INTRODUCTION TO STATA

UCLA OARC STATISTICAL METHODS AND DATA ANALYTICS

Purpose of the seminar

An introduction to using 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
- Outputting to Word and Excel

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

Why use Stata?

Command syntax is very compact

Syntax is consistent across commands

Competitive with other software regarding variety of statistical tools

Excellent documentation

Exceptionally strong support for

- Econometric models and methods
- Complex survey data analysis tools

Versions of Stata *

Flavors of Stata are BE SE and MP

- The main difference is the size of dataset allowed and speed of processing
  - BE ≤ SE ≤ MP
- For more information click here https://www.stata.com/products/which-stata-is-right-for-me/

See our webpage for more information about using Stata at UCLA

Navigating Stata's interface

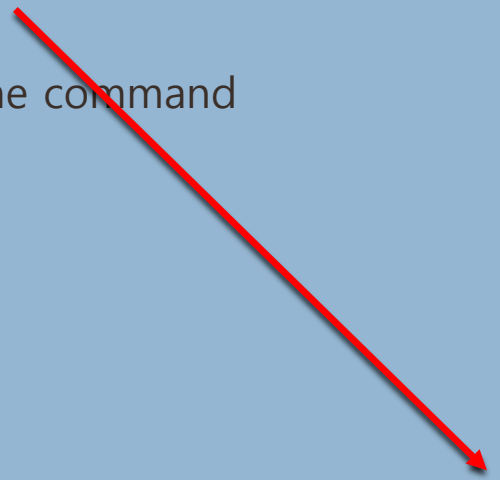
```
cd      change working directory
```

Command window

You can enter commands directly into the Command window

This command will load a Stata dataset over the internet

Go ahead and enter the command



Log Viewer Graph Do-file Editor Data

Results

```
  ___  ____  ____  ____  ____ ®
 /__    /   ____/   /   ____/      StataNow 19.5
___/   /   /___/   /   /___/       BE-Basic Edition

 Statistics and Data Science       Copyright 1985-2025 StataCorp LLC
                                   StataCorp
                                   4905 Lakeway Drive
                                   College Station, Texas 77845 USA
                                   800-782-8272        https://www.stata.com
                                   979-696-4600        service@stata.com

Stata license: Single-user , expiring 25 May 2026
Serial number: 301909300802
  Licensed to: Kotrina Kajokaite
               UCLA

Notes:
      1. Unicode is supported; see help unicode_advice.

.
```

Command

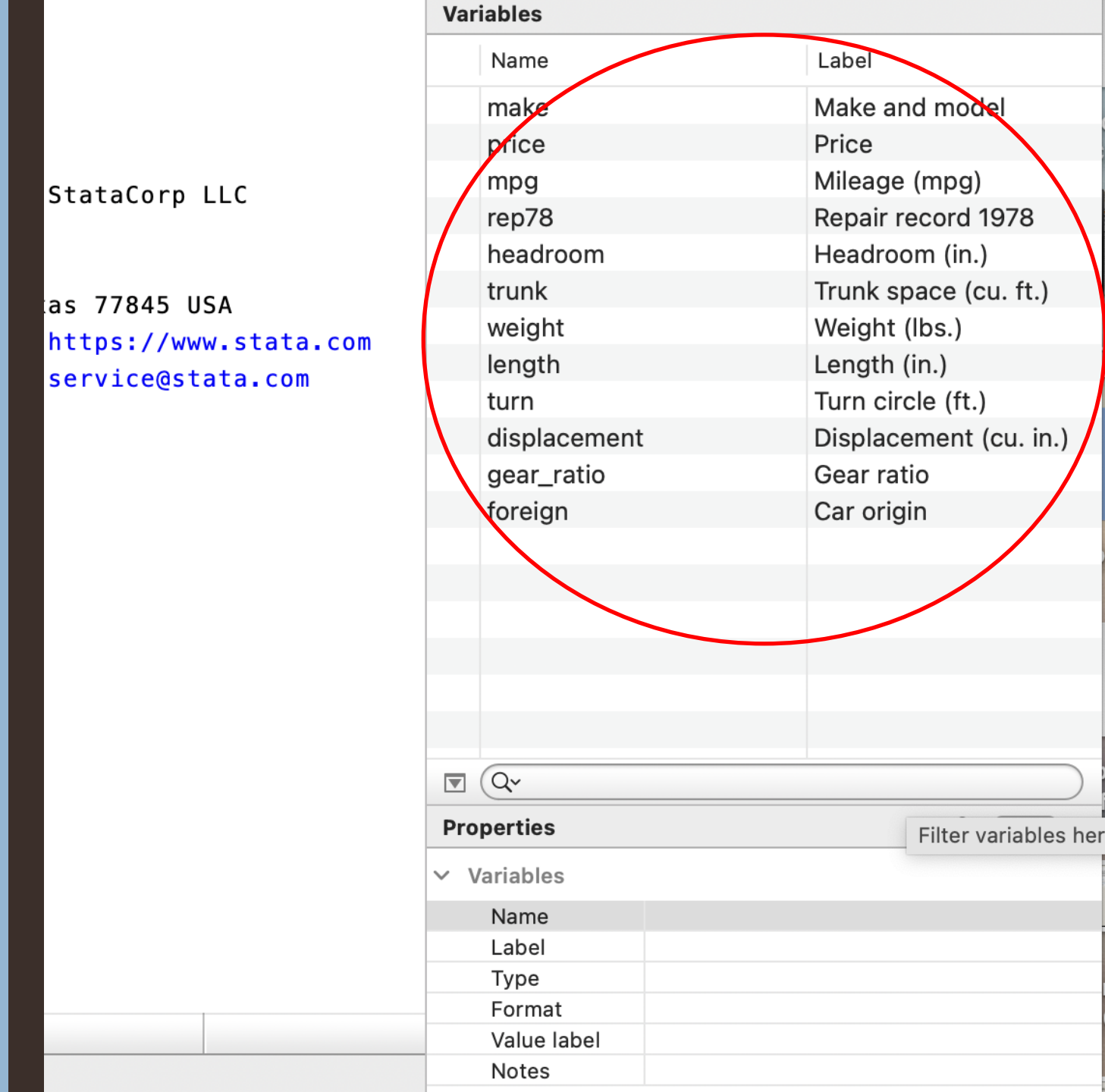
```
webuse auto, clear
```

Variables window

Once data are loaded, variables and their labels appear in the Variable window

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

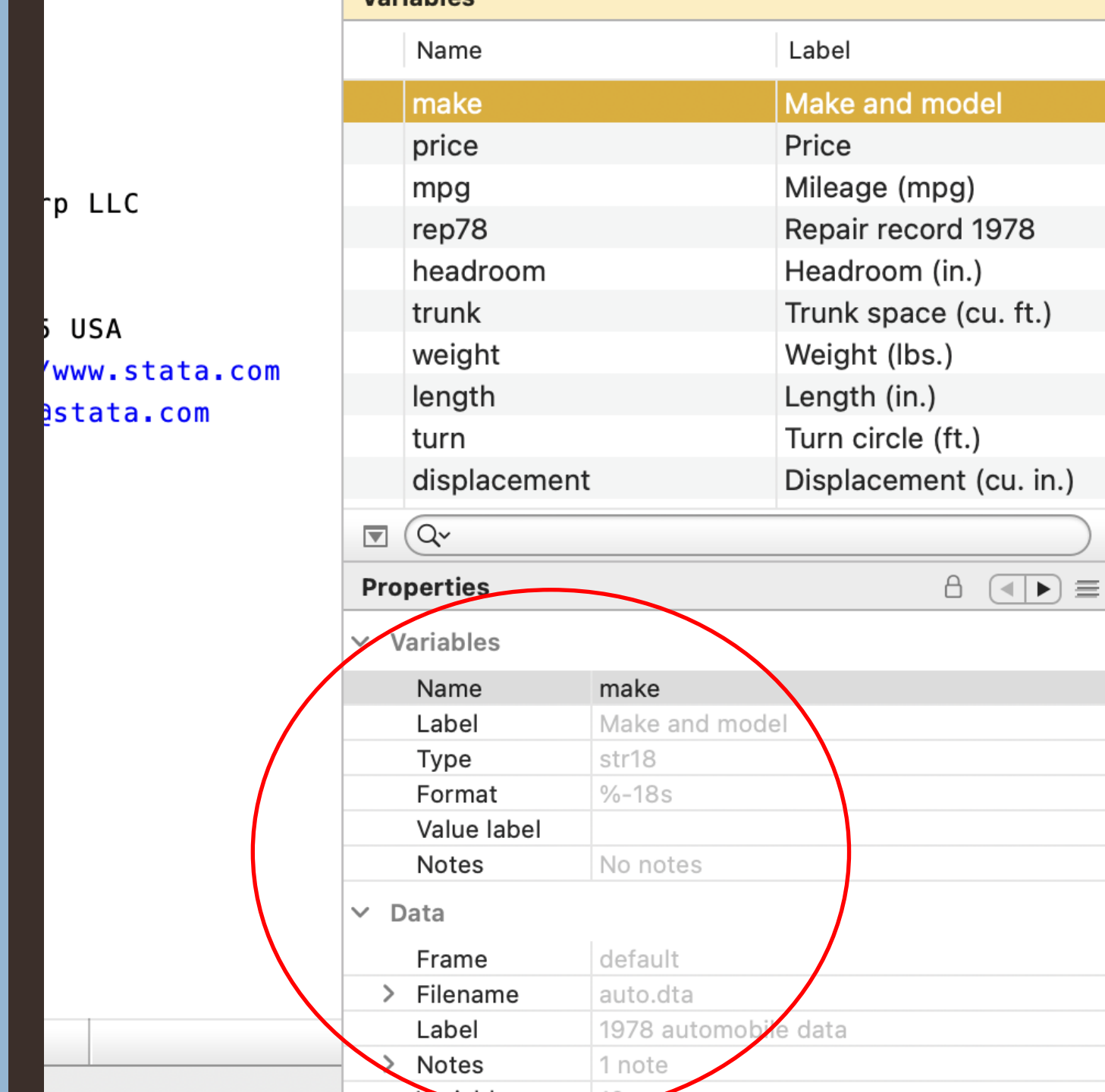
Double-clicking a variable name will cause it to appear in the Command Window



Properties window

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

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



History window

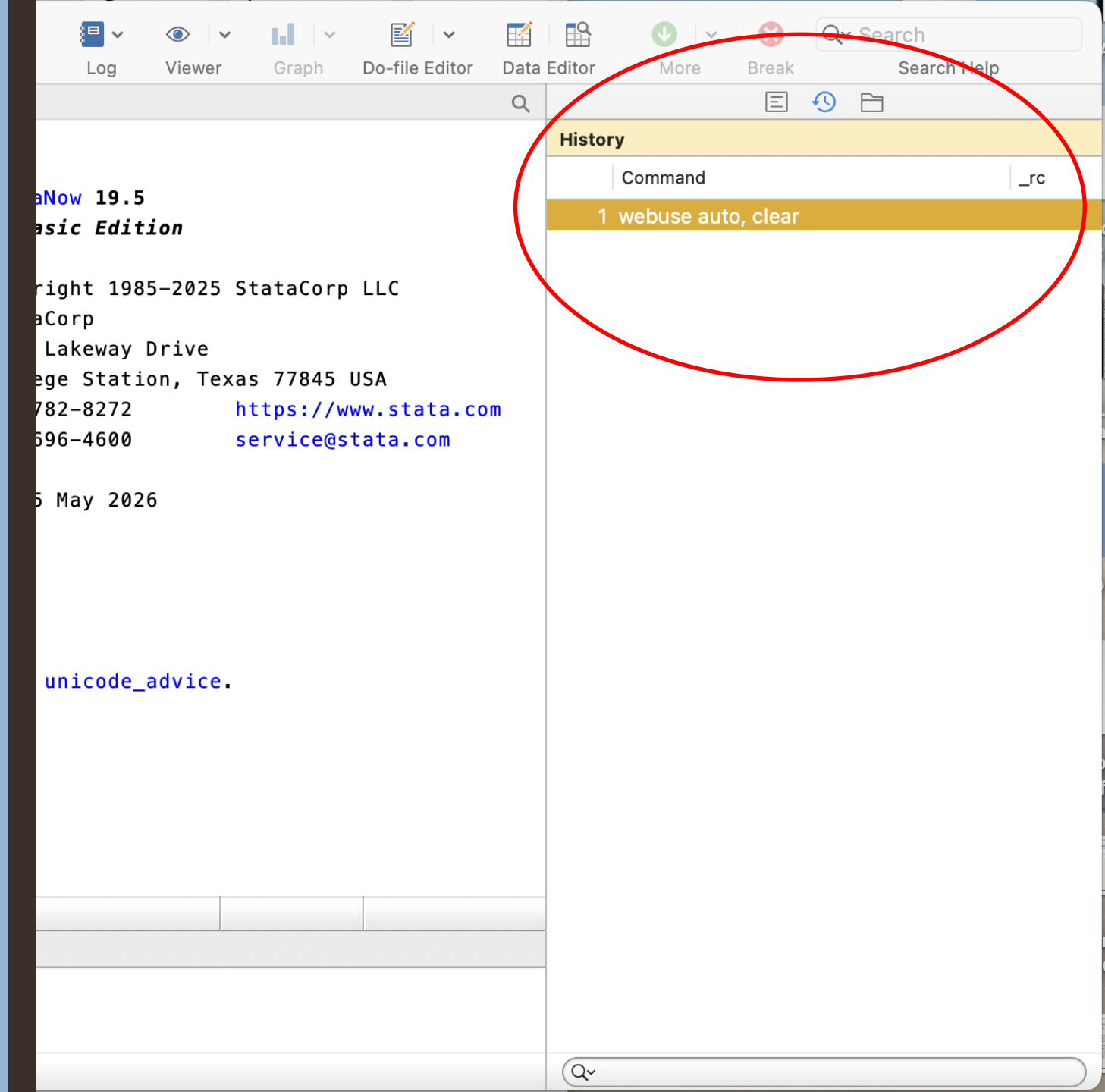
The History window lists previously issued commands

Successful commands will appear black

Unsuccessful commands will appear red

Double-click a command to run it again

Hitting PageUp will also recall previously used commands (Fn + PageUp on a Mac)

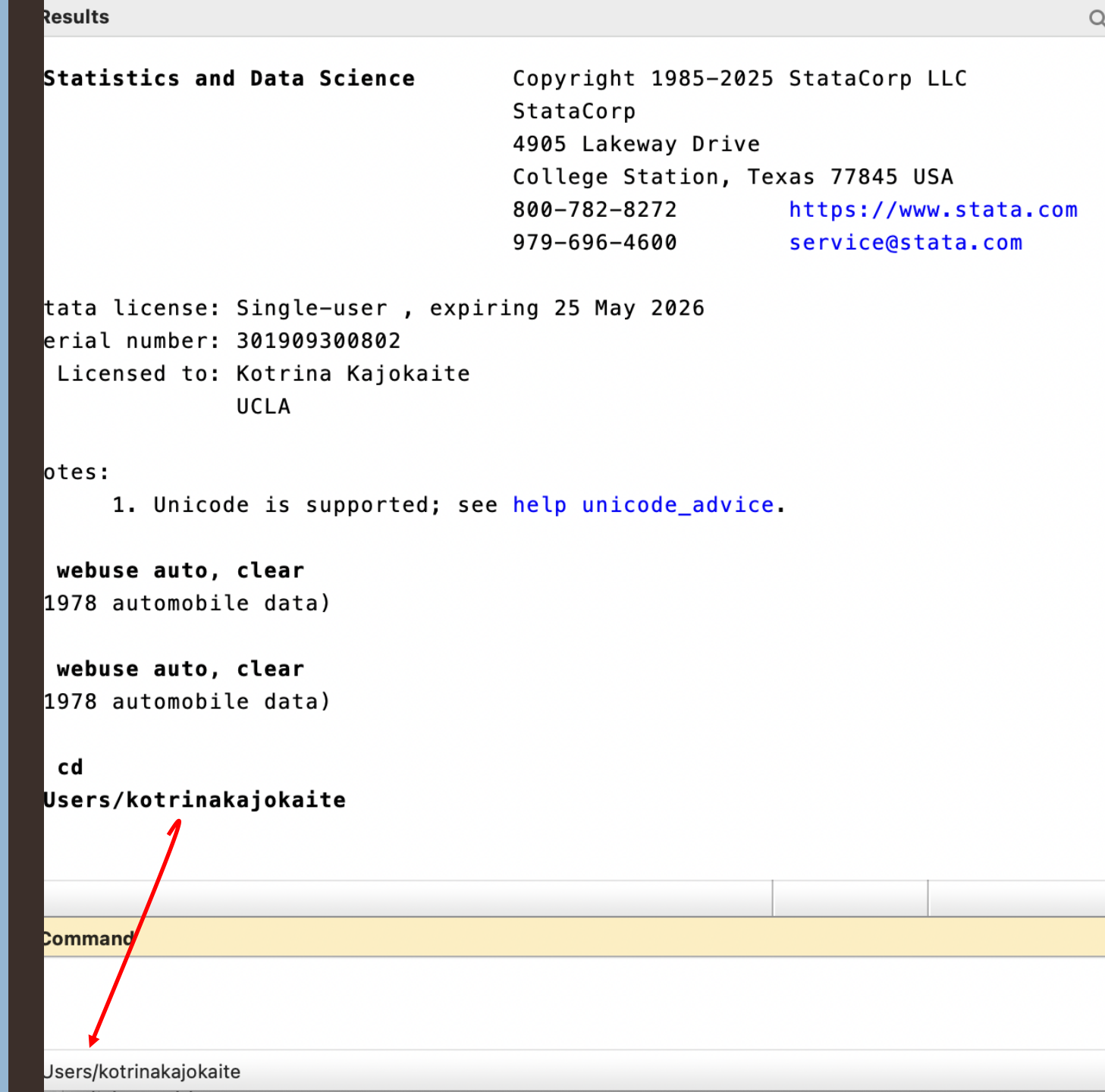


Working directory

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

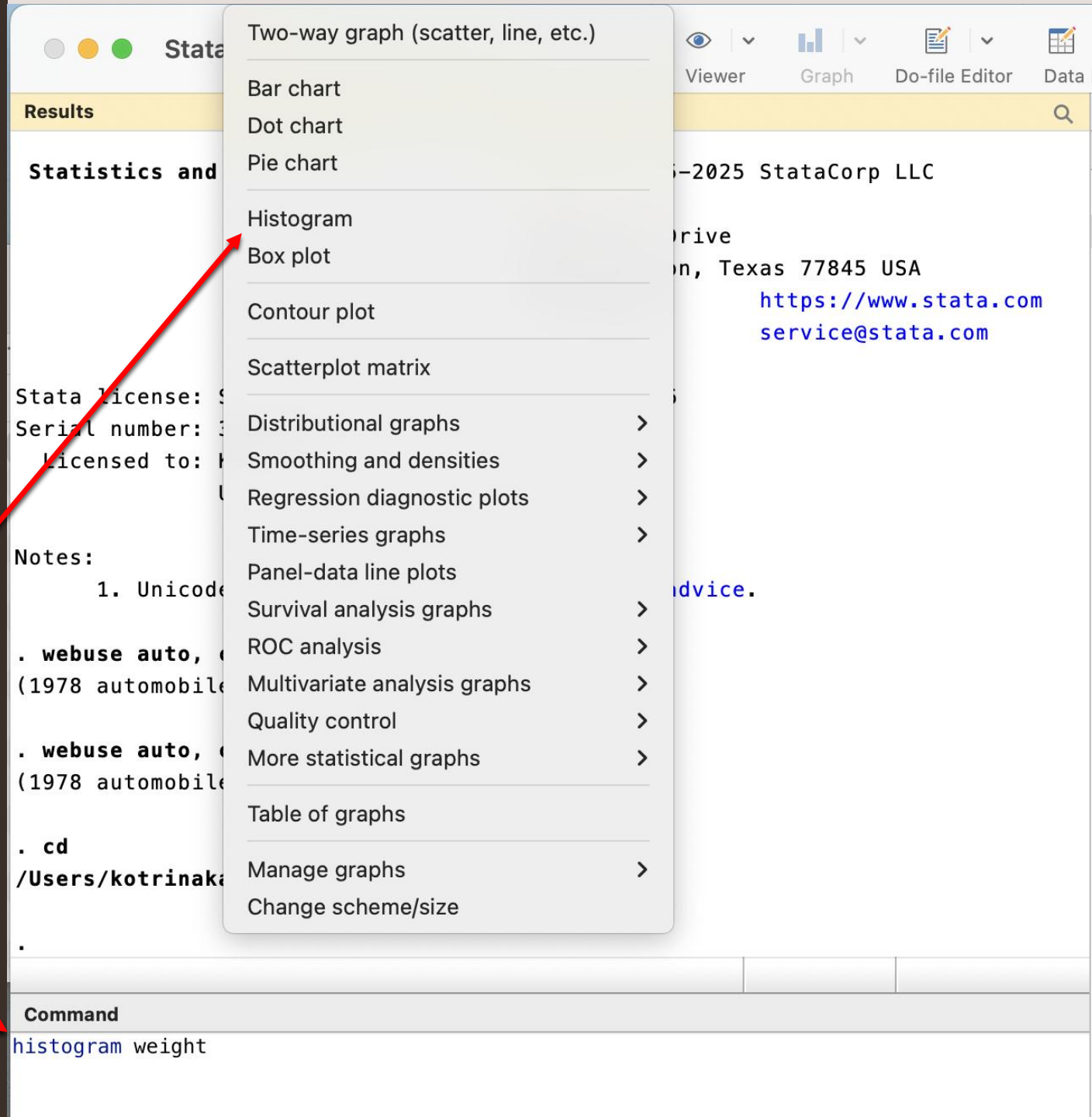


Stata menus

Typically 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

Stata Do-files

Text files where users write, save, and run sequences of commands

- Ensure **reproducibility**
- Facilitate **debugging** and **editing**

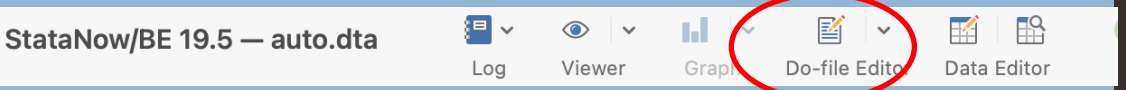
We recommend *always* using a do-file when using Stata

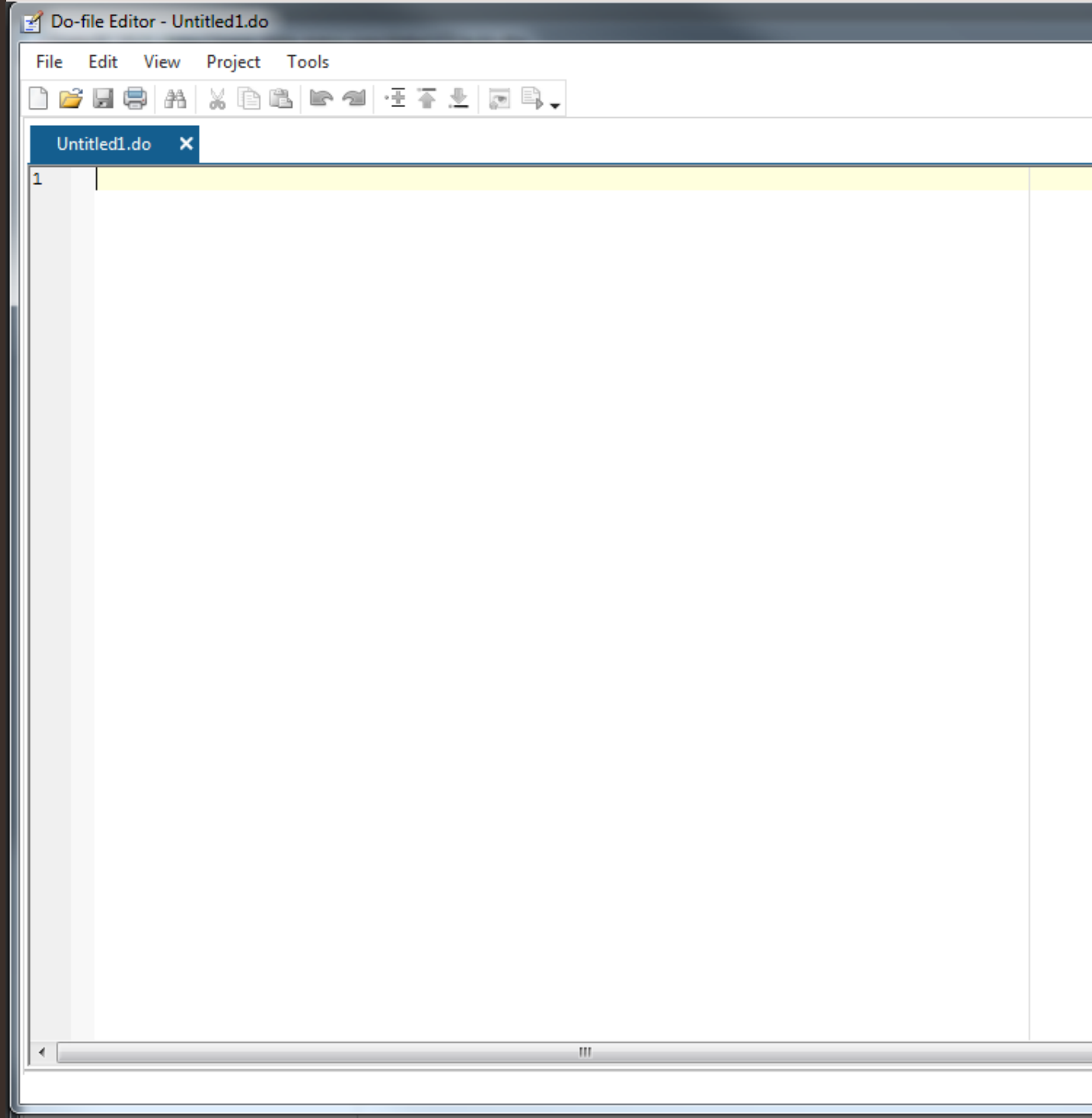
The file extension .do is used for do-files

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





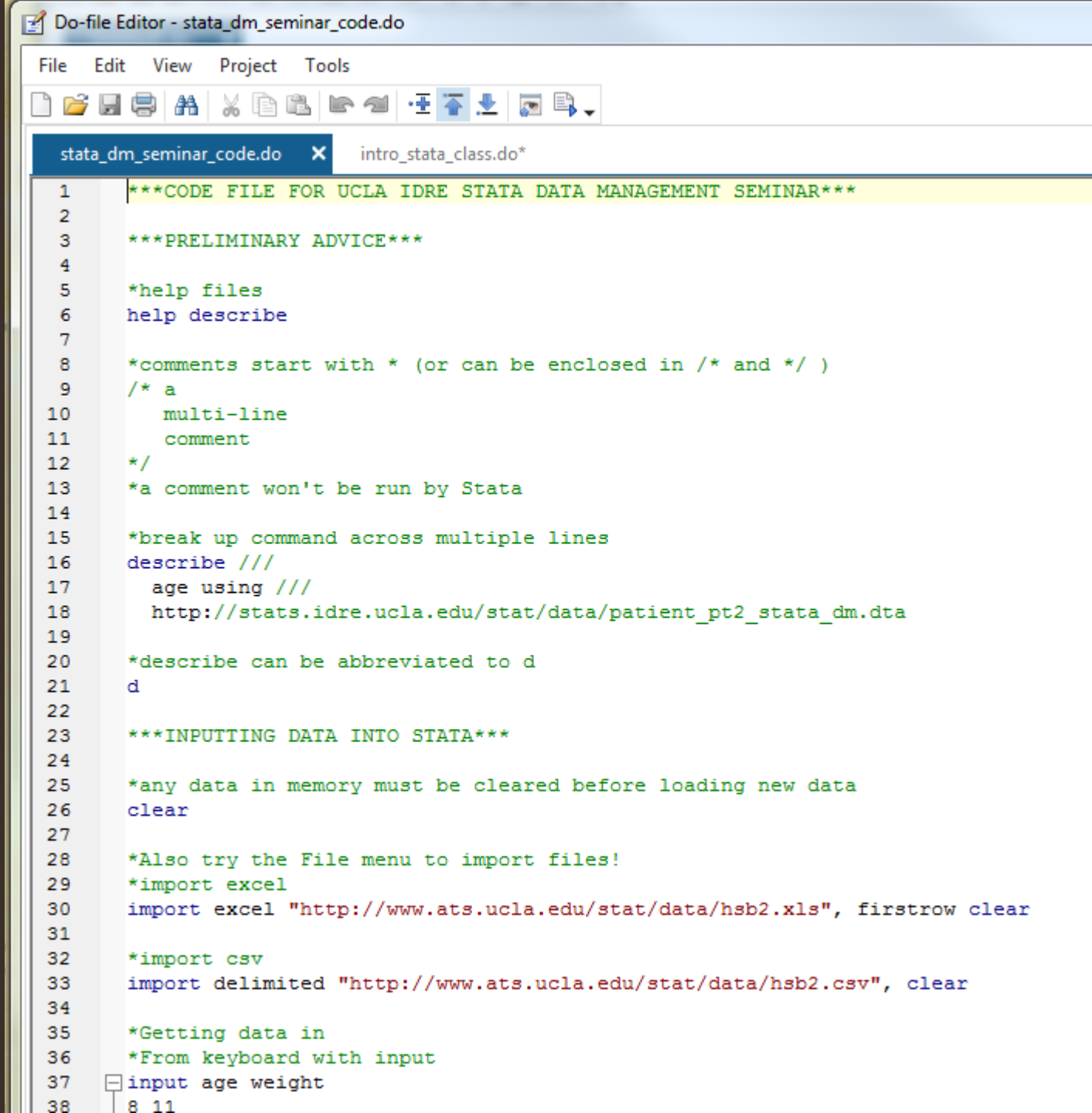
Syntax highlighting *

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"

As of Stata 16, tab can be used to auto-complete Stata commands and previously typed words

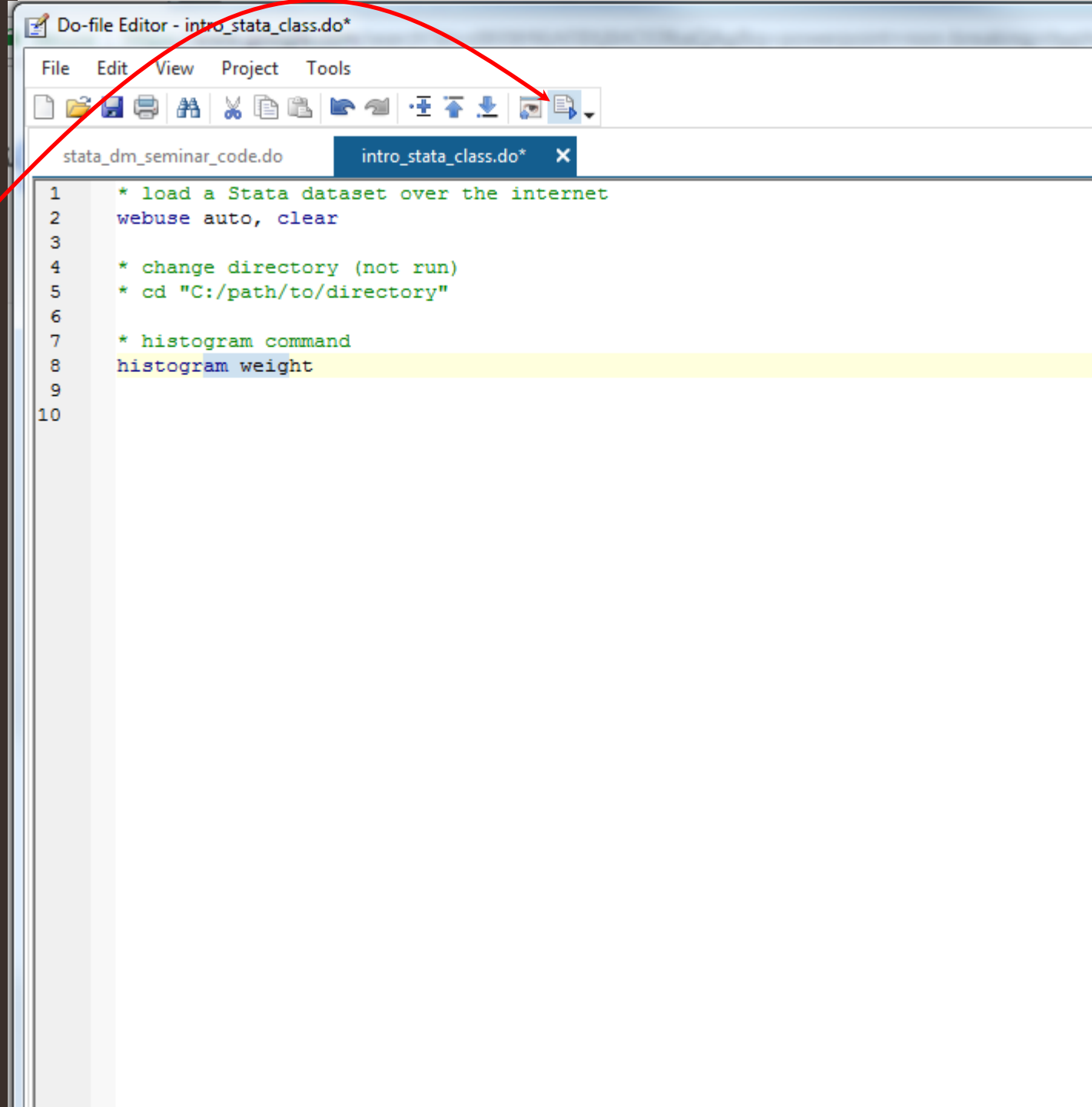


Running commands from the do-file

To run a command from the do-file:

1. Highlight part or all of the command,
2. Hit Ctrl-D (Mac: Shift+Cmd+D) or click the "Execute(do)" icon, the rightmost icon on the do-file editor toolbar

Multiple commands can be selected and executed

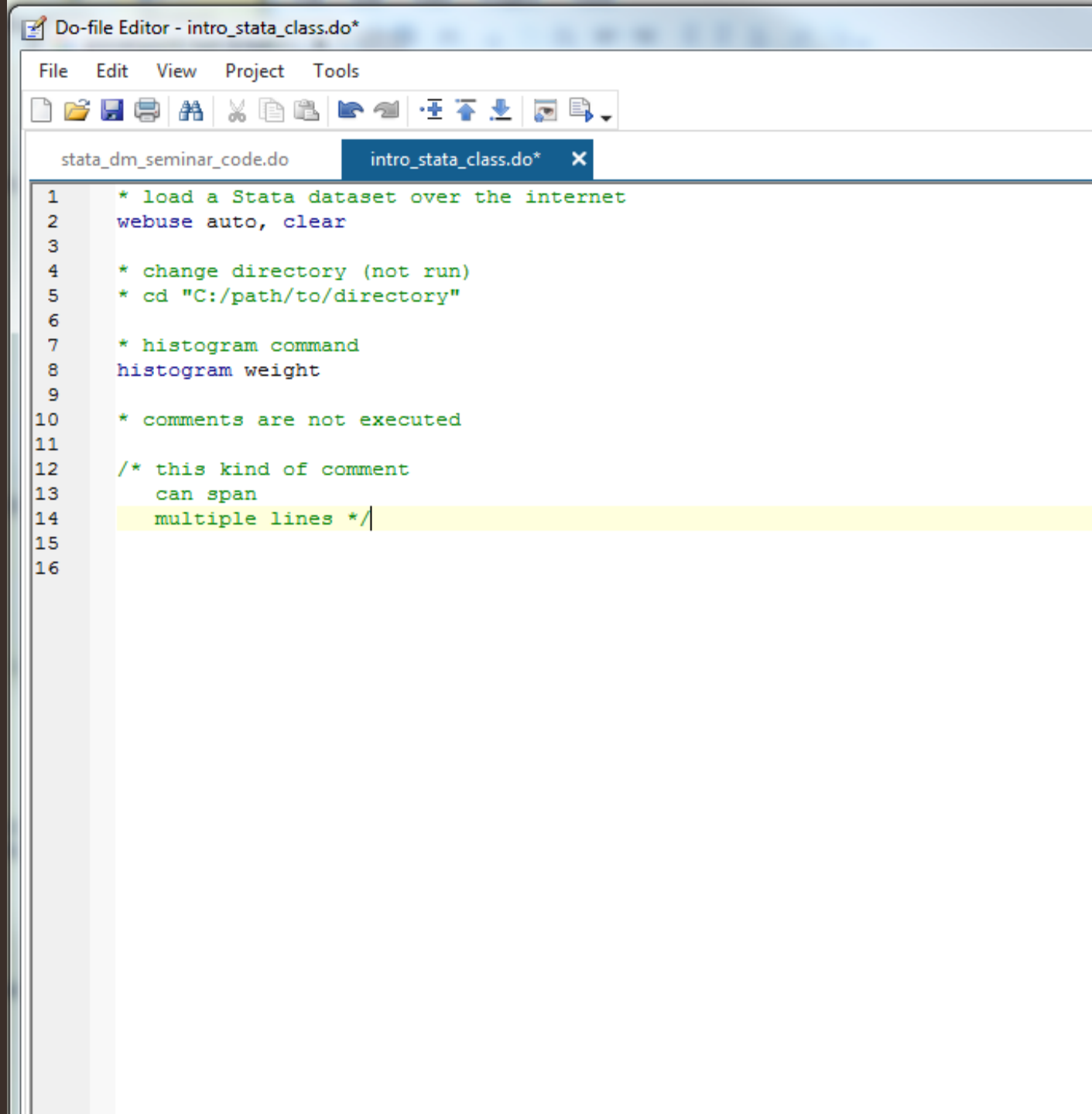


Comments

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



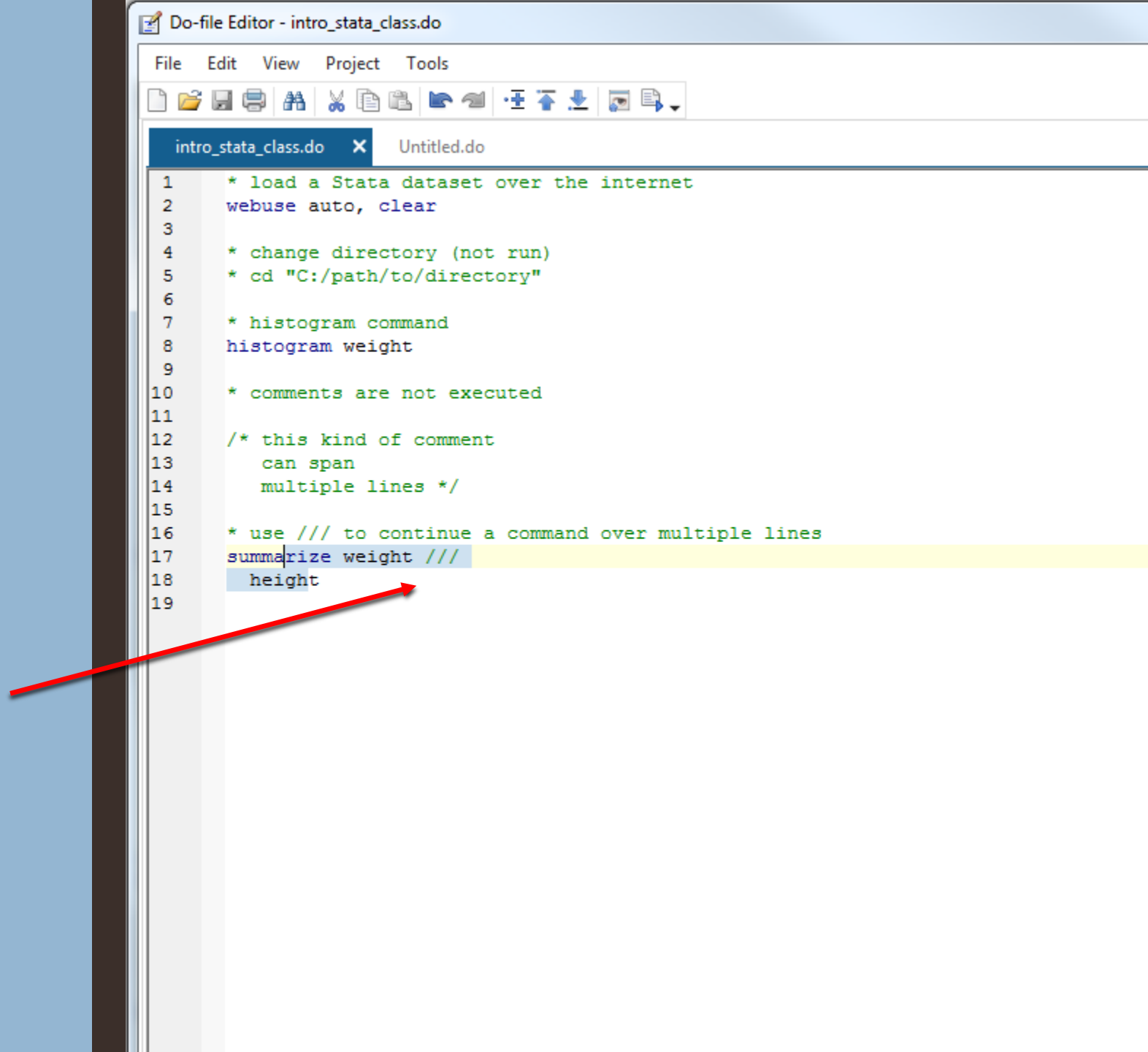
Long lines in do-files *

As of Stata 16 long commands automatically wrap

In older versions, to extend commands over multiple lines use /// 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 use the extension .dta

- Remember: code is written in "do-files," which usually have a .do extension

Double clicking on a .dta file in Windows will open the data in a *new* instance of Stata (not in the current instance)

- Be careful of having multiple Statas open

Loading and saving .dta files

`use` loads Stata .dta files

- .dta files can be loaded from a hard drive or over the internet (using a web address)

`save` stores data in Stata's .dta format

- The `replace` overwrites 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

By default, Stata will only hold one data set in memory at a time.

If the data are altered it in any way, we must `clear` these data from memory before loading in new data.

All data import commands have a `clear` option that 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
```

Frames *

As of Stata 16, multiple datasets can be loaded simultaneously with the `frame` suite of commands.

`frame create` *`newframename`* creates a location in memory called *`newframename`* to store a dataset.

`frame` *`newframename`*`:` *`command`* can then be used to perform *`command`* on the data stored in *`newframename`*.

`frame dir` lists the current data frames.

Commands not preceded by `frame` *`newframename`*`:` will be performed on the data in the default frame.

```
* create data frame called data2
frame create data2

* load nhanes2 data into data2 frame
frame data2: webuse nhanes2

* describe height and weight in nhanes2 data
frame data2: describe height weight

* look at data frames
frame dir
```

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("mysheet") firstrow clear
```

Importing .csv data sets *

Comma-separated values (.csv) 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)

Syntax and options are very similar to `import excel`

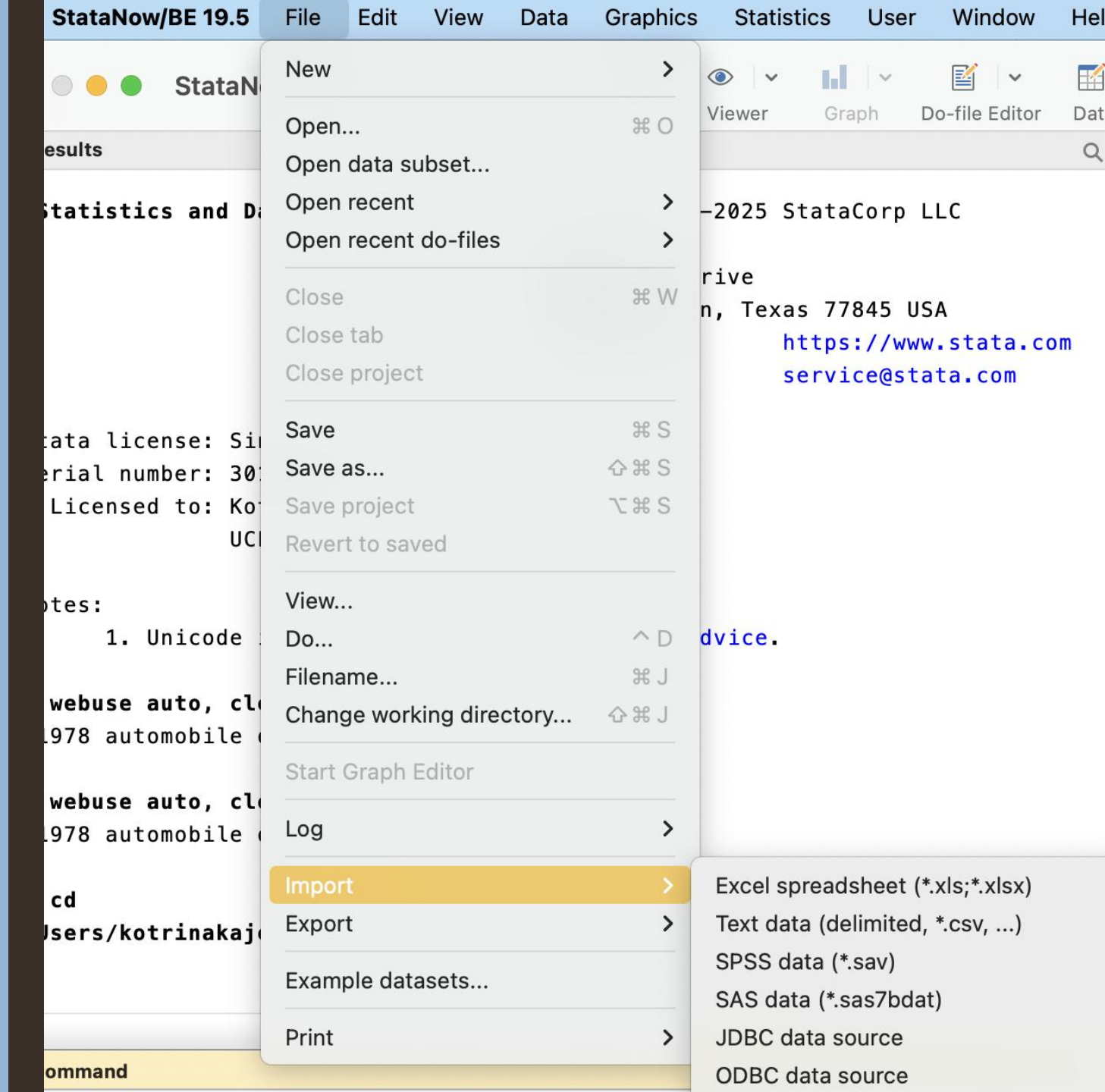
- But no need for `firstrow` option (first row is assumed to be variable names in .csv files)

```
* 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.

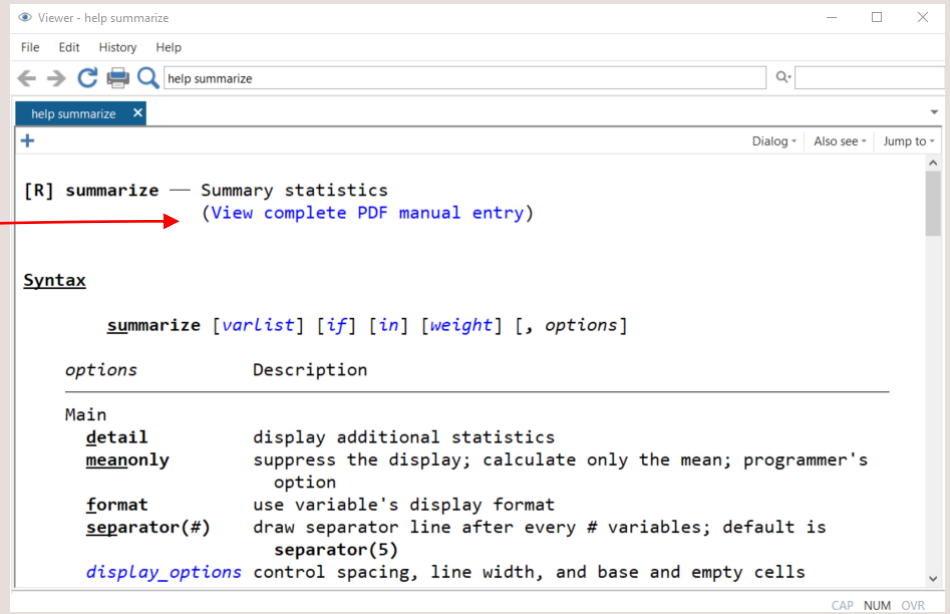
```
*open help file for command summarize

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

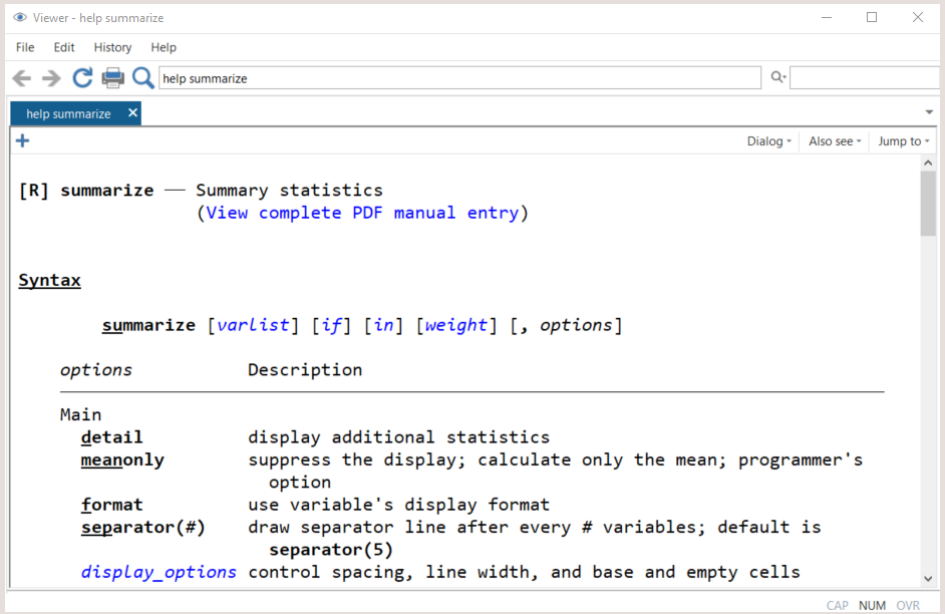
- We can use `su` for `summarize`

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



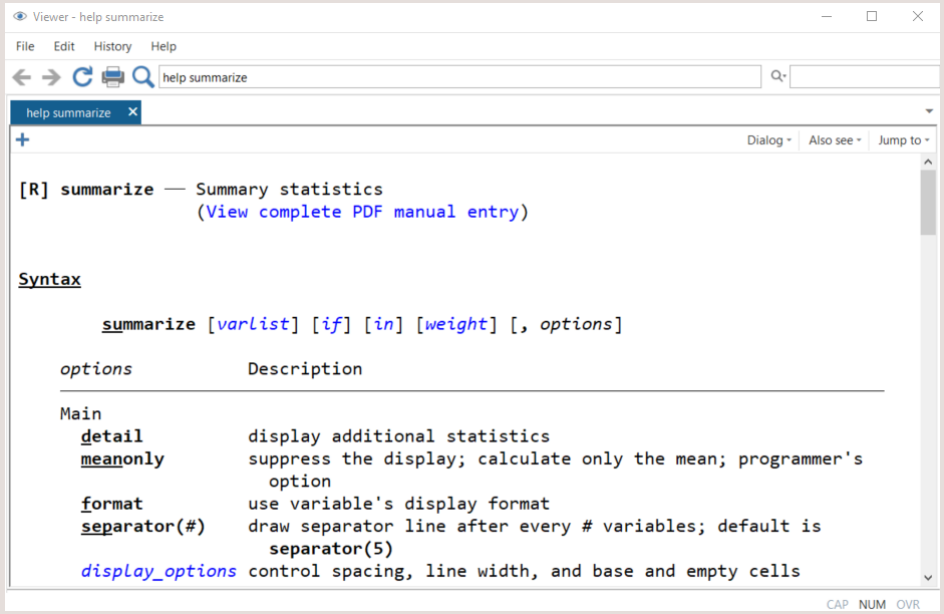
Help file: options section

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

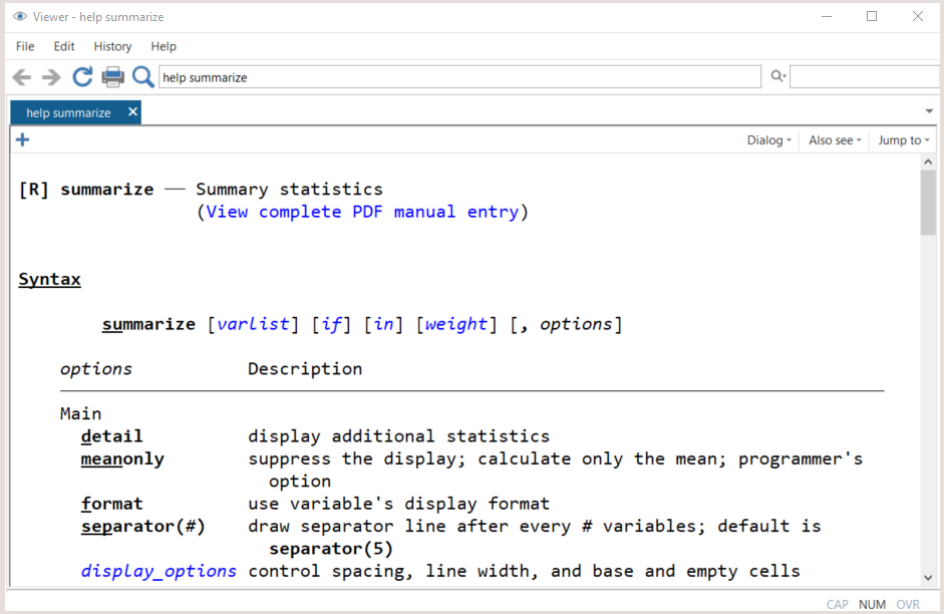
```
summarize
```

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

```
su read write
```

Provide additional statistics for variable read

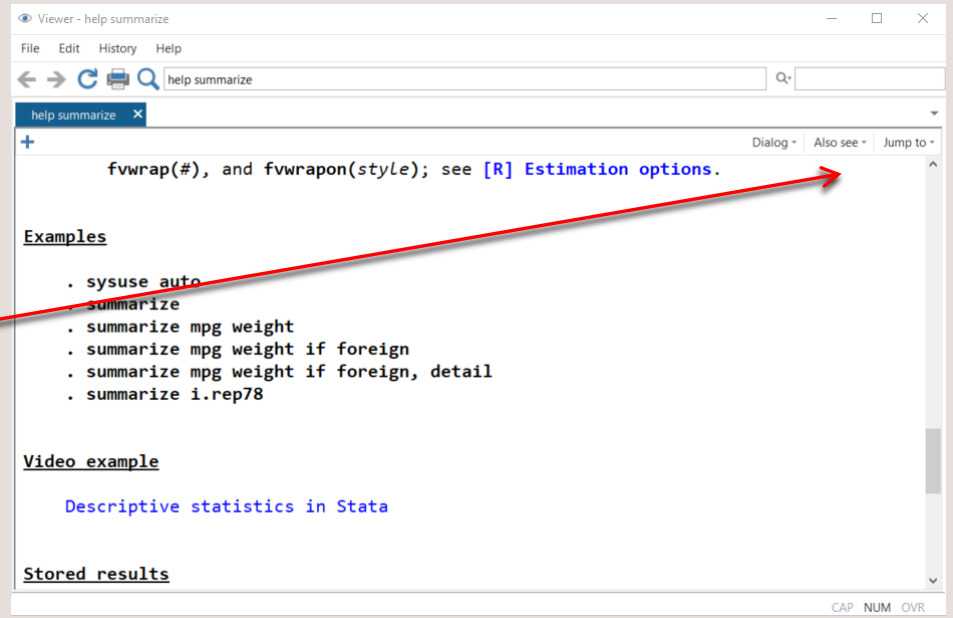
```
su read, detail
```



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
```

Seminar dataset

We will use a dataset consisting of 200 observations (rows) and 13 variables (columns)

Each observation is a student

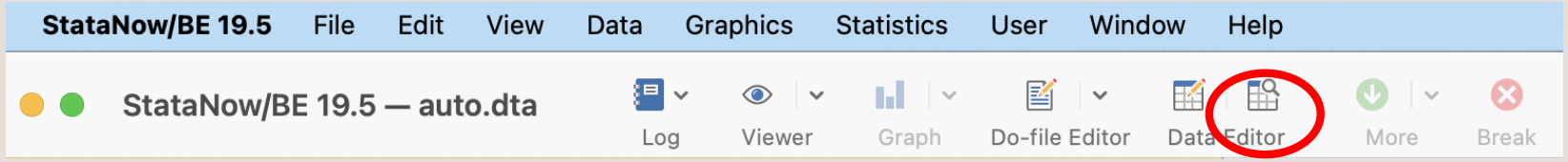
Variables

- 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!

```
* seminar dataset

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

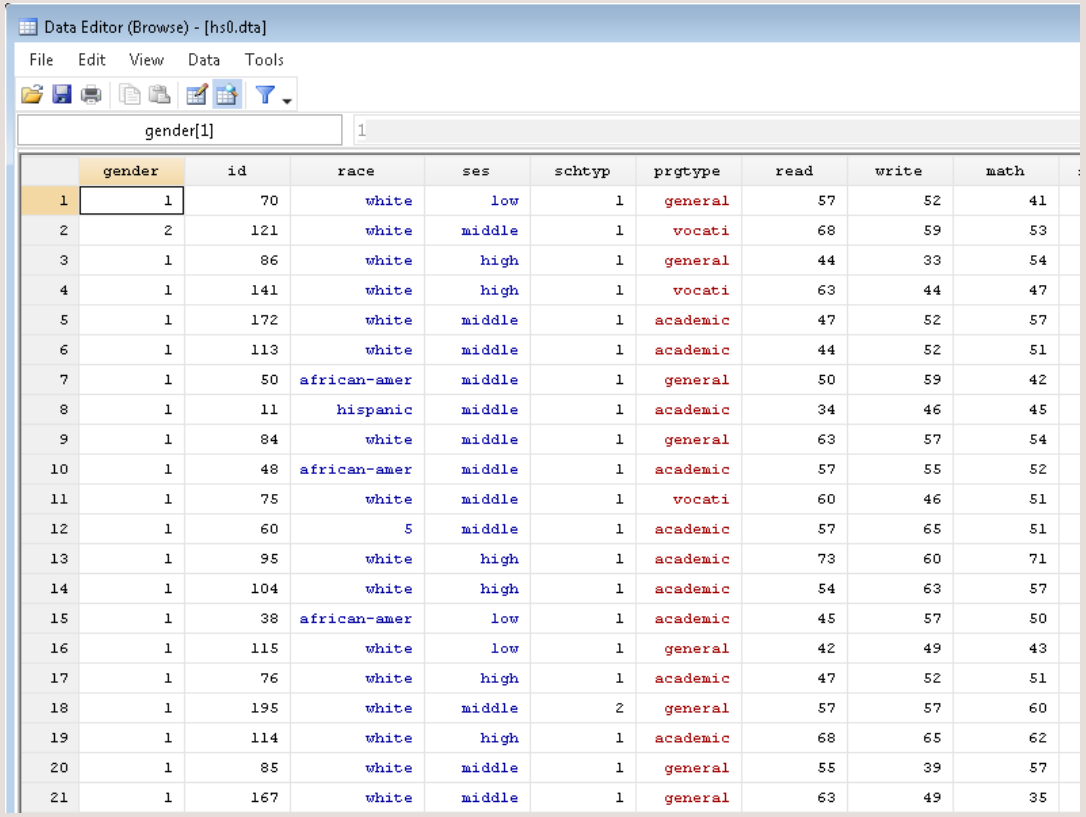
Browsing the dataset



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



	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

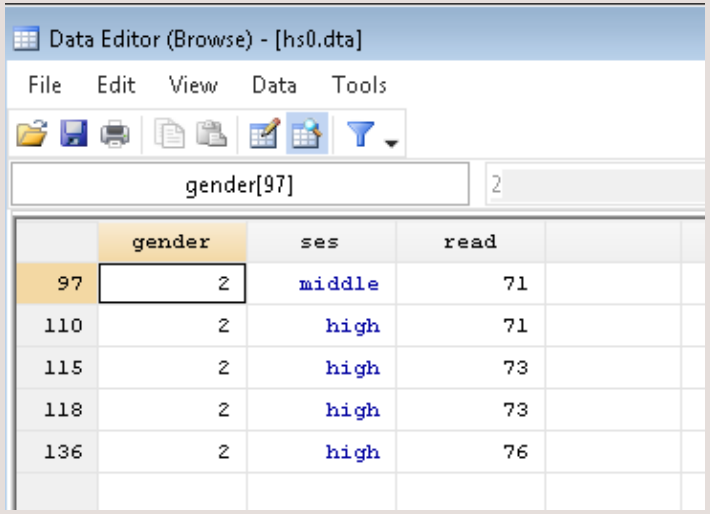
```
* list gender, ses, and math if math > 70
•    with clean output
li gender ses math if math > 70, clean
```

	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

Load the hs0 dataset using the code:

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

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 seminar 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

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
```

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
- 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

Visualizing data helps reveal patterns, trends, and relationships.

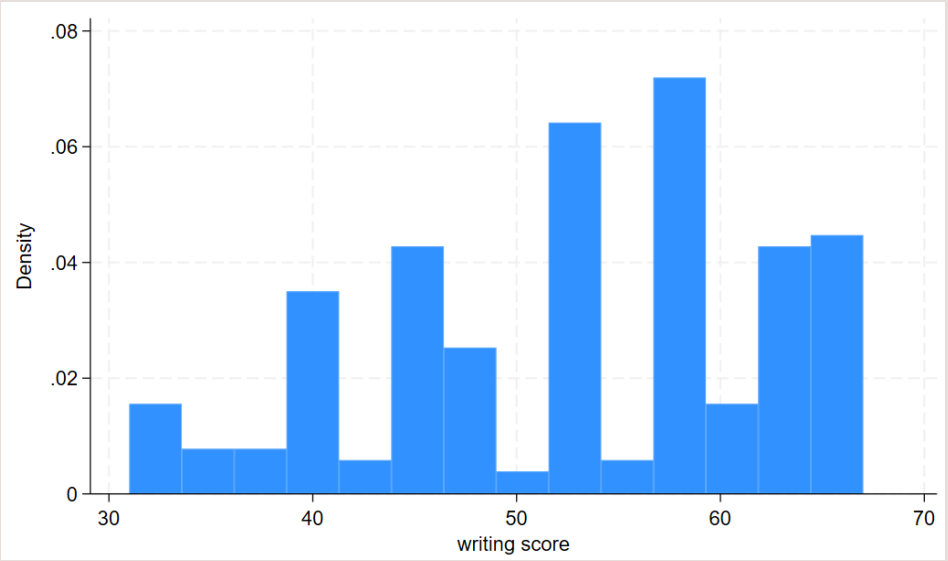
Graphs can be used to explore your data, to familiarize yourself with distributions and associations in your data

Graphs can also be used to present the results of statistical analysis

Histograms

Histograms plot distributions of variables by displaying counts (or density) of values that fall into various intervals of the variable

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

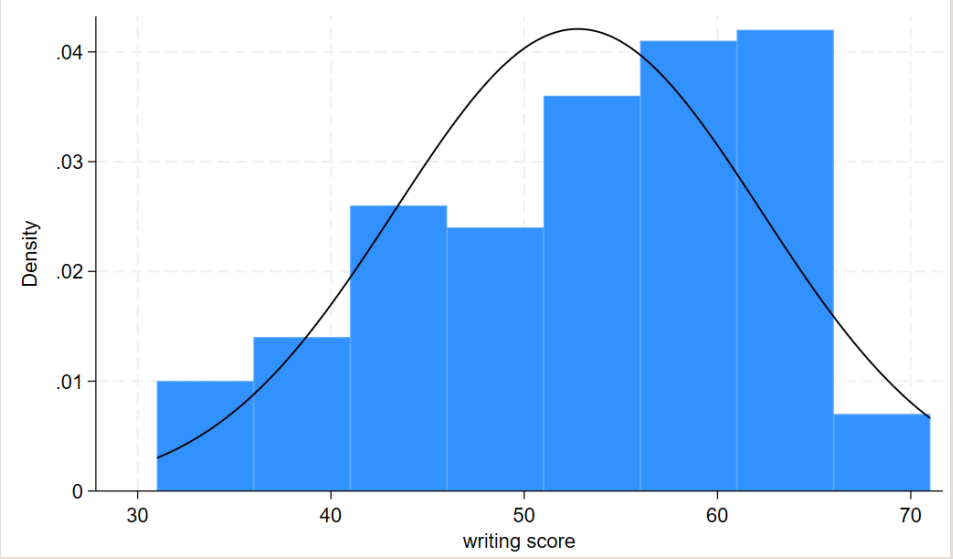


`histogram` options *

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)
```



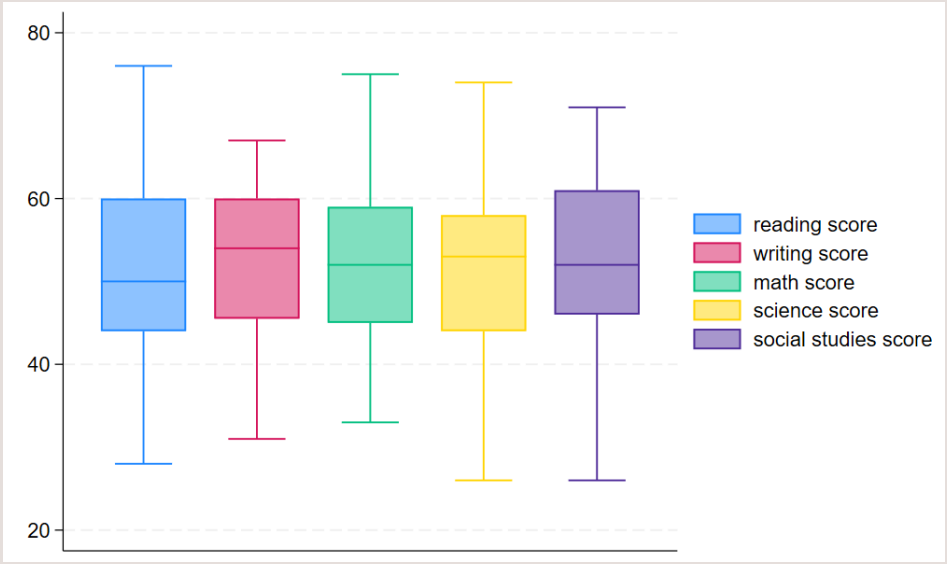
Boxplots *

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
```

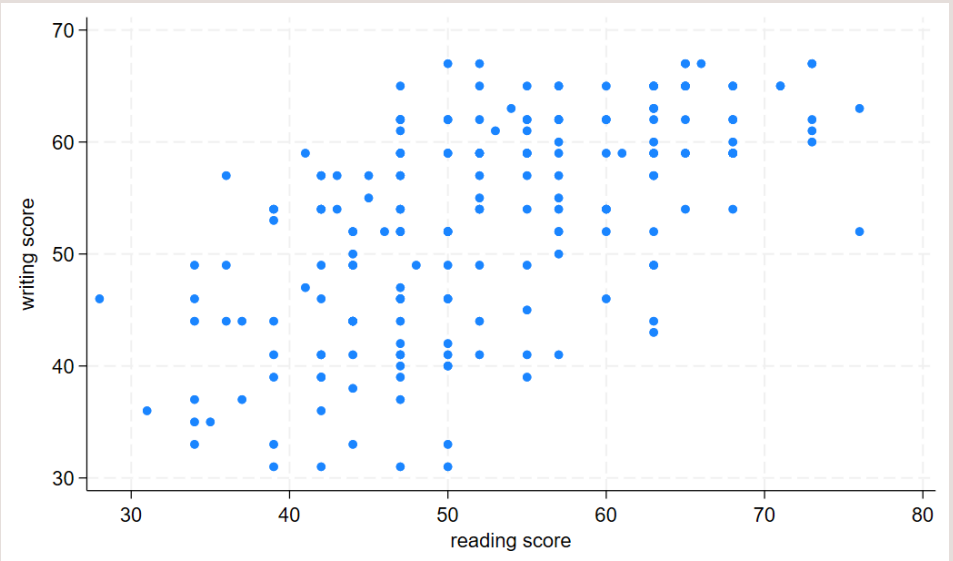


Scatter plots

Explore the relationship between two 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

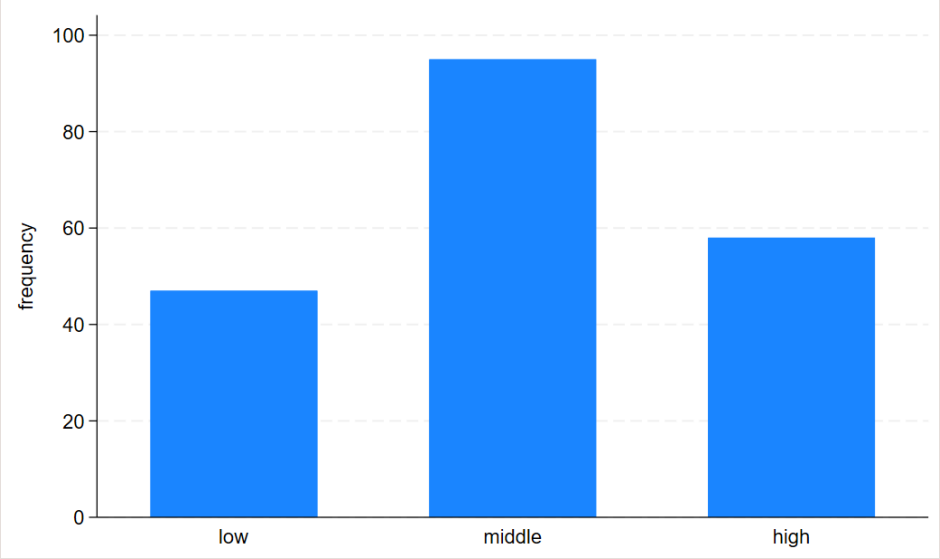
`graph bar` produces bar graphs in Stata

- 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)
```

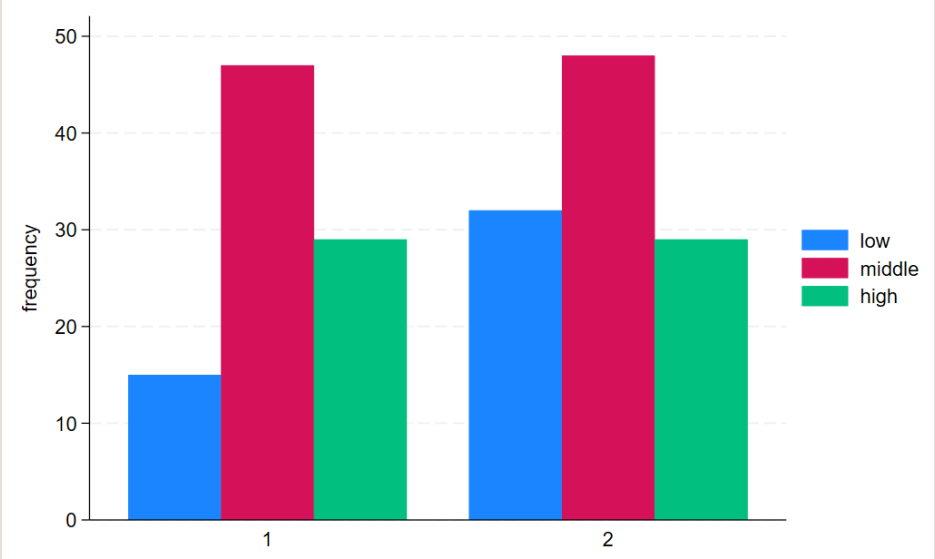


Two-way bar graphs *

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

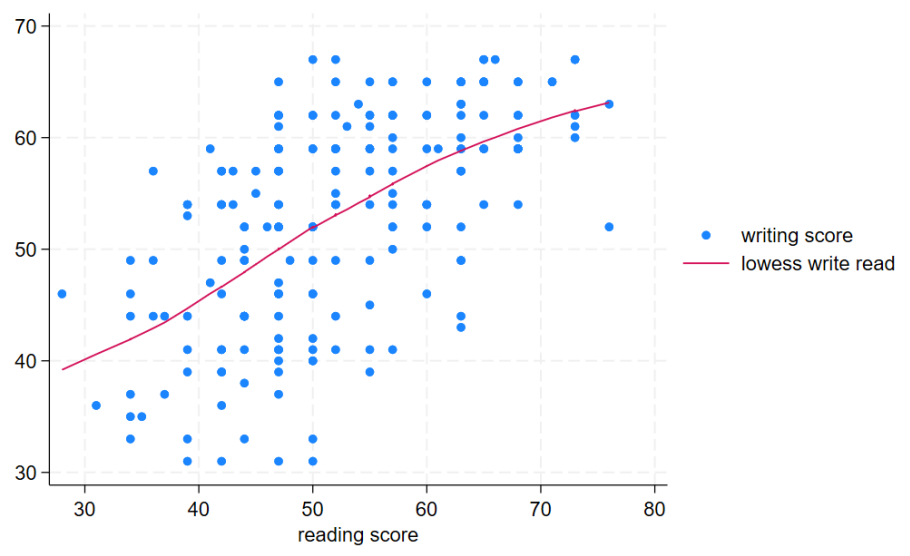
Each plot should involve a y-variable and an x-variable that appear on the y-axis and x-axis, respectively

- 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)
```

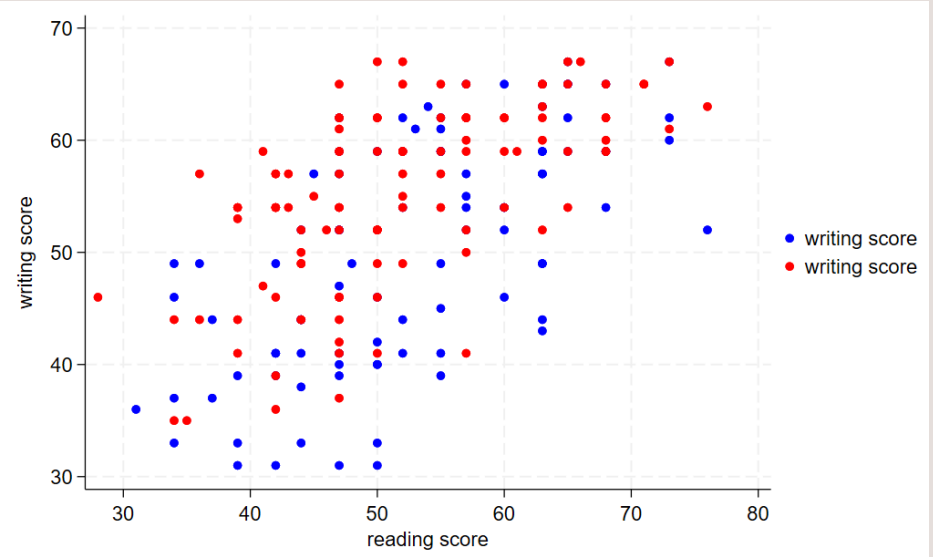


Layered graph example 2*

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))
```



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

- e.g. sums, logs, recodes

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 observations with missing on any variable used will generally 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

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	

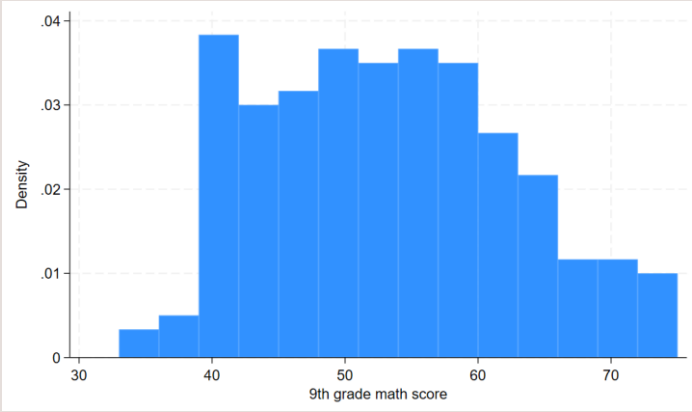
Labeling variables (1)

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 (2)

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
```

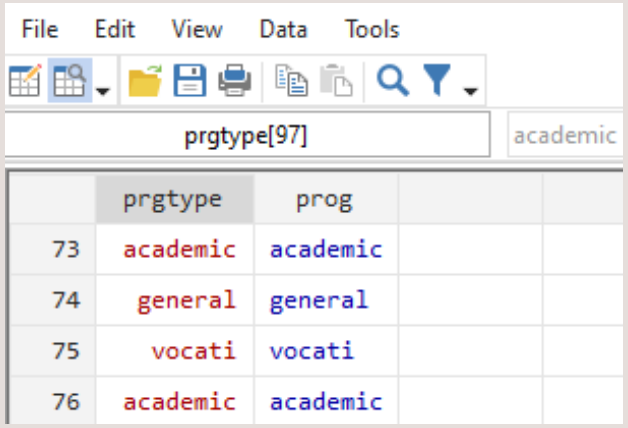
```
public/priv |
 ate school |      Freq.     Percent        Cum.
------------+-----------------------------------
     public |        168       84.00       84.00
    private |         32       16.00      100.00
------------+-----------------------------------
      Total |        200      100.00
```

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

`drop` removes the selected variables and keeps the rest

- Use `drop` if you want to remove a few variables but keep most of them

`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 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

Re-load 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-1merge

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
--------------------------------------------------------------
```

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
```

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
```

prog	ses: 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
------------------------------------------------------------------------------
```

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

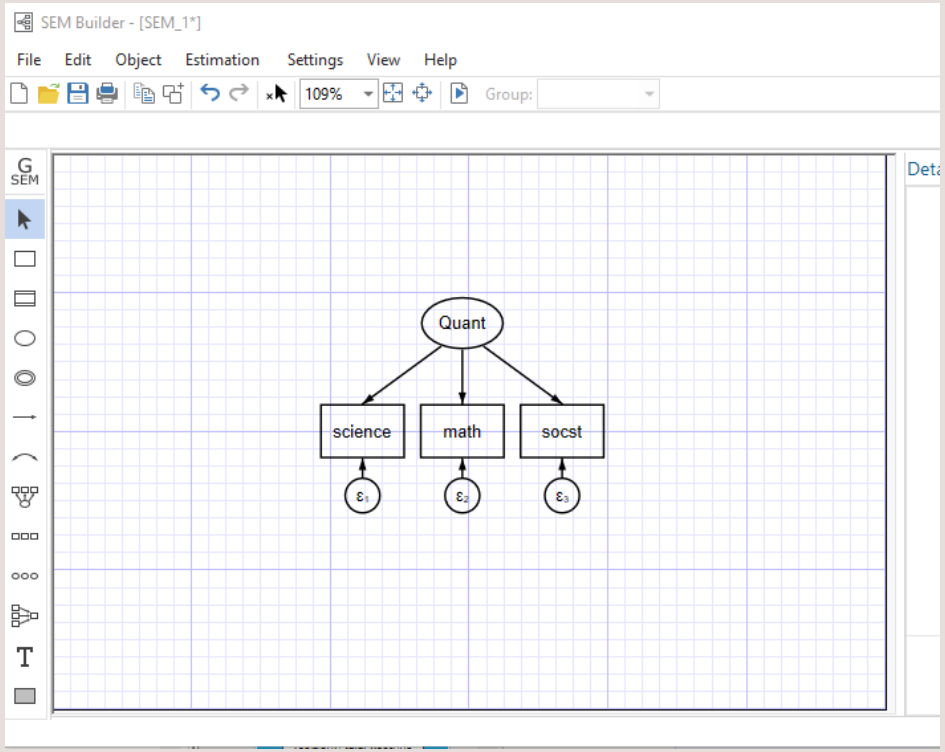
Stata features 2 ways to build a structural equation model (SEM)

- 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



OUTPUTTING TO WORD AND EXCEL

Outputting to Word

putdocx begin	Create a .docx file
putdocx paragraph	Initiate a new paragraph
putdocx textblock begin **putdocx textblock end**	Insert a block of text
putdocx table	Insert a table
putdocx image	Insert an image
putdocx save	Save .docx file

Opening the Word file and writing blocks of text

The `putdocx` suite of commands is used to output text, tables and images to Word files.

Most commands have many options to format and style the output

`putdocx begin` opens a .docx file for output

- The name of the file is specified when The file is saved

`putdocx textblock begin` and `putdocx textblock end` enclose text to be written to the Word file as is.

```
* create a Word docx file to export results, set default font size to 12
putdocx begin, font(12)

* first line starts a block of text in a new paragraph, using a Title style
* second line is the text to appear
* third line ends the text block
putdocx textblock begin, style(Title)
Regression of read results
putdocx textblock end

* start a block of text in a new paragraph
putdocx textblock begin
This report displays the results of the regression of read on write and math and a residuals vs fitted plot to check some of the assumptions of the linear regression model.
putdocx textblock end
```

Inserting a table of results and an image

`putdocx paragraph` initiates a new paragraph

`putdocx table` *`tablename`* `= etable` inserts a table of results from the last command run

`putdocx image` *`filename`* inserts an image

```
* run regression of read on write, math and prog
regress read write math

* put table of regression results into Word table
putdocx table model1 = etable

* make residuals vs fitted plot
rvfplot, yline(0)
* export plot to .png file
graph export rvf.png, replace
* start a new paragraph to add spacing
putdocx paragraph
* insert .png file
putdocx image rvf.png
```

Saving the Word output file

`putdocx save` *`filename`* closes and saves the Word file

- The `replace` option replaces the file if it exists
- If the file exists, make sure it is closed before using `putdocx save`

```
* save and close the .docx file, replace
file if it already exists
putdocx save regress_read.docx, replace
```

Outputting to Excel

putexcel set	Create a working spreadsheet for output
putexcel *cell* =	Insert something (e.g. text, table, image) into *cell* of working spreadsheet
putexcel save	Save .xlsx file

Open a working spreadsheet for output

The `putexcel` suite of commands is used to output text, matrices, tables and images to Excel files.

`putexcel set` *`filename`*`, sheet(`*`sheetname`*`)` opens sheet *`sheetname`* in file *`filename`* for output

```
* open Excel file for output, replace if it exists already
putexcel set "output.xlsx", sheet("regress results") replace
```

Adding content to spreadsheet cells

`putexcel` *`cell`* `=` "*`text`*" adds *`text`* to *`cell`* (e.g `A1`) in the open sheet

`putexcel` *`cell`* `= etable` adds the last set of estimation results to *`cell`*

`putexcel` *`cell`* `= image(`*`filename`*`)` inserts the image store in filename into *`cell`*

`cell` will hold the upper left part of a table of results or image

```
* regression of read on write and female
regress read write female
* write bolded text to cell A1 in Excel file
putexcel A1 = "Regression of read", bold
* write regression table to cell A2 in Excel file
putexcel A2 = etable

* make a boxplot of read by female
graph box read, over(female)
* export plot to .png file
graph export boxread.png, replace
* add plot to cell A7 of Excel file
putexcel A7 = image("boxread.png")
```

Close and save the Excel spreadsheet

`putexcel` *`save`* closes and saves the open spreadsheet

```
* close and save Excel file
putexcel save
```

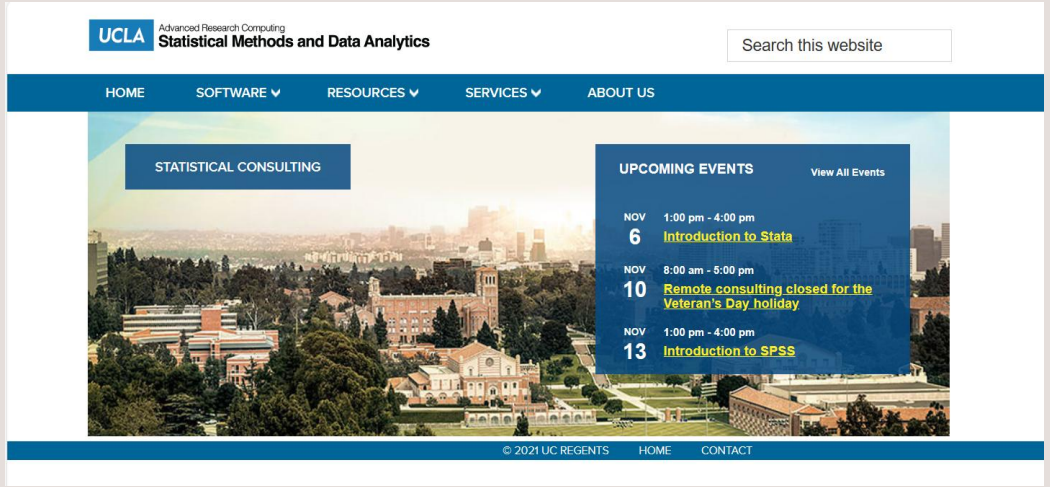
ADDITIONAL RESOURCES FOR LEARNING STATA

OARC Statistical Methods and Data Analytics website

The OARC Statistical Methods and Data Analytics 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



OARC statistical consulting website Stata pages

On the website landing page for Stata, you'll find many links to our Stata resources pages

- Stata (ucla.edu)

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)

OARC YouTube channel (Stata and other workshops)

END
THANK YOU!