588735    10.39   0.000      .495415    .7276286
             |
         ses |
      middle |  -.5499235   1.346566    -0.41   0.683    -3.205542    2.105695
        high |   1.014773    1.52553     0.67   0.507    -1.993786    4.023333
             |
       _cons |   20.54836   3.093807     6.64   0.000     14.44694    26.64979
------------------------------------------------------------------------------
```

ESTIMATING STATISTICS BASED ON A MODEL

- Stata provides excellent support for estimating and testing additional statistics after a regression model has been run
- Stata refers to these as "postestimation" commands, and they can be used after most regression models
  - To see which commands can be issued as follow-ups to a model estimation command, use:

    help model_command postestimation

    Where *model_command* is a Stata model command

    e.g. for *regress*, try *help regress postestimation*
- Examples: model predictions, joint tests of coefficients or linear combination of statistics, marginal estimates

POSTESTIMATION EXAMPLE 1: PREDICTION

- The *predict*: command can be used to make model-based predictions of various statistics such as:
  - Predicted value of dependent variable (default)
  - Residuals (difference between observed and predicted dependent variable)
    - Add option *residuals* to *predict*
  - Influence statistics
    - e.g. add option *cooksd* to *predict*

```
* predicted dependent variable
predict pred

* get residuals
predict res, residuals

* first 5 predicted values and residuals with
observed write
li pred res write in 1/5
```

```
     +-----------------------------+
     |     pred         res   write |
     |-----------------------------|
  1. |  45.62076    6.379242      52 |
  2. |   52.4091    6.590904      59 |
  3. |  54.58532   -21.58531      33 |
  4. |  50.30466   -6.304662      44 |
  5. |  54.85518   -2.855183      52 |
     +-----------------------------+
```

EXERCISE 6

- Use the *regress* command to determine if the variables female (categorical) and science (continuous) are predictive of the dependent variable math.
- One of the assumptions of linear regression is that the errors (estimated by residuals) are normally distributed. Use the *predict* command and the *histogram* command to assess this assumption.

ANALYSIS OF CATEGORICAL OUTCOMES

tab …, chi2	*chi-square test of independence*
logit	*logistic regression*

CHI-SQUARE TEST OF INDEPENDENCE

- The chi-square test of independence assesses association between 2 categorical variables
  - Answers the question: Are the category proportions of one variable the same across levels of another variable?
- Syntax: `tab var1 var2, chi2`

```
* chi square test of independence
tab prog ses, chi2
```

```
           |                ses
      prog |       low    middle      high |     Total
-----------+---------------------------------+----------
  academic |        19        44        42 |       105
   general |        16        20         9 |        45
   vocati |        12        31         7 |        50
-----------+---------------------------------+----------
     Total |        47        95        58 |       200

          Pearson chi2(4) =  16.6044   Pr = 0.002
```

LOGISTIC REGRESSION

- Logistic regression is used to estimate the effect of multiple predictors on a binary outcome
- Syntax very similar to *regress*: *logit depvar indepvarlist,* where *depvar* is a binary outcome variable and *indepvarlist* is a list of predictors
- Add the *or* option to output the coefficients as odds ratios
- Let's perform a logistic regression:
  - We will use the binary variable "highmath" that we created in exercise 4 as the outcome
  - The variables write (continuous) and ses (categorical) will serve as predictors

LOGISTIC REGRESSION EXAMPLE

```
*  logistic regression of binary outcome highmath predicted by
*    by continuous(write) and female (categorical)
logit highmath c.write i.female, or

Logistic regression                               Number of obs     =        200
                                                  LR chi2(2)        =      62.16
                                                  Prob > chi2       =     0.0000
Log likelihood = -74.300928                       Pseudo R2         =     0.2949

------------------------------------------------------------------------------
    highmath | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       write |   1.253272   .050584     5.59   0.000     1.157949    1.356442
   1.female |   .4330014   .1823694    -1.99   0.047     .1896638    .9885398
       _cons |   1.19e-06   2.82e-06    -5.76   0.000     1.14e-08    .0001237
------------------------------------------------------------------------------
```

EXERCISE 7

- Use the *tab* command to run a chi-square test of independence to test for association between ses and race.
- Fisher's exact test is often used in place of the chi-square test of independence when the (expected) cell sizes are small. Use the help file for *tabulate twoway* (which is just the *tabulate* command for 2 variables) to run a Fisher's exact test to test the association between ses and race. How does the p-value compare to the result of the chi-square test?

ADDITIONAL STATA MODELING COMMANDS

A FEW OF STATA'S ADDITIONAL REGRESSION COMMANDS

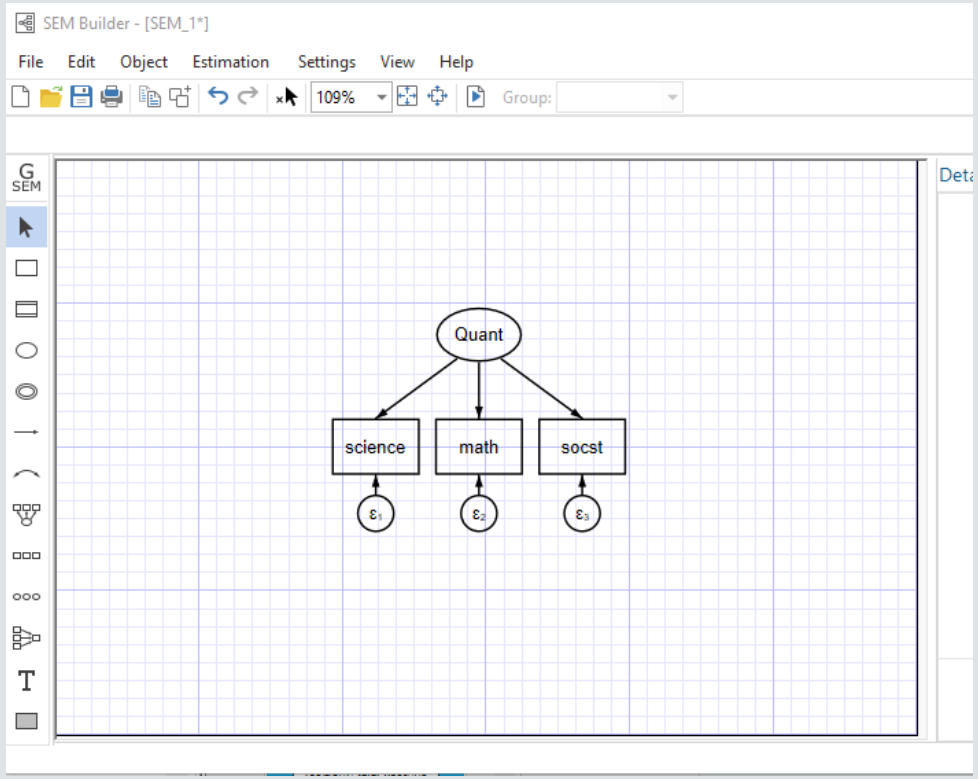
- *glm*: generalized linear model
- *ologit* and *mlogit*: ordinal logistic and multinomial logistic regression
- *poisson* and *nbreg*: poisson and negative binomial regression (count outcomes)
- *mixed* – mixed effects (multilevel) regression
- *meglm* – mixed effects generalized linear model
- *stcox* – Cox proportional hazards model
- *ivregress* – instrumental variable regression

STRUCTURAL EQUATION MODELING

- Stata features 2 ways to build a structural equation model (SEM)
  - Through syntax:

    `sem (Quant -> science math socst)`

  - And through the SEM Builder, accessible through the "Statistics menu" through Statistics>SEM (structural equation modeling)> Model building and estimation
- The *gsem* command is used for generalized SEM, which allows for non-normally distributed outcomes, multilevel models, and categorical latent variables, among other extensions



ADDITIONAL RESOURCES FOR LEARNING STATA

IDRE STATISTICAL CONSULTING WEBSITE

- The IDRE Statistical Consulting website is a well-known resource for coding support for several statistical software packages
  - https://stats.idre.ucla.edu
- Stata was beloved by previous members of the group, so Stata is particularly well represented on our website

IDRE STATISTICAL CONSULTING WEBSITE STATA PAGES

- On the website landing page for Stata, you'll find many links to our Stata resources pages
  - https://stats.idre.ucla.edu/stata/
- These resources include:
  - seminars, deeper dives into Stata topics that are often delivered live on campus
  - learning modules for basic Stata commands
  - data analysis examples of many different regression commands
  - annotated output of many regression commands

EXTERNAL RESOURCES

- Stata YouTube channel (run by StataCorp)
- Stata FAQ (compiled by StataCorp)
- Stata cheat sheets (compact guides to Stata commands)

END
THANK YOU!