Unit 4

Introduction to Stata *version 16*

Dear 2020 Class– These notes are fine for earlier versions

"Data! Data! Data! I can't make bricks without clay."

- Sherlock Holmes

Stata is one of many statistical software programs available for data management, statistical analysis and the production of publication grade graphics. It is often used in public health research and is the statistical software program used in this course.

It has several good features: (1) it can be used on either a PC or a MAC (2) it is easy to learn (3) many analyses can be done using menus and dialog boxes and later, when you are ready (4) it permits high level command driven analyses that are quite sophisticated and (5) it permits batch file processing; you can write programs with command syntax that can be submitted as a "batch" job.

Stata can be used three ways: (1) interactively with menus and dialog boxes; (2) interactively using command syntax; and (3) batch using pre-written programs of code called do-files.

Table of Contents

- continued-

Table of Contents - continued

Learning Objectives

When you have finished this unit, you should be able to:

- Launch Stata, exit Stata, and obtain help**;**
- Use Stata as a calculator;
- Load a Stata data set into working memory;
- Input and output excel data;
- Use the data editor in Stata to create a new data set;
- Creating variable label and variable value labels;
- Create new variables;
- Assign missing value codes;
- Select subsets of observations for analysis;
- Manage multiple data sets using the append and merge commands;
- Screen a data set for a variety of "errors"; and
- Create a **do-file** of Stata commands.

1. Sample Session

Tip! If you are new to Stata, play with the menus and dialog boxes being sure to record a log of your session (more on logs in 3.1 "Maintain a Log of Your Session." page 12) The log you produce will then be a record of the command syntax that Stata used. How handy!

Key
Green – comments (Note - comments begin with an asterisk *)
Black – commands
Blue – output

```
. *-------------------------------------------------------------------------
. *  BIOSTATS 690C - Fall 2020
. *
. *    prog:     Carol Bigelow
. *    date:     September 16, 2020
. *   input:     …/your path and folder/ivf.dta
. *  output:     none
. *   title:     Sample Session for Unit 4 - Introduction to Stata
. *-------------------------------------------------------------
. *
. *---------------------  Preliminaries ---------------------
. *  Turn off the 'one screen at a time' display of results
. set more off

. * Read sample data from site on the internet (or download data from course website and use FILE > OPEN)
. use "http://www.pauldickman.com/survival/ivf.dta"
(In Vitro Fertilization data)



. *
. *--------------- Obtain summary information on variables, types, sample size, etc ---*
. codebook, compact

Variable   Obs Unique      Mean    Min    Max  Label
-------------------------------------------------------------------------------------------
id         641    641       321      1    641  identity number
matage     641     21  33.97192     23     43  maternal age (years)
hyp        639      2  .1392801      0      1  hypertension (1=yes, 0=no)
gestwks    641    177  38.68725  24.69  42.35  gestational age (weeks)
sex        641      2   1.49142      1      2  sex of the baby
bweight    641    295  3129.137    630   4650  birthweight (g)
-------------------------------------------------------------------------------------------
```

```
. *------------ Obtain detailed information on a continuous variable:  matage -------*
. codebook matage

-------------------------------------------------------------------------------------------------
matage                                                                    maternal age (years)
-------------------------------------------------------------------------------------------------

                  type:  numeric (byte)

                 range:  [23,43]                      units:  1
         unique values:  21                       missing .:  0/641

                  mean:   33.9719
             std. dev:   3.87046

           percentiles:        10%       25%       50%       75%       90%
                                29        31        34        37        39

. *-------------- Obtain detailed information on a discrete variable:  hyp ------*
. codebook hyp

-------------------------------------------------------------------------------------------------
hyp                                                          hypertension (1=yes, 0=no)
-------------------------------------------------------------------------------------------------

                  type:  numeric (byte)

                 range:  [0,1]                        units:  1
         unique values:  2                        missing .:  2/641

            tabulation:  Freq.  Value
                           550  0
                            89  1
                             2  .

. * ------------  List first 5 observations, all variables -------*
. list in 1/5

     +-------------------------------------------------+
     | id   matage   hyp   gestwks      sex   bweight |
     |-------------------------------------------------|
  1. |  1       33     0     37.74   female      2410 |
  2. |  2       34     0     39.15   female      2977 |
  3. |  3       34     0     35.72   female      2100 |
  4. |  4       30     0     39.29     male      3270 |
  5. |  5       35     0     38.38   female      2620 |
     +-------------------------------------------------+
```

```
. *------------ List first 20 observations of two variables matage and bweight -----*
. list matage bweight in 1/20

     +------------------+
     | matage   bweight |
     |------------------|
  1. |     33      2410 |
  2. |     34      2977 |
  3. |     34      2100 |
  4. |     30      3270 |
  5. |     35      2620 |
     |------------------|
  6. |     37      3260 |
  7. |     31      3750 |
  8. |     31      1450 |
  9. |     33      3200 |
 10. |     33      3675 |
     |------------------|
 11. |     29      3640 |
 12. |     37      3771 |
 13. |     36      3950 |
 14. |     39      3400 |
 15. |     34      3100 |
     |------------------|
 16. |     36      3100 |
 17. |     37      4020 |
 18. |     35      2730 |
 19. |     38      3000 |
 20. |     34      3040 |
     +------------------+


. *--------  Descriptives for a categorial variable ------*
. tabulate hyp

hypertensio |
  n (1=yes, |
      0=no) |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        550       86.07       86.07
          1 |         89       13.93      100.00
------------+-----------------------------------
      Total |        639      100.00


. *-----------  Descriptives for a categorical variable with display of missing ---*
. tabulate hyp, missing

hypertensio |
  n (1=yes, |
      0=no) |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |        550       85.80       85.80
          1 |         89       13.88       99.69
          . |          2        0.31      100.00
------------+-----------------------------------
      Total |        641      100.00
```

```
. *------- Crosstabulation of two categorical variables, with display of missing -----*
. tab2 sex hyp, missing

-> tabulation of sex by hyp

sex of the |    hypertension (1=yes, 0=no)
      baby |         0          1          . |     Total
-----------+---------------------------------+----------
      male |       273         52          1 |       326
    female |       277         37          1 |       315
-----------+---------------------------------+----------
     Total |       550         89          2 |       641


. *----------  Cross tab with row percentages and display of missing ----*
. tab2 sex hyp, missing row

-> tabulation of sex by hyp

+----------------+
| Key            |
|----------------|
|   frequency    |
| row percentage |
+----------------+

sex of the |    hypertension (1=yes, 0=no)
      baby |         0          1          . |     Total
-----------+---------------------------------+----------
      male |       273         52          1 |       326
           |     83.74      15.95       0.31 |    100.00
-----------+---------------------------------+----------
    female |       277         37          1 |       315
           |     87.94      11.75       0.32 |    100.00
-----------+---------------------------------+----------
     Total |       550         89          2 |       641
           |     85.80      13.88       0.31 |    100.00




. *-----------  Detailed descriptives - continuous variable -----*
. summarize matage, detail

                    maternal age (years)
-------------------------------------------------------------
      Percentiles      Smallest
 1%           25             23
 5%           27             23
10%           29             24       Obs                 641
25%           31             24       Sum of Wgt.         641

50%           34                      Mean           33.97192
                        Largest       Std. Dev.       3.87046
75%           37             42
90%           39             43       Variance       14.98046
95%           40             43       Skewness      -.2659265
99%           41             43       Kurtosis       2.523825
```

```
. *------------  Selected descriptives - continuous variable ---*
. tabstat matage, stat(mean sd min max)


    variable |      mean        sd       min       max
-------------+--------------------------------------------
      matage |  33.97192   3.87046        23        43
----------------------------------------------------------

. *-----------  Selected descriptives - continuous variable, grouped (stratified) --*
. sort sex
. tabstat matage, by(sex) stat(mean sd min max)

Summary for variables: matage
     by categories of: sex (sex of the baby)

   sex |      mean        sd       min       max
-------+--------------------------------------------
  male |  34.21166  3.790026        25        43
female |  33.72381  3.942651        23        43
-------+--------------------------------------------
 Total |  33.97192   3.87046        23        43
----------------------------------------------------

. *----------------  Using Stata to count --------*
. count if bweight <= 2000 & sex==1
  18

. *-----------  Using Stata as a calculator ---------*
. display 2+2
4
```

2. Launching, Exiting, and Getting Help

Launching

There are multiple ways to launch Stata

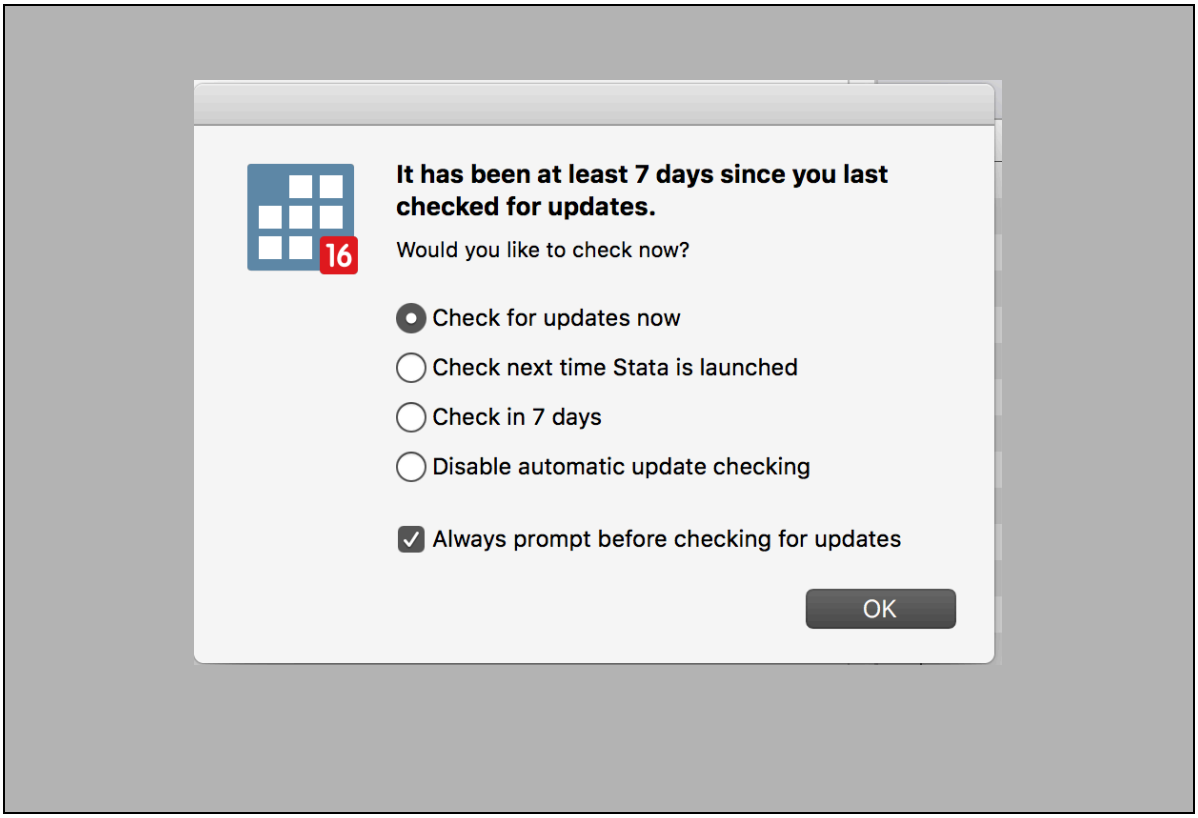
Windows Users

___ **START > ALL PROGRAMS > Stata;** or
___ **Double click** on the Stata icon on your desktop; or
___ **Double click** on a previously created Stata do-file.; or
___ **Double click** on a previously created Stata data set. These have extension ".dta"

Mac Users

___ **APPLICATIONS > STATA folder > Stata**; or
___ **Double click** on the Stata icon on your dock; or
___ **Double click** on a previously created Stata do-file.; or
___ **Double click** on a previously created Stata data set. These have extension ".dta"

You might see something like the following:



Stata will ask you if you would like to check for updates. Click OK.

Tip! Checking for updates is a good idea. And it doesn't take long. If this check is not done for you automatically, you can do it manually as follows from the main toolbar at top:

HELP > OFFICIAL UPDATES

Exiting

Tip! Before exiting Stata, be sure you know *exactly* what you are doing vis a vis saving data.

___ (1) From the toolbar: Click on the **SAVE** icon; or
___ (2) From the command window: Type in your syntax choice; eg -

. save ***filename***
. save, replace ← The option replace (after the comma) will overwrite the current data set
. save, ***newfilename***

There are three ways to exit Stata

___ (1) From the main menu: **Stata 16 > Quit Stata;** *or*
___ (2) **Click** the {X} button at the upper right corner of the screen: *or*
___ (3) Using the command window: Type **exit**

Getting Help **See also page 19.**

Tip - Use the help features! Stata has lots of help that can be accessed from the command line.

___ (1) From the menu bar at top: Click on HELP

___ (2) ***I use this ALL the time!!*** From the command window: Type help followed by the command name; eg - **help tabstat**

___ (3) Stata Corp. Resources for Learning Stata are at **https://www.stata.com/links/resources-for-learning-stata/**

___ (4) UCLA resources for learning Stata are at **https://stats.idre.ucla.edu/stata/**

___ ***BEWARE!!*** - Stata is ***case sensitive*** for all variable names and for most commands (stick to lower case)

3. Some Basics

3.1 Maintain a Log of Your Session

How to Maintain a Log of Your Session

Action	Command in Stata
To begin log	**FILE > LOG > BEGIN** At the format type box: **STATA LOG**
To close, suspend or resume	**FILE > LOG** Make your selection
To close	There are 2 ways to do this: (1) **FILE > LOG > CLOSE** (2) Exit Stata
To convert from one format to another	**FILE > LOG > TRANSLATE**

Note – The maintenance of a log can also be done from the command window. See Section 3.3

Tip! Always create a log of your session.

A log file is a record of everything you do in Stata (data manipulations, command syntax, output plus error messages). ***Beware*** – Stata log files do not record pop up graphs or help windows. **Graphs have to be saved separately and explicitly.**

A Stata log can be saved in either of two formats: **"smcl" or "log"**

.smcl – smcl stands for ***"Stata markup and control language"***. This format preserves all the Stata formatting and controls.

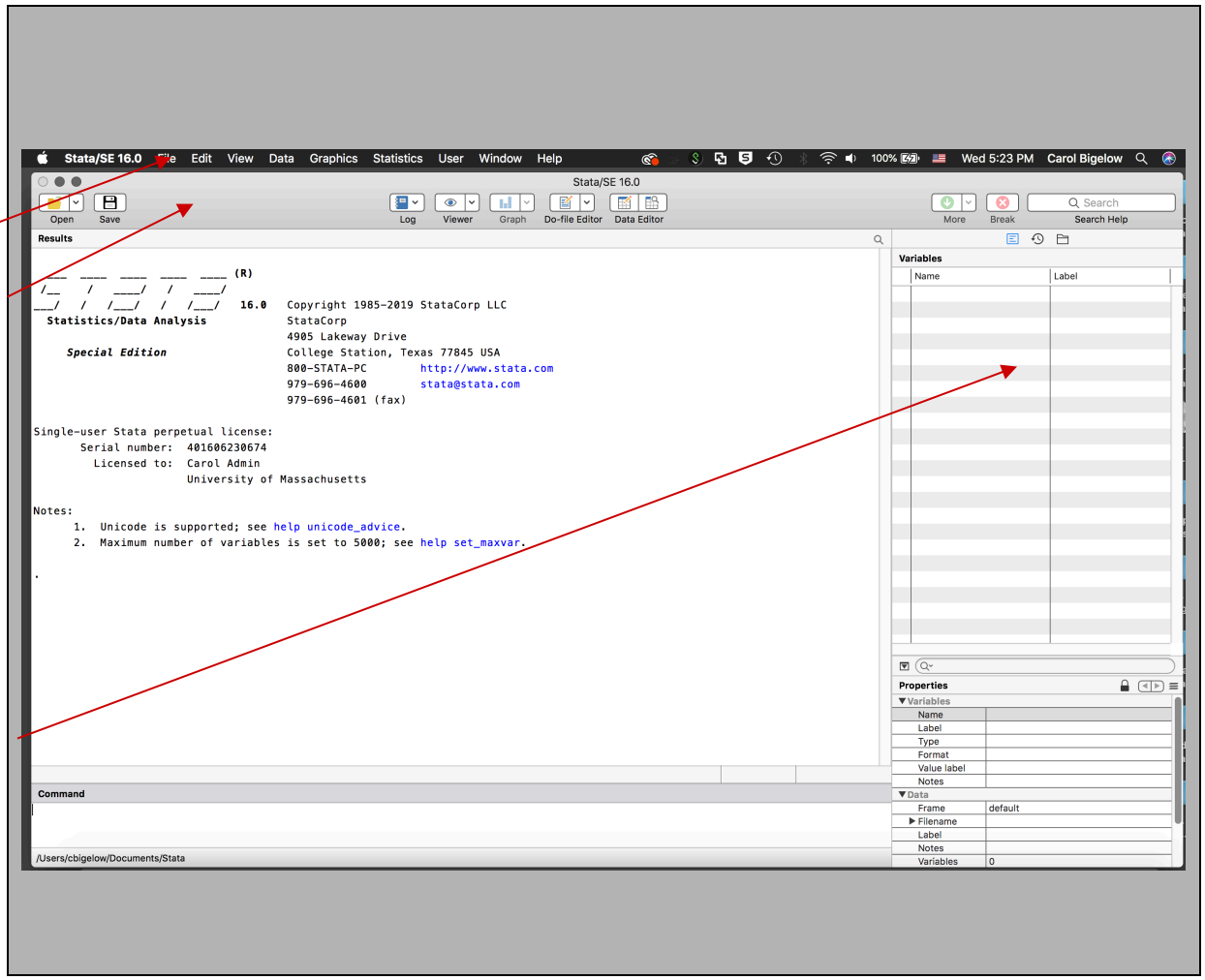
.log – This is a plain text format. This format is easily imported into MS Word or Notepad.

Note - It is possible to "translate" the format of your log from one format to other formats that are readable by other applications. See table below.

3.2 Introduction to the Windows in Stata

When Stata starts up, your screen will display the following (unless you have changed your windows preferences):

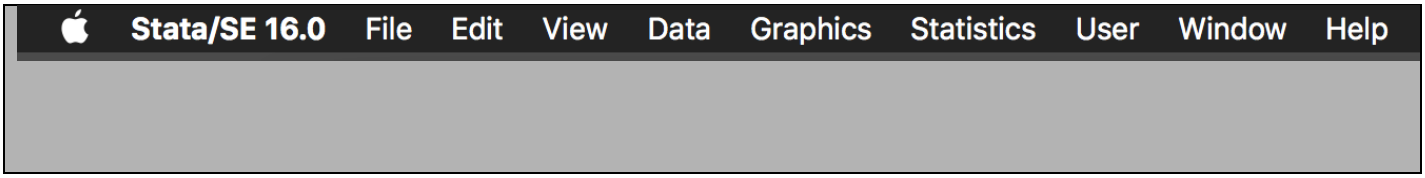
- **main menu**: This is the narrow toolbar located at the top of your screen,
- **tool bar**: This is located just below main menu, plus
- **windows**: In this picture, you see the variables window.



Design ..... Data Collection ..... Data Management ..... Data Summarization ..... Statistical Analysis ..... Reporting

Main Menu

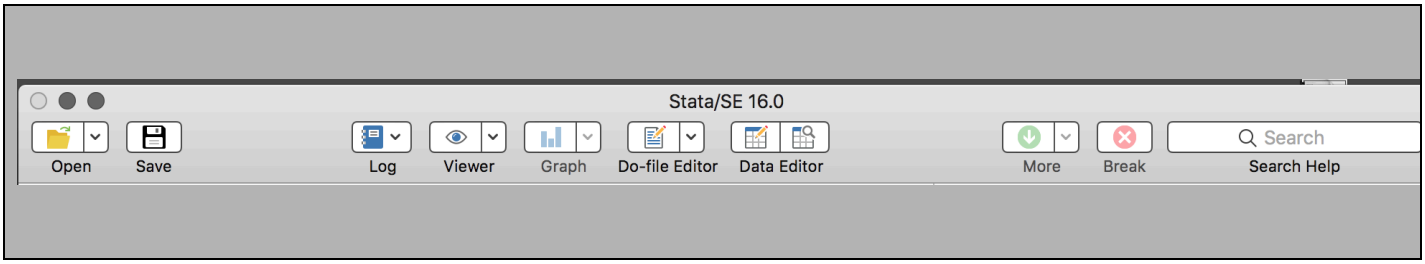
Tip - To obtain a description of each icon, simply place your mouse over the icon. A description should appear. The main menu is similar in layout to the main menu in many software programs. Each selection ("File", "Edit", etc) produces a drop down menu from which you can do additional, related, selections



Key to Main Menu

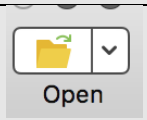
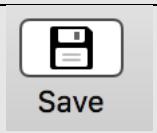
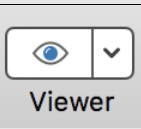
	Drop Down Menus (partial listing)
File	• **Open**: open/use a Stata data set • **Save/Save as**: save the current Stata data set • **Do**: Execute a Stata do-file • **Filename**: Copy a file to the command line • **Print**: Print log or graph • **Exit**: Quit Stata
Edit	• **Copy/Paste**: use to copy /paste of **text** among command, results and log windows • **Copy/Table**: use to copy a **table** in the results window to another file
View	Click here to view and manage data, add variable labels, change orientation of windows, view and modify graphs, etc.
Data	Click here to browse or edit data. From here you can use menus to obtain variable labels and discrete value code labels.
Graphics	Click here to obtain menu driven graphs (eg – histogram, XY scatter, etc)
Statistics	Click here to obtain menu driven statistical analyses (eg – normal theory linear regression, analysis of variance, logistic regression, etc)
User	Click here for menus of user-supplied Stata commands
Window	Click here to navigate among the windows
Help	Click here for help.

Toolbar

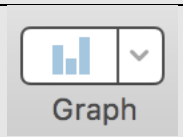
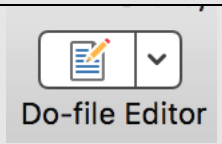
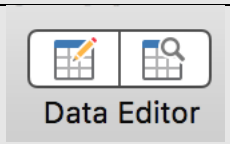
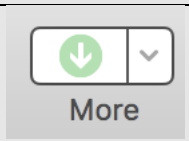


The default arrangement of windows and toolbars in Stata includes a tool bar located below the main menu. It provides short cuts to some frequently used facilities. Many of the icons, like the icons in the main menu bar, produce drop down menus from which you can make additional, related, selections,.

Key to Toolbar

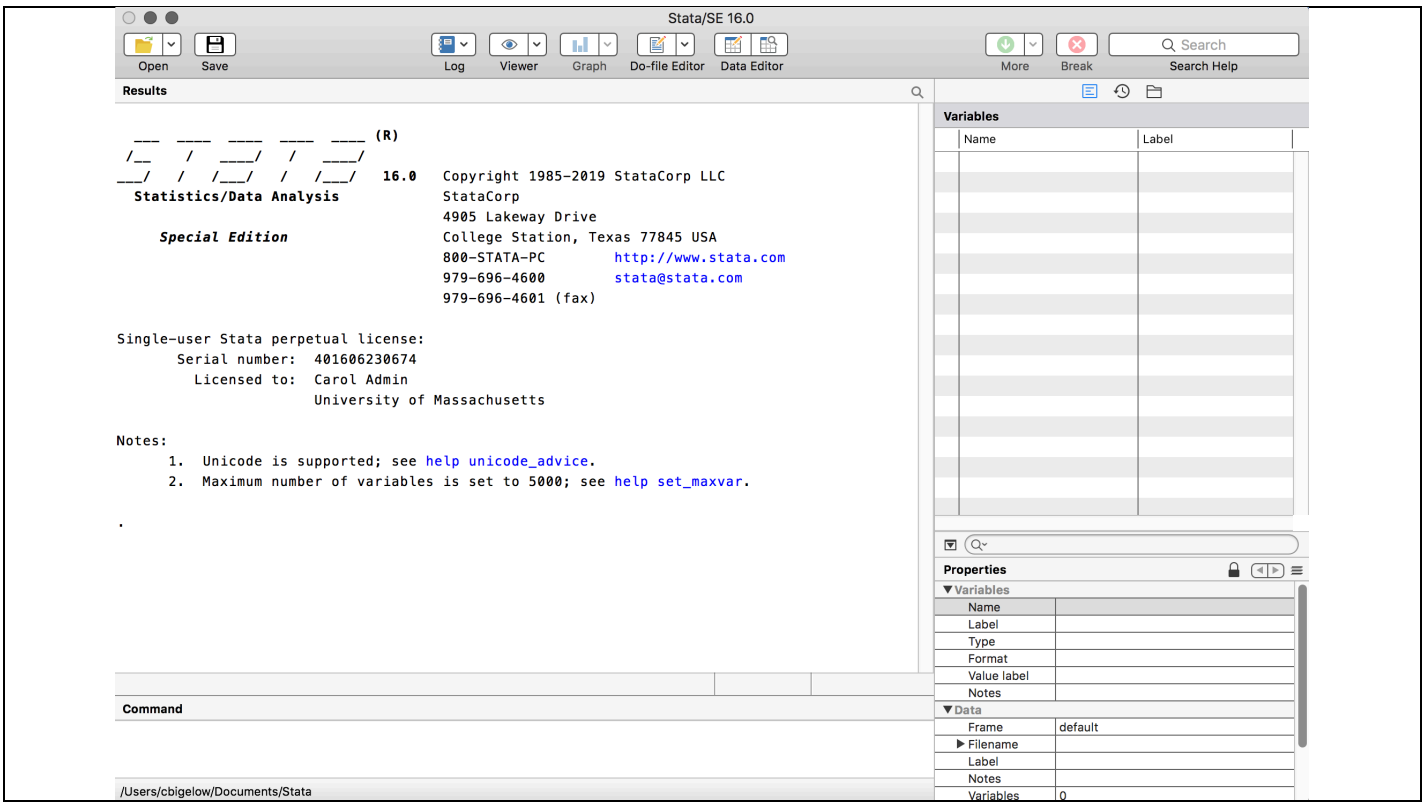
Tool	Description
	**Open** Use to open a Stata data set for use in Stata
	**Save** Use to save a Stata data set. See also the drop down menu for the **Save As** utility
	**Log** Use to create, pause, resume or exit a log of your session. This is an alternative to the FILE > LOG instruction from the main menu.
	**Viewer** The viewer tool is used, primarily, as a resource for obtaining help.

Key to Toolbar – *continued*

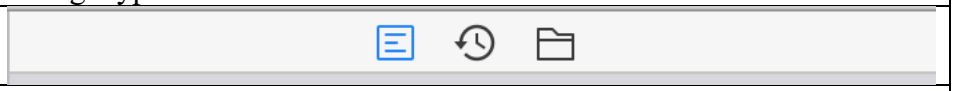
Tool	Description
	**Bring graph window to front** Click here to bring the graph window to the front. See also the drop down menu for specific graph selections.
	**Do File Editor** Click here to launch the do file editor. This is the window you activate to write your own Stata-do files
	**Data Editor(left) & Data Browser(right)** Click at left to create a new data set if there is no current data set or, to edit the current data set if it exists. **Tip** – ***Never use the data editor to make changes to your data set.*** Instead, make changes using commands in a do-file or log so that you save a record. Click at right to browse/view the data, without risk of inadvertently making changes.
	**More** This tells Stata to continue processing commands when it has paused.
	**Break** This tells Stata to stop the current command(s) it is executing

Four (+) Main Windows

You should see four main windows: (1) "Command" (lower left), (2) "Results" (upper left), (3) "Properties" (lower right), *plus* (4) at the upper right you will see a window with 3 little tabs on top




```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   16.0   Copyright 1985-2019 StataCorp LLC
  Statistics/Data Analysis             StataCorp
                                       4905 Lakeway Drive
     Special Edition                   College Station, Texas 77845 USA
                                       800-STATA-PC        http://www.stata.com
                                       979-696-4600        stata@stata.com
                                       979-696-4601 (fax)

Single-user Stata perpetual license:
       Serial number:  401606230674
         Licensed to:  Carol Admin
                       University of Massachusetts

Notes:
      1.  Unicode is supported; see help unicode_advice.
      2.  Maximum number of variables is set to 5000; see help set_maxvar.

.
```

	Window	Description
Lower left	**Command**	Type your commands here
Upper left	**Results**	Stata will show you output here, including error messages
Lower right	**Properties**	Use this to manage your variables – names, labels, notes, formats, and storage types
Upper right	**Varies because you have 3 choices (tab)**	
		Variables window: Use this to manage variables (names, labels etc)
		History window: Every command that is issued will appear in this history window. Good for either: re-using commands (simply highlight) or sending them to do-files (more on do-files later)
		Project Manager window:

Design ..... **Data Collection** ..... **Data Management** ..... **Data Summarization** ..... **Statistical Analysis** ..... **Reporting**

Tips for the Use of Windows

Stata provides other windows as well: Viewer, Data Editor, Data Browser, Do-file Editor, Graph, and Graph Editor. More on these later.

To navigate between windows and to select the window you wish to make active	From the main menu at top: **WINDOW** From drop down menu: choose window that you want to be active
To resize a window	Position cursor at the edge of the window. Click and drag.
Stata lets you choose the arrangement and order of the windows on your screen.	From the main menu at top: **VIEW> LAYOUT** From the drop down menu: Choose the orientation and order of the windows on your screen.

3.3 Basic Commands

Examples of Stata commands

Important Tip – You do not type the leading period. Stata provides this leading "dot" for you

Stata command	Description
.list state	Print the values of the variable state
.list state if (area==4) and (lung >=10)	Print the values of the variable state for the subset of observations for which the value of area =4 and the value of the variable lung is greater than or equal to 10.
.list state if (area==4) and (lung >=10), nolabel	Print the values of the variable state for the subset of observations for which the value of area =4 and the value of the variable lung is greater than or equal to 10. Implement the option that suppresses printing of label

Basic Structure of a Command

Every Stata command has a basic structure with elements that are specified in the correct order as follows

. Command *variablelist* if *expression*, options

- Command (possibly with additional subcommands), *followed by*
- Dependent variable, *followed by*
- Independent variable, *followed by*
- Expression/qualifier such as **in** and/or **if**, *followed by*
- Comma, *followed by*
- Option(s)

Simply type your command in the command window and press the enter/return key. ***Do NOT type a leading "dot". Stata provides this.***

DON'T FORGET!
Stata is case sensitive.

Design ..... Data Collection ..... Data Management ..... Data Summarization ..... Statistical Analysis ..... Reporting

(a) Tip! At the Start of Your Session (note - these are in suggested order…)

Description	Stata Command
Display the current directory (**pwd**) Set the "working directory" (**cd**)	To display current directory: • **pwd** Specify a new current directory using correct direction of slashes! • **MAC users use forward slashes** **cd "/Users/cbigelow/Desktop"** *** PC Users use backward slashes** **cd "c:\Users\cbigelow\Desktop"** ***Note on Use of QUOTES* – The surrounding quotes are only needed when your full pathname has blanks in it. I recommended always using the quotes, just to be safe.**
Start a log of your session. Close the log of your session	• **log using *yourfilename*** • **log close**
Suppress automatic pausing of results, screen by screen Restore pausing of results	• **set more off** • **set more on**
Increase the allocation of memory - **Notes - The default allocation is 1 MB if you are using Stata/IC. Be careful -You cannot change the memory allocation if there is already data in memory. In the example at right, I have increased the allocation of memory from 1 MB to 25 MB.**	If you have not yet opened a data set: **. set memory 25m** If you have an active data set already in use: **. save *olddatafile*** **. clear** **. set memory *25m***

(b) Tip! Use the help command often!

Description	Stata Command
For help with a particular command	• **help *particularcommand***
For help with help	• **help**
For list of help topics	• **help contents**
To search the help document for a particular topic	• **search *particulartopic***
To obtain all references to a topic, both on-line and in the Stata manuals	• **lookup *particulartopic***

(c) Stata Operators

Type	Operators in Stata
Basic arithmetic	**+, -, *, /, ^ (power)**
Basic relational	**>, >=, <, <=, ==(equal), != (not equal)**
Logical	**&(and), |(or), !(not)**
Assignment	**=**
Concatenation	**+**

(d) Mathematical Functions

Description	Stata Command
Absolute value	**abs(x)**
Truncations	**int(x) or round(x)**
Combinatorial "n choose k"	**comb(n,k)**
Exponentiation	**exp(x)**
Natural logarithm	**ln(x) or log(x)**
Base 10 logarithm	**log10(x)**
Maximum, minimum	**max(x) min(x)**
Sign	**sign(x)**
Square root	**sqrt(x)**
Sum	**sum(x)**

(e) Stata as a Hand Calculator. Begin command with display or di

Example	Example
Print out the value of the Stata system variable for the constant π.	. display _pi
Print the value of an arithmetic calculation such as that at right	. display 5*5*_pi ^2

(f) Probability Distribution Calculators.

Description	Stata Command
Binomial Distribution with n trials and event probability p: (1)Probability of **exactly k** events (2) Probability of **k or more** events (3) Probability of **k or fewer** events	**(1) . display binomialp(n,k,p)** **(2) . display binomialtail (n,k,p)** **(3) . display binomial (n,k,p)**
Chi Square Distribution with degrees of freedom equal to df: (1)Probability **X <= x** (2) Probability **X >= x** (3) The **pth quantile**	**(1) . display chi2(df,x)** **(2) . display chi2tail (df,x)** **(3) . display invchi2 (df, p)**
F Distribution with degrees of freedom equal to df1 and df2: (1)Probability **X <= x** (2) Probability **X >= x** (3) The **(1-p)th quantile** ***Note – This works a little differently than the previous examples. Here if you want the 95th percentile, this corresponds to the upper 5%. So the function requested would be invFtail(df1, df2,.05)***	**(1) . display F(df1,df2, x)** **(2) . display Ftail (df1,df2,x)** **(3) . display invFtail (df1,df2, p)** ***Tip! In all of these F distribution calculations, be sure the "F" is capital-ized. Stata is case sensitive***
Standard Normal Distribution with mean=0 and variance=1: (1)Probability **Z <= z** (2) The **pth quantile**	**(1) . display normal(z)** **(2) . display invnormal(p)**
Student t Distribution with degrees of freedom df: (1)Probability **X >=x** (2) The **(1-p)th quantile** ***Note – This works a little differently than the previous examples. Here if you want the 95th percentile, this corresponds to the upper 5%. So the function requested would be invttail(df,.05)***	**(1) . display ttail(df,x)** **(2) . display invttail(df,p)**

Examples *(green – comments, black – commands, blue – output)*

```
. * This is a comment.  It begins with an asterisk.
. * Illustrations of using the DISPLAY command

. * What is the probability of 7 successes in 20 trials with event prob = .5?
. display binomialp(20,7,.5)
.07392883

. * What is the 97.5th percentile of a Normal(0,1)?
. display invnormal(.975)
1.959964


. * What is the 97.5th percentile of a Student t with 30 degrees of freedom?
. display invttail(30,.025)
2.0422725

. * What is the 5th percentile of an F distribution with numerator df = 12 and
denominator df = 14? Hint:  Set upper tail area to .95
. display invFtail(12,14,.95)
  0.37920104

. * What is the 95th percentile of an F distribution with numerator df = 12 and
denominator df = 14? Hint:  Set upper tail area to .05
 display invFtail(12,14,.05)
  2.5342433
```

Tip when using command display – Consider including an embedded quote that explains the calculation so that your results are more readable. For example

```
. * Binomial probability of 7 successes in 20 trials with event prob = .5

. display " Pr [7 succcesses in 20 trials w event prob = .5 ] =   "
binomialp(20,7,.5)

 Pr [7 succcesses in 20 trials w event prob = .5 ] =   .07392883
```

4. Getting Data In and Out of Stata

Before you begin, don't forget:

__1. Your path specification will be different
__2. MAC path specifications using forward (/) slashes.
__3. PC path specifications use backward (\) slashes

Example
. use "z:/bigelow/teaching/690c/Stata/exercise1.dta", clear

Example
. cd z:/bigelow/teaching/690c/Stata
. use exercise1.dta, clear

Data can be input into Stata via:

(4.1) an existing Stata data set;
(4.2) importing an excel file;
(4.3) conversion from another program (such as SAS), or
(4.4) the creation of a new data set within Stata using the data editor.

4.1 Input an Existing Stata Data Set

Tip! Set your current working directory first by using the cd command (see page 19)

From the toolbar: Click on **OPEN**	Stata will open the current directory for you. From there, browse and click.
From the command window if you ***have*** set the current directory; . use ***"filename"***, clear ←	"clear" clears the working directory
From the command window if you have ***NOT*** set the current directory: . use ***"fullpath and filename"***, clear	

4.2 Import Data from Excel (including dates)

FILE > IMPORT > Excel Spreadsheet

It gets easier and easier as new versions of Stata become available. In version 16, there are at least three ways to import an excel spreadsheet:

METHOD I: Using the drop-down menus (recommended)
METHOD II: Copy and paste (recommended for very small excel datasets only)
METHOD III: Command driven (good as you become more expert in Stata)

Example –

We will use the Excel data set **playdata.xlsx**. It has n=6 observations on 6 variables. You can download it from the course website.

studyid	favorite	mpg	date_birth	date_today	age_excel
1	chocolate	21.65	6/9/1956	9/16/2020	64.27
2	avocado	31.00	10/1/1953	7/4/2020	66.76
3	shrimp	50.62	10/14/2002	4/11/2020	17.49
4	pistachios	28.76	11/4/1951	6/18/1999	47.62
5	gruyere	22.40	7/5/2015	5/15/2020	4.86
6	oj	18.90	1/12/1968	9/12/2020	52.67

CHECKLIST: Important Preliminaries in Excel:

__1. Stata expects a **simple spreadsheet** (matrix) with rows defining observations and columns defining variables

__2. In excel, be sure your **column headings** (these will be your variable names in Stata) follow the **Stata rules for naming variables**: NO blanks, no special characters (the exception is the underbar character _), etc. ***TIP***- use all lowercase (Stata is case sensitive)

__3. In excel, ***before you save***, be sure to **format your columns of data** as appropriate. Recommended:

text; or

number (then you specify number of digits after the decimal point); or

date (then you specify the type)
IMPORTANT- Be sure the "type" selection you choose matches the way you entered dates

Example – *continued*

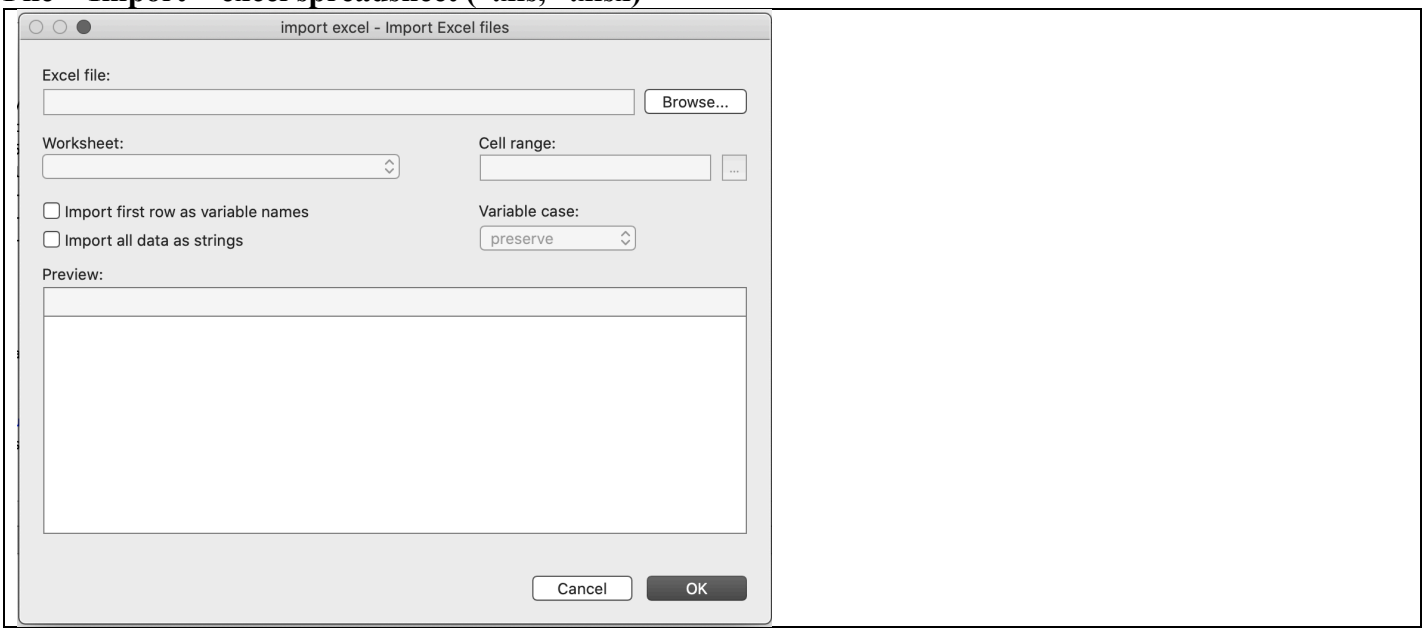
In Excel, column heading → Stata variable name	In Excel, format cells as	And specify (still in Excel)
studyid	number	decimal places: 0
favorite	text	-
mpg	number	decimal places: 2
date_birth	date	* type: 3/14/2012
date_today	date	type: 3/14/2012
age_excel	number	decimal places: 2

* ***IMPORTANT***- Be sure the "***type***" selection you choose matches the way you entered dates

__4. In excel, don't forget to ***save.***

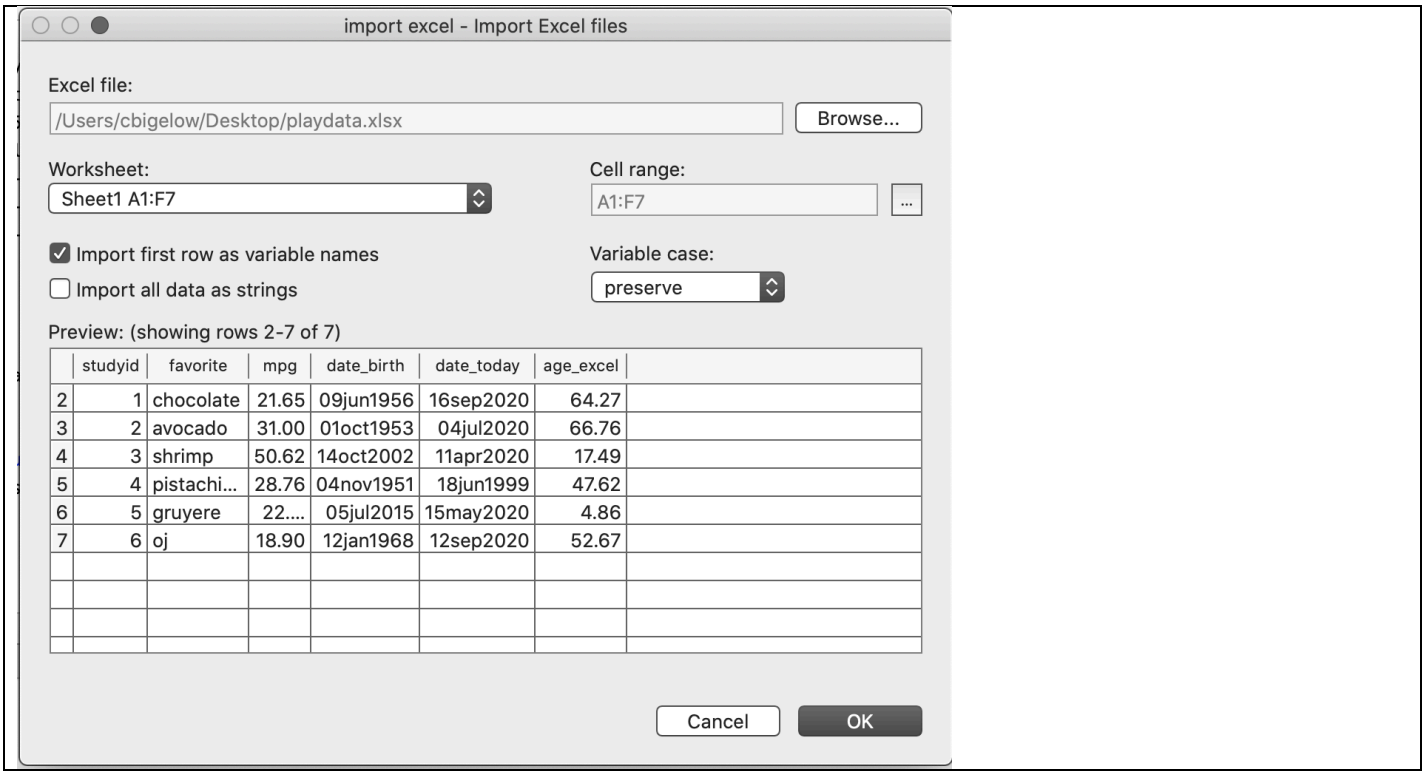
METHOD I: Using the drop-down menus (recommended)

File > Import > excel spreadsheet (*.xls, *.xlsx)



From the import dialog box:

- Browse to select your excel file
- (if need be) select the "worksheet"
- (if need be) choose cell range
- Be sure to check the box: "import first row as variable names"
- Take a look at the "Preview" to make sure everything is alright.
- All is well? Click at bottom right on **OK**



	studyid	favorite	mpg	date_birth	date_today	age_excel
2	1	chocolate	21.65	09jun1956	16sep2020	64.27
3	2	avocado	31.00	01oct1953	04jul2020	66.76
4	3	shrimp	50.62	14oct2002	11apr2020	17.49
5	4	pistachi...	28.76	04nov1951	18jun1999	47.62
6	5	gruyere	22....	05jul2015	15may2020	4.86
7	6	oj	18.90	12jan1968	12sep2020	52.67

Stata rewards you by showing you the command that was executed. ***Tip*** – Select this command from your history window and send it to a DO file for re-using later.

```
. import excel "/Users/cbigelow/Desktop/playdata.xlsx", sheet("Sheet1") firstrow
```

Check to make sure that all is well, using the command describe

```
. describe

Contains data from /Users/cbigelow/Desktop/playdata.dta
  obs:             6
 vars:             6                          16 Sep 2020 14:15
-------------------------------------------------------------------------------------------------------------

              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------------------------
studyid         byte    %10.0g                studyid
favorite        str10   %10s                  favorite
mpg             double  %14.2f                mpg
date_birth      int     %td                   date_birth
date_today      int     %td                   date_today
age_excel       double  %14.2f                age_excel
-------------------------------------------------------------------------------------------------------------
```

Also check that the dates were imported correctly.
Here, I create a new variable called **age_stata** and then compare its values to the values of **age_excel**. They should match.

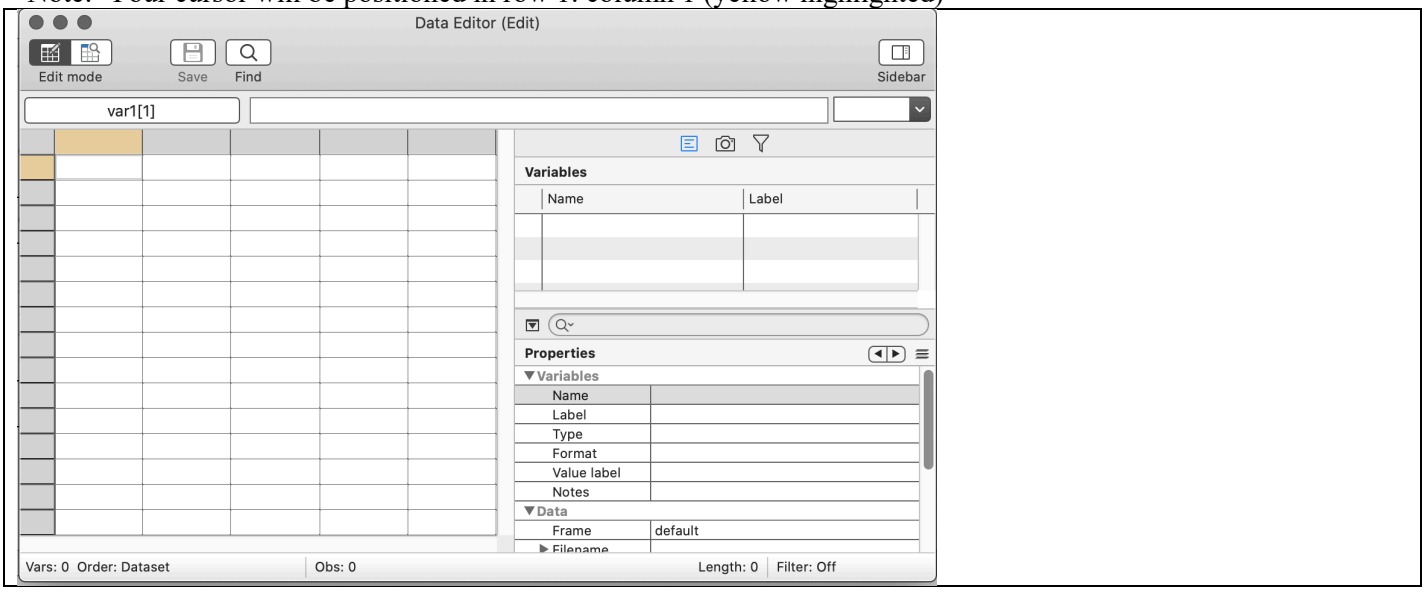
```
. generate age_stata = (date_today - date_birth)/365.25
. list studyid age_excel age_stata

     +-------------------------------+
     | studyid   age_ex~l   age_st~a |
     |-------------------------------|
  1. |       1      64.27   64.27105 |
  2. |       2      66.76   66.75702 |
  3. |       3      17.49   17.49213 |
  4. |       4      47.62   47.61944 |
  5. |       5       4.86   4.862423 |
     |-------------------------------|
  6. |       6      52.67   52.66804 |
     +-------------------------------+                Hooray
```

FINALLY!!! Be sure to save your imported data to a Stata dataset

```
. * Save imported excel data to stata dataset called playdata.dta
. save "/Users/cbigelow/Desktop/playdata.dta"

file /Users/cbigelow/Desktop/playdata.dta saved
```

METHOD II: Copy and paste (recommended for very small excel datasets only)

In Stata, open a new blank Stata spreadsheet
Data > Data Editor > Data Editor (edit)
- Note: Your cursor will be positioned in row 1: column 1 (yellow highlighted)



Copy from Excel and Paste into Stata
In Excel, select cells, EDIT > COPY (control/command – C)
In Stata (with your cursor positioned in row 1: column 1), EDIT > PASTE (control/command-V)
Stata will ask you if the first row should be used as variable names. Click on **variable names**
You should now see:

	studyid	favorite	mpg	date_birth	date_today	age_excel
1	1	chocolate	21.65	6/9/1956	9/16/2020	64.27
2	2	avocado	31	10/1/1953	7/4/2020	66.76
3	3	shrimp	50.62	10/14/2002	4/11/2020	17.49
4	4	pistachios	28.76	11/4/1951	6/18/1999	47.62
5	5	gruyere	22.4	7/5/2015	5/15/2020	4.86
6	6	oj	18.9	1/12/1968	9/12/2020	52.67

Key:
black: numeric
red: string

Oh, too bad – the dates are all wrong!!! With the cut and paste methods, Excel dates come in as string.

```
. describe

Contains data
  obs:             6
 vars:             6
-------------------------------------------------------------------------------------------------------------

              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------------------------

studyid         byte    %8.0g
favorite        str10   %10s
mpg             float   %8.0g
date_birth      str10   %10s       → We do NOT want storage type = str10 (this is string!)
date_today      str9    %9s        → Same issue.  We do not want storage type = str9 (also string!)
age_excel       float   %8.0g
```

Here is the solution **(painfully annoying and suggesting that if your Excel file has dates, never do CUT AND PASTE!)**
How to Convert excel dates that come in as strings to date. **Hack:** Take care to allow for the possibility that you may have some dates in the 1900's and some in the 2000's.

```
. * If last 2 digits of year are less than 21, then assume year is 20xx and use "MD20Y"(Start with this)
. * If last 2 digits of year are > 21, then assume year is 19xx century and use "MD19Y" (Fix if needed)

. generate date_birth2 = date(date_birth, "MD20Y")
. replace date_birth2=date(date_birth, "MD19Y") if substr(date_birth,-2,2) > "21"

. generate date_today2 = date(date_today, "MD20Y")
. replace date_today2=date(date_today, "MD19Y") if substr(date_today,-2,2) > "21"

. generate age_stata = (date_today2 - date_birth2)/365.25

. list studyid age_excel age_stata, clean

     studyid   age_ex~l   age_st~a
  1.       1      64.27   64.27105
  2.       2      66.76   66.75702
  3.       3      17.49   17.49213
  4.       4      47.62   47.61944
  5.       5       4.86   4.862423
  6.       6      52.67   52.66804
```

METHOD III: Command driven (good as you become more expert in Stata)

In method I, we used the tool bar and drop-down menu to import the Excel data set **playdata.xlsx**. This produced the command in the command window:

```
. import excel "/Users/cbigelow/Desktop/playdata.xlsx", sheet("Sheet1") firstrow
```

Tip – A good practice is to copy this command into your DO file. More on Do files in Section 9. Stay tuned.

If you want to learn more details and options pertaining to importing and exporting Excel data, type the following into the **command window**: **help import excel.** From the help screen, at the top, then click on **View complete pdf manual entry.** This will bring you to the online manual. There's a nice Quick Start to get you on your way:

Quick start

Check the contents of Excel file `mydata.xls` before importing

```
import excel mydata, describe
```

As above, but for `mydata.xlsx`

```
import excel mydata.xlsx, describe
```

Load data from `mydata.xls`

```
import excel mydata
```

As above, but load data from cells A1:G10 of `mysheet`

```
import excel mydata, cellrange(A1:G10) sheet(mysheet)
```

Read first row as lowercase variable names

```
import excel mydata, firstrow case(lower)
```

Import only `v1` and `v2`

```
import excel v1 v2 using mydata
```

Save data in memory to `mydata.xls`

```
export excel mydata
```

As above, but export variables `v1`, `v2`, and `v3`

```
export excel v1 v2 v3 using mydata
```

4.3 Output Data as a Stata Data Set

There are at least two ways to accomplish this:

Method 1: From menu bar, using **FILE > SAVE AS**
Method 2: In the command window using command **save**.

Method I

Tip! Set your current working directory first by using the cd command (see page 20)

From the toolbar: Click on **SAVE**	Stata will open the current directory for you. From there, browse and click.
From the command window if you ***have*** set the current directory; . save ***"filename"***, replace	Stata supplies the extenstion ".dta" for you. "replace" tells Stata to overwrite any existing data set with the same name.
From the command window if you have ***NOT*** set the current directory: .save ***"fullpath/filename"***, replace	

METHOD II Example (one step – specify full path)

- *** Mac Users**
  . save "z:/bigelow/teaching/690c/Stata/exercise1.dta", replace

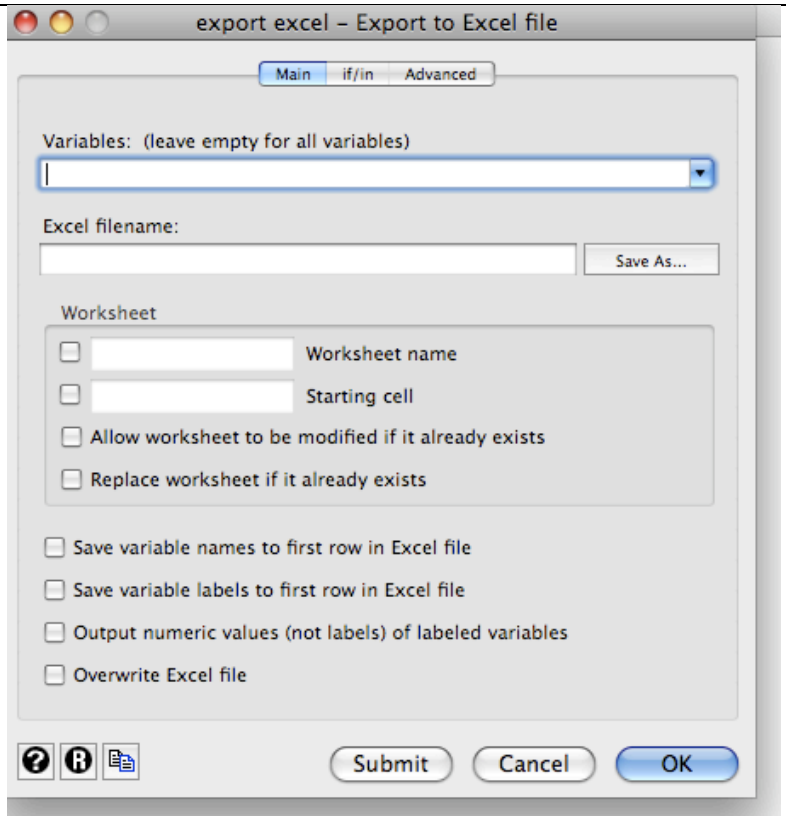
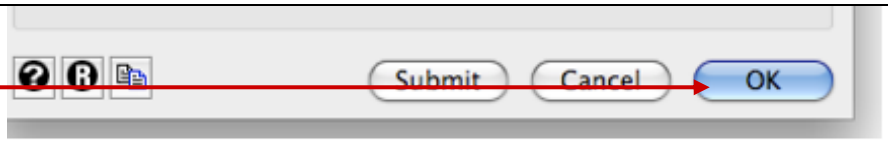
  PC Users
- **. save "z:\bigelow\teaching\690c\Stata\exercise1.dta", replace**

METHOD II Example (two steps – change directories first, then save)

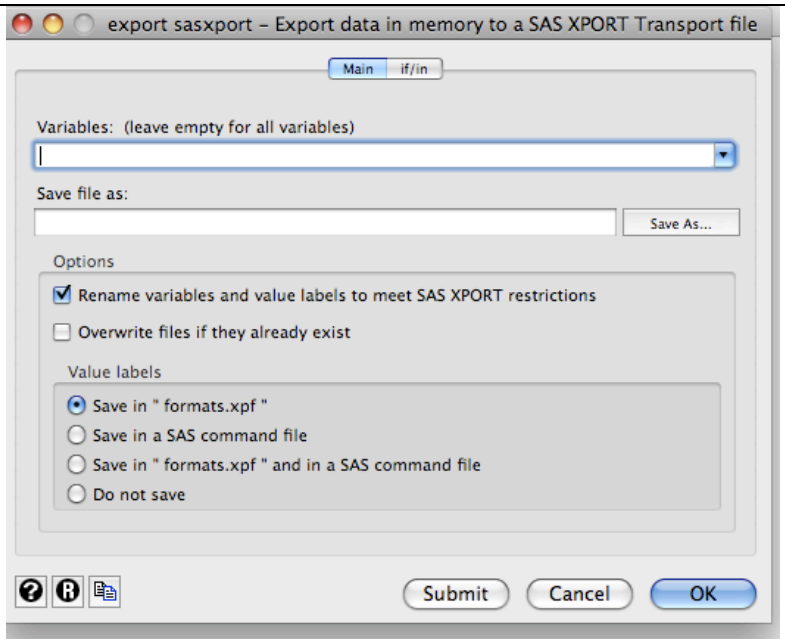
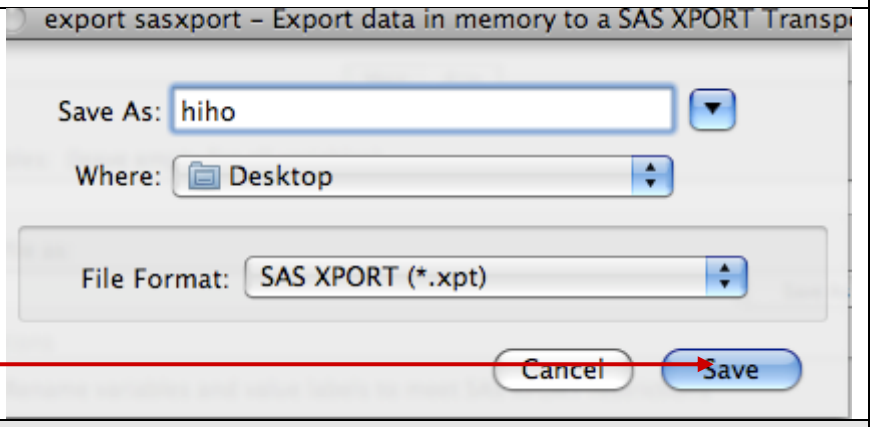
. * Mac Users
. cd "z:/bigelow/teaching/690c/Stata"
. save "exercise1.dta", replace

. * PC Users
. cd "z:\bigelow\teaching\690c\Stata"
. save "exercise1.dta", replace

4.6 Export Data to Excel

Important Preliminary **In Stata:** Be sure you have saved your data as a Stata data set if you want to keep a Stata version.	
Important Preliminary Decide which variables you wish to export.	***Note – Otherwise, Stata will export to Excel the data for every variable in your Stata data set.***
Step 1: **FILE > EXPORT > Data to Excel Spreadsheet (*.xls, *.xlsx)** Variables box: Decide which variables you want to export Excel filename box: Enter the excel filename you want. It is NOT necessary to type the extension .xlsx Worksheet selections: As you like!	
Finished with your selections? Click **OK**	

4.7 Export Data to SAS

Important Preliminary **In Stata** Be sure you have saved your data as a Stata data set if you want to keep a Stata version.	
Important Preliminary Decide which variables you wish to export to SAS.	***Note – Otherwise, Stata will export to SAS the data for every variable in your Stata data set.***
In Stata **FILE > EXPORT > SAS XPORT**	
In the EXPORT SASXPORT Window: Again, four things to do here: (1) Save As: Type in your filename choice (2) Where: Use drop down to choose location (3) File Format: Choose SAS XPORT. Last but not least, don't forget to save! (4) Click **SAVE.**	

5. Working with Variables

5.1 Variable Types and Storage

In BIOSTATS 540, we distinguished a variety of types of measurement: **discrete** versus **continuous**. Within each of these broader headings of measurement type, we learned sub-types. These distinctions were important because, depending on the variable type, some analyses are possible while others are not.

BIOSTATS 540 Introductory Biostatistics Course

Measurement Type	Sub-Type
Discrete	Nominal Ordinal
Continuous	Interval Ratio

Stata functions a little differently. It distinguishes two types of data: **numeric** and **string**. This distinction is important for reasons having to do with ***data storage***.

Stata

Data Type	Storage Type Name	Storage Used in bytes
Numeric: **(Integer, ratio, date/time, missing)**	Byte Int Long Float* Double	1 2 4 4 8
String	Str1 to str244	1 to 244

***The default storage of numeric data in Stata is as 'float'; it provides 7 digits of accuracy after the decimal point. This is plenty for most (but not all) purposes**

Tip! Enclose string variable values in quotes.

Tip!
When creating data sets, especially large ones, choose a storage selection that uses only the space that you actually need ("parsimony"). **For help with changing storage type, see description of the command recast on page 37**

Float, Double - Use for continuous measurements
Byte, Int, Long – Use for measurements that are integer valued only
Str# - Use for measurements that are string valued (text) only

Design ..... Data Collection ..... Data Management ..... Data Summarization ..... Statistical Analysis ..... Reporting

Tips for Data Management

Description	Stata Command
codebook Use the command **codebook** to learn the storage type of every variable in your data set or for the storage type of a selection of variables. Option **mv** **Tip!** Request the option **mv** to obtain information about the pattern of missing values	For the entire data set **. codebook, options** For a selection of variables **. codebook *variable variable,* options** **Example** **. codebook gender age, mv**
encode **String to Numeric**: Use the command **encode** to create a new variable that is numeric from an old variable that is string ***provided*** that the old variable does ***NOT*** contain numbers such as 0 or 1 that happen to be stored as string. How it works - The string values of the old variable are assigned numeric values of 1, 2, etc. for the new variable according to the alphabetic order of the string values. .	**. encode *stringvariable*, generate (*numericvariable)*** **Example** Suppose the string variable is SEX with possible string values of "male" and "female". We wish to create a numeric variable called GENDER. The instruction to Stata is the following. Notice the use of the codebook command to check our work! **. * Create numeric (gender) from string (sex)** **. encode sex, generate(gender)** **. codebook gender**
decode **Numeric to String**: Use the command **decode** to create a new variable that is string from an old variable that is numeric. Note – the numeric variable must be formatted with labels for this to work You ***must*** use either the option **generate** or the option **replace**	**. decode *numericvariable*, generate(*stringvariable*)** **. * Create string (sex) from numeric (gender)** **. label genderf 1 "male" 2 "female"** **. label values gender genderf** **. decode gender, generate(sex)**

Tips for Data Management - *continued*

Description	Stata Command
recast Use the command **recast** to change the storage type of variables **Tip!** Choose storage types that are parsimonious.	For a selection of variables **. recast *newtype variablelist,* options** Newtype choices are **byte** **int** **long** **float** **double** **str1, str2, …, str244** **TIP -** Do **not** use the option **force** If you use the "force" option, Stata will change the storage selection to your choice regardless of whether or not precision is lost. **Example** **. recast byte age height weight** **Example** **. recast string6 sex**

5.2 Generating New Variables

The Single Equal Sign
v
The Double Equal Sign

Use the single equal = when you are defining variable values

Use the double equal = = when you are checking a true/false condition

Example – There is no condition here, so there is no need for a double equal sign.
. generate hourly = wkly_earn/wkly_hours

Example – In this example, there is a condition and so the use of the **double equal** sign is necessary.
. generate northeast=1 if region==1

There are multiple ways to create a new variable:

(1) From the main menu: **Data > Create or change variables**; or
(2) From the command window: .**generate [type] newvariable = expression if condition**
(3) From the command window: . **egen newvariable = expression if condition**

The command **generate** can be abbreviated as **gen** or as **g**.
The command **egen** is a special extension of the generate command. More on this below.

Some Helpful Tips …

In practice, the command **generate** is often followed by the command **replace**. Sometimes, the command generate is replaced by the command **recode**. See examples on the following pages.

Beware of the handling of missing values whenever you create a new variable. Former SAS users should note the following:

In Stata: missing values are greater than (to the right of) valid numbers. This is in contrast to
In SAS: missing values are less than (to the left of) valid numbers

STATA Variable Naming Rules

(1) Begin variable name with a character
(2) Use ONLY letters or numbers or the underscore (_)
(3) Do NOT use any other special characters in the variable name (such as !@#$)
(4) Variable names must be no longer than 32 characters

Some Additional Suggestions …

(5) Use as short a name as possible; 8-10 characters or less is ideal
(6) Choose a name that is as descriptive as possible
(7) Avoid names that are the same as Stata names for commands or functions

Always Handle Missing Values Explicitly

The condition **!missing(*oldvariable*)** ensures that only non-missing values of the oldvariable will be used in the creation of the new variable. Use the following syntax:

.generate *newvariable* = expression if !missing(*oldvariable*)

Example

```
. * Code obese with the value of 1 (for "true") for values of bmi ≥30 but only when bmi is NOT missing
. generate obese = (bmi>=30) if !missing(bmi)
```

Selected Variable Creation Commands.

	Stata Command
Generate Use the command **generate** to create a new variable as a function of an existing variables	**. generate *newvariable*=expression(*oldvariable*) if !missing(*oldvariable*)** **Example** **. generate obese = bmi>=30 if !missing(bmi)**
Generate, Replace Use the command **generate** *followed by* the command **replace** to create a new variable that has a different value depending on multiple ***discrete*** conditions	**. generate *newvariable*=expression(*oldvariable*) if *oldvariable* = = expression** **. replace *newvariable*=expression(*oldvariable*) if *oldvariable* = = expression** **Example** **. generate predprice=1.05*price if !missing(foreign)** **. replace predprice=1.10*price if foreign= = 1**
bysort Use the command **bysort** when you want to generate a new variable separately for each sub-group defined by your sort variable	**. bysort *sortvariable*: generate *newvariable*=expression(*oldvariable*)** **Example** **. bysort gender: generate obsno=_n** This example produces a separate numbering of subjects within each sex and makes use of the Stata system variable _n.
egen Use the command **egen** to create a new variable that is a function of some selected arguments **Tip! Egen is wonderful for saving summary statistics as new variables** **Tip! Egen is wonderful for creating new variables from repeated measures**	**. egen *newvariable*=function (*variablelist*)** **Example** **. egen meanage=mean(age)** This example produces the mean age taken over all the observations in the data set **Example** **. by gender: egen meanage=mean(age)** This example produces the mean age taken over all the observations in the data set, separately by gender. **Example** **. egen bmimax=rowmax(bmi_base bmi_t1 bmi_t2)** This example produces the maximum bmi recorded for each individual with respect to his/her three repeated measures, baseline, T1 and T2.

Selected Variable Creation Commands - continued.

	Stata Command
Generate, Recode Use the command **generate** *followed by* the command **recode** to create a new variable that has discrete value codes **Example** **Create a grouped measure of age called agegrp and that has values of 1, 2 or 3 depending on age (<18, 18-64, ≥65, respectively)**	**. generate *newvariable* =*oldvariable* if !missing(*oldvariable*)** **. recode *newvariable* (expression) (expression)** **Example** **. generate agegrp = age if !missing(age)** **. recode agegrp (min/17=1) (18/64=2) (65/max=3)**

5.3 Missing Values

Missing values are not used in analysis.

From a data management perspective, missing values are of interest.
For example, we might want to know the number and relative frequency of missing values in our dataset. Or, we might want to know how many missing values are of the type "don't know" or of the type "blank" or of the type "refused". Etc. Stata provides utilities that accommodate these possibilities.

Stata stores a missing value with a leading period.

(1) A Stata system missing value is: .

(2) User defined missing values are: .a, .b, .c, etc.

Tip! Sometimes it is helpful to create user defined missing values.

Example -

Reason for Missing Value	User Defined Missing Value*
Item left blank	.
"Don't know"	.d
Refused	.r
Skipped	.s

* These are just suggested conventions. Choose what works for you!

Important – STATA and SAS are Different! Stata treats missing values as being greater than actual numbers; this is the opposite of what SAS does. This is relevant when you are creating new variables based on the value of one or more source variables

In Stata, missing values are always LARGER than (to the right of) valid values

All valid numbers < . < .a < .b < … < . z

Be very careful in the use of logical expressions as missing values may be inadvertedly recoded to a new variable value that is non-missing. More on this in the next pages.

Tip!

Always Handle Missing Values Explicitly
Use the Stata command **mvdecode** to declare your missing values. In doing so, observe the missing value conventions that Stata uses. Separate each missing value designation by a back slash **. mvdecode *variablename*, mv(*number=missingcode\number=missngcode*)** **Be sure to begin each Stata missing code with a period.**

Example

A survey queries age at menstruation. The variable name is **age_men**. Three types of missing values are recorded, depending on the response given:

997 for missing because "not applicable"
998 for missing because "don't remember"
999 for missing because "blank"

You would like to use the following correspondence:

Missing value on survey	Missing value designation in Stata
997	.n
998	.d
999	.

The following command syntax would instruct Stata to denote these three types of missing values

. mvdecode *age_men*, mv(*997=.n\998=.d\999=.*)

5.4 Indicator and Design Variables

Recall that

(1) A 0/1 ***indicator variable*** has value = 1 when the trait of interest is present and the value =0 otherwise.

Example: Indicator of female gender
I_female = 1 if subject is female
0 otherwise

(2) ***Design variables*** are a set of 0/1 indicator variables that "flag" the separate values of a discrete variable. Usually the source discrete variable is nominal, but sometimes it is ordinal.

If the source variable has ***k*** possible values, then the number of design variables needed is ***(k-1)***. The response without an explicit indicator variable is the '***referent***'

Example: Design variables for season. K=4 means 3 design variables needed.

Season = 1 if winter
2 if spring
3 if summer
4 if fall

I_spring = 1 if (season=2) **I_summer = 1 if (season=3)** **I_fall = 1 if (season=4)**
0 otherwise **0 otherwise** **0 otherwise**

Regression analysis exploring the influence of season out some outcome would include I_spring, I_summer, and I_fall as predictors but not season

Some Helpful Tips …

(1) Stata has utilities that will create design variables for you. Avoid these unless you are really sure of what you are doing. **Tip!** ***Define your design variables yourself***.

(2) ***Take care in choosing your referent group*** . Choose the category or value that makes good sense to you as referent; eg – 'not exposed', 'untreated', 'youngest age category'.

(3) If there is no clear choice of 'referent' consider choosing as 'referent' the category that is most frequent.

(4) **REMEMBER!!!** ***It is very easy to make a mistake in the handling of missing values***. Don't forget that in Stata: **valid numbers < missing value codes**

Indicator and Design Variable Creation

	Stata Command
Generate, Replace Use the command **generate** *followed by* the command **replace** **Example** **Create a 0/1 indicator of retirement that is equal to 1 if age ≥ 65 and is equal to 0 otherwise.** **Example** **Create a 0/1 indicator of child that is equal to 1 if age < 18 and is equal to 0 otherwise. Note that, here, I chose to do explicit coding of missing and note use of two equal signs**	**. generate *newvariable* = 1 if *oldvariable* = = expression & !missing(*oldvariable*)** **. replace *newvariable*= 0 if *oldvariable* = = expression** **Example** **. generate retire=1 if age >= 65 and !missing(age)** **. replace retire=0 if age < 65** **Example** **. generate child=1 if age < 18** **. replace child=0 if age >=18** **. replace child=. if age ==.**
Using logical operators Indicator variables can also be created using logical operators Take care to handle missing values explicitly **Example** **Create a 0/1 indicator of child that is equal to 1 if age < 18 and is equal to 0 otherwise. Note again the explicit coding of missing values and use of two equal signs.**	. **Example** **. generate child= (age < 18)** **. replace child=. if age ==.**

Tip!
Always check your variable creation

Example -
In this example, a cross-tab of the source variable against the indicator variable serves as a check of the creation of the indicator variable. Note that values of string variables are enclosed in quotes (more on string variables in Section 3.5). Also note the use of the option ***missing*** in the ***tabulate*** command.

green-comment black-command blue-output

```
. * Source variable is the variable sex and is a string variable
. list sex

     +--------+
     |    sex |
     |--------|
  1. | female |
  2. | female |
  3. |   male |
  4. |   male |
  5. |   male |
     |--------|
  6. |      . |
     +--------+

. * Create 0/1 indicator of female gender

. generate I_female=(sex=="female") if !missing(sex)

. tabulate sex I_female, missing

           |            I_female
       sex |         0          1          . |     Total
-----------+---------------------------------+----------
         . |         0          0          1 |         1
    female |         0          2          0 |         2
      male |         3          0          0 |         3
-----------+---------------------------------+----------
     Total |         3          2          1 |         6
```

5.5. How to Create Quantile Groups

Sometimes, it is of interest to record for each individual the quantile group of their measured value on some variable.

Example – Our data includes measurements of the continuous variable **age**, measured in years. We want to work with the associated quantile groupings rather than with the values of age themselves.

METHOD 1

(source: IDRE UCLA)
Use the command **egen** with the **cut()** function to make a variable called **agecat** that groups the variable write into the following 4 categories.

*** egen group_variable = cut(continuous_variable), at(30,40,50,60,70)**
egen agecat = cut(age), at(30,40,50,60,70)

Note: This is what you get
30 up to (but not including) 40
40 up to (but not including) 50
50 up to (but not including) 60
60 up to (but not including) 70

METHOD 2 – Better because you get group code values starting at 0 (possibly your referent?)

(source: IDRE UCLA)
Use the same command as above but now include the **icodes** option. This will instruct **cut()** to create integer codes beginning with 0. In the example below, you can see that it created codes 0, 1, 2 and 3.

egen agecat = cut(age), at(30,40,50,60,70) icodes
table agecat, contents(min write max write)

```
----------------------------------
   agecat | min(age)      max(age)
----------+-----------------------
        0 |         31          39
        1 |         40          49
        2 |         50          59
        3 |         60          67
----------------------------------
```

METHOD 3 – Best (imho) Elaboration that replaces simple grouping indicator with midpoint of the interval.

This is handy. The following is a two-step solution that, first, obtains quantile groupings and then, second, replaces the grouping indicators with the corresponding midpoints of the associated the intervals.

Step 1: Obtain values of 0th, 20th, 40th, 60th, 80th, and 100th percentiles

```
. * solve for P0, P20, P40, P60, P80, P100
. centile age, centile(0,20,40,60,80,100)

                                                      -- Binom. Interp. --
    Variable |       Obs  Percentile    Centile        [95% Conf. Interval]
-------------+-------------------------------------------------------------
         age |     1,000           0          30              30          30*
             |                    20        37.2              37          38
             |                    40          42              42          43
             |                    60          48              47          49
             |                    80        54.8              54          55
             |                   100          66              66          66*
```

Step 2: Create ordinal indicator of quintile of age, **agegroupi**

```
. egen agegroupi = cut(age), at(30,37.2,42,48,54.8,66) icodes
(1 missing value generated)
. fre agegroupi

agegroupi
-------------------------------------------------------------
              |      Freq.    Percent      Valid       Cum.
--------------+----------------------------------------------
Valid   0     |         200      20.00      20.02      20.02
        1     |         169      16.90      16.92      36.94
        2     |         212      21.20      21.22      58.16
        3     |         219      21.90      21.92      80.08
        4     |         199      19.90      19.92     100.00
        Total |         999      99.90     100.00
Missing .     |           1       0.10
Total         |        1000     100.00
-------------------------------------------------------------
```

Step 3: Obtain min, median, and max of age in each quintile

```
. table agegroupi, contents(min age median age max age)

-------------------------------------------
agegroupi |   min(age)    med(age)    max(age)
----------+--------------------------------
        0 |         30          35          37
        1 |         38          39          41
        2 |         42          44          47
        3 |         48          51          54
        4 |         55          58          65
-------------------------------------------
```

Step 4: Create grouped value of age with value equal to midpoint of quintile, **agequintile.** Check.

```
. generate agequintile=agegroupi
(1 missing value generated)

. recode agequintile (0=35) (1=39) (2=44) (3=51) (4=58)
(agequintile: 999 changes made)

. tab2 agegroupi agequintile
-> tabulation of agegroupi by agequintile

           |                      agequintile
 agegroupi |        35         39         44         51         58 |     Total
-----------+-------------------------------------------------------+----------
         0 |       200          0          0          0          0 |       200
         1 |         0        169          0          0          0 |       169
         2 |         0          0        212          0          0 |       212
         3 |         0          0          0        219          0 |       219
         4 |         0          0          0          0        199 |       199
-----------+-------------------------------------------------------+----------
     Total |       200        169        212        219        199 |       999
```

5.6 Date Variables

A date variable is a numeric variable
Stata stores a date by setting it to the number of days since January 1, 1960

Example: Stata stores January 3, 1960 with the value = 3
Example: Stata stores December 30, 1959 with the value = -2

The Stata command format provides display formats for our convenience

Format	Example of Display
%td	14oct1993
%tdDD.NN.CCYY	14.10.1993
%tdNN/DD/ CCYY	10/14/1993

Example
green-comment black-command blue-output

```
. * Generate a variable bday
. generate bday=12345


. * Create some copies for display using different formats
. generate bday2=bday
. format bday2 %td

. generate bday3=bday
. format bday3 %tdDD.NN.CCYY

. generate bday4=bday
. format bday4 %tdNN/DD/CCYY

. list, clear

     bday            bday2            bday3           bday4
1.   12345        19oct1993       19.10.1993      10/19/1993
```

Date Variable Creation

	Stata Command
mdy(, ,) Use the function **mdy(, ,)** to generate a Stata date variable from separate source variables for **month, day,** and **year**	**. generate** ***datevariable*** **= mdy(*monthvariable, dayvariable, yearvariable)*** **. format** ***newvariable*** **%td** **Example** **. generate dob=mdy(birthmos, birthday, birthyr)** **. format dob %tdNN/DD/CCYY**
date (,) Use the function **date (, ,)** to generate a Stata date variable from a **string date variable**	. **. generate** ***datevariable*** **= date (*stringvariable, mask)*** **. format** ***newvariable*** **%td** **Example** **. generate dob=date (sdob, "DMY")** **. format dob %tdNN/DD/CCYY** In this example, the source variable sdob is a string variable with string values such as "14oct1993. The Stata **date( )** function "knows" how to read many input formats. The "DMY" tells Stata that the sequence of arguments is day followed by month followed by year.
day( ), month( ), and year( ) Use these three functions to ***extract*** separate variables for day, month, and year.	**. generate** ***dayvariable*** **= day(*datevariable)*** **. generate** ***monthvariable*** **= month(*datevariable)*** **. generate** ***yearvariable*** **= year(*datevariable)***

Calculation with Dates

Example

. generate numdays = date1 – date2

Example

. generate numyears = (date1 – date2)/365.25

5.6 Labeling Variables

Example

. label variable q2 "q2: Ever pregnant"

Example

. label variable age "age: Age (years)"

Example

. label variable mpg "mpg: Mileage (miles per gallon)"

The Stata command **label variable** is used to provide a description of your variable.

The variable label is enclosed in quotes.

. label variable *variable* "*variablelabel*"

Tips!

- Create labels as short as possible, provided they are still informative
- Include the variable name in the variable label.
- Include units of measurement in the label, if appropriate

5.7 Labeling Variable Value Codes

Example

. **label define sexf 1 "male" 2 "male"**
. **label values sex sexf**

Example

. **label define yesnof 0 "no" 1 "yes"**
. **label values q1-q15 yesnof**

Labeling variable value codes is a **two step** procedure. A third step is possible if you later decide that you want to modify or add new labels.

Step 1: Create a dictionary that defines correspondences between value and value label
Use the **label define** command to define the value codes. Think of this as a "dictionary". Each value of the variable is listed and is followed by its value code. If the value code has a blank, it must be enclosed in quotes. **Tip!** ***As a matter of habit, enclose all value codes in quotes***.

Step 2: Apply the correct dictionary for your particular variable
Use the command **label values** to associate the value codes with the variable. You can associate several variables to one set of value codes. For example, you might have several variables with possible values of yes or no.

Step 3: To modify or add new value labels: Use the command **label values** to associate the value codes with the variable. Follow this command with the option **modify**

Step 4: To see your variable value codings: Use the command **label list**

```
. label define dictionaryname value "valuelabel" value "valuelabel"
. label values variable variable variable dictionaryname
. label define dictionaryname value "valuelabel" value "valuelabel" , modify
.  label list
```

Tips!

- Create labels as short as possible, provided they are still informative
- Include the variable name in the variable label.
- Include units of measurement in the label, if appropriate
- **numlabel, add** to see both value and the value label
- **numlabel, remove** to not see value. You will only see the label.

6. Working with Observations

6.1 Numbering of Observations

Stata allows you to work with individual observations by maintaining an internal numbering (also called "subscripting") of the observations in your data set. For example, for a data set containing information on a variable called age:

age[1]	Value of age for 1st observation
age[_n]	Value of age for current observation
age[_N]	Value of age for last observation
age[_n-1]	Value of age for previous observation
age[_n+1]	Value of age for next observation
age[27]	Value of age for 27th observation

Sometimes, you will want to manipulate individual observations. Consider the following examples.

Description	Stata Commands
For observations 2 through the last (this is _N), set the value of age equal to the value in the data set for the 1st observation.	**. replace age = age[1]**
For the 28th observation, replace the old value of age (whatever that is) with a new value equal to 15 years.	**. replace age = 15 in 28**
Suppose you have multiple observations for each patient in your data set and that patients are identified by the variable **patid.** Suppose further that each observation represents its own hospital encounter. You want to create a new variable that is a count of the number of hospitalizations and name it **admit_tot**	**. sort patid** **. by patid: generate admit_tot = _N** **. label variable admit_tot "No. Admissions"**

Example – Working with Multiple Records Per Patient

Suppose you have a data set of hospital encounters in which the number of admissions varies from patient to patient. You would like to number the individual admissions and you would like to have the option of selecting particular admissions for subgroup analyses (eg – perhaps you want to do a subgroup analysis that considers only the first or last admission). In particular, suppose

Patid - Patient identification variable
Date_adm - Date of admission variable

Step 1
* Sort data first by patient identification and second by date of admission
. sort patid date_adm

Step 2
* Separately for each patient, create a new variable **admit_num** that indexes the current admission
. by patid: generate admit_num = _n
. label admit_num "Patient Admission Number"

Step 3
* Separately for each patient, create a new variable **admit_tot** that is equal to the total number of admssions
. by patid: generate admit_tot = _N
. label admit_tot "Patients Total Number Admissions"

To do a subgroup analysis utilizing ONLY the first admissions
There are 2 ways to do this: 1) within the analysis command itself; and 2) Using preserve and restore.
More on preserve and restore in Section 4.3

. * Method 1 – Within the analysis command itself (my personal preference)
. * ***analysiscommand* if *condition***
. tabulate gender if admit_num==1

. * Method 2 – Set aside full data, work with selection, and finally restore full data
. preserve *set aside (preserve) a copy of data in memory*
. keep if admit_num==1 *work with a selection (namely) only first admissions observations*
.... Analyses ....
. restore *retrieve (restore) the full data set copy of data in memory*

6.2 Selecting Observations With Keep and Drop

Before you Begin

Stata works with the data in memory ONLY, not with data that you have saved.
You must explicitly save any changes that you make to data in memory; otherwise they will be lost when you exit Stata.

Fortunately, upon exiting, Stata will let you know that you have not saved your changes and will prompt you to save your data

The **keep** and **drop** commands allow you to make temporary selections of observations and temporary selections of variables.

Consider the following examples.

Description	Stata Commands
To temporarily select subjects for whom sex=1	**. keep if sex==1** *note – the "equal sign" is 2 symbols = =* *or* **. drop if sex !=1** *or* **. use "/Users/cbigelow/Desktop/clinic.dta" if sex ==1**
To select the first 100 observations in the data set	**. keep in 1/100** *or* **. use "/Users/cbigelow/Desktop/clinic.dta" in 1/100**

Note – The KEEP and DROP commands can also be used to make temporary selections of variables.

6.3 Subgroup Analysis

Sometimes, you will want to execute a command or an analysis for a subset of your data set. There are multiple ways to do this, each with its own advantages.

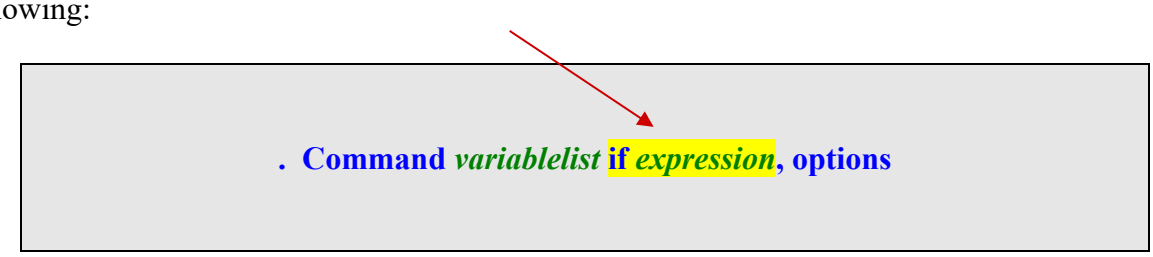
a. Restricted Command With If

Example -

•list state if (area==4) and (lung >=10)

This command instructs Stata to print the values of the variable state for the subset of the data set for which the value of area=4 and the value of lung is greater than or equal to 10. **Reminder!** ***Note the use of the two equal signs in the restriction of the values of the variable area.***

In section 3.3, the basic structure of a Stata command was introduced. See page 18. Within the syntax of a Stata command is the possibility of restricting its execution according to an "***if expression***". Recall the following:



b. Stratified Analysis With By

Example –
Suppose it is of interest to examine the mileage of automobiles separately for foreign and domestic cars.
. sort foreign
. by foreign: summarize mpg

Example -
.bysort foreign: summarize mpg

In both examples, Stata sorts the data by value of the variable foreign. Summary statistics of the variable mpg are produced separately for subgroups defined by foreign.

A stratified analysis using the Stata command **by** is recommended when you want to repeat the same command or analysis for every group defined by the values of some stratifying variable.

Method 1
. sort *stratifyingvariable*
. by *stratifying variable*: command, options

Method 2
. bysort *stratifying variable*: command, options

c Subgroup Analysis With Preserve and Restore (I don't actually like this approach but if you do…)

Tip – If you want to execute ONLY ONE command for just a subgroup of interest, I encourage you to do this by embedding the condition in your analysis command as illustrated on page 52 and duplicated here.

```
. * analysiscommand if condition
. tabulate gender if admit_num==1
```

But if want to execute SEVERAL commands or analyses for a particular subgroup of interest, then you might want to make use of the **preserve** and **restore** commands. They allow you to make temporary selections of observations and temporary selections of variables.

.preserve - Instructs Stata to set aside, for later, a copy of the current data that is in memory. This command has NO effect on data saved elsewhere on disk

.restore - Instructs Stata to retrieve the data that has been set aside

```
. * Example
. preserve                  set aside (preserve) a copy of data in memory
. keep if admit_num==1      work with a selection (namely) only first admissions observations
      .... Analyses ....
. restore                   retrieve (restore) the full data set copy of data in memory
```

Method Using *keep if*
. preserve
. keep if *expression*
. … *analyses* …
. clear
.restore

Method Using *drop if*
. preserve
. drop if *expression*
. … *analyses* …
. clear
.restore

Example – Working with a Portion of a Huge Data Set

Suppose you have a huge data set called **clinic.dta** that is stored on your desktop and that has lots of variables and lots of observations. You would like to work with just a few variables (eg – **height** and **weight**) and you would like to consider a subgroup only (**admit_num=1**). The following commands will allow you to do this "temporary" selection:

```
. preserve
. keep if admit_num==1        Note the use of two equal signs in the KEEP statement!
. … analyses …
. clear
.restore
```

REMINDER!

Stata works with the data in memory ONLY, not with data that you have saved.
You must explicitly save any changes that you make to data in memory; otherwise they will be lost when you exit Stata.

Fortunately, upon exiting, Stata will let you know that you have not saved your changes and will prompt you to save your data

6.4 Random Sampling From a Data Set Using Sample

Before you Begin

Stata works with the data in memory ONLY, not with data that you have saved.
You must explicitly save any changes that you make to data in memory; otherwise they will be lost when you exit Stata.

Fortunately, upon exiting, Stata will let you know that you have not saved your changes and will prompt you to save your data

The Stata command **sample** produces a ***simple random sample*** from a data set. This can be done as a percent sample or as a simple random sample of a specific count. Use the **count** option to obtain the latter.

Consider the following examples.

Description	Stata Commands
To obtain a 10% simple random sample	**. sample 10**
To obtain a simple random sample of exactly 93 observations	**. sample 93, count**

6.5 For Repeated Measures: Wide to Long Using Reshape Long

When analyzing repeated measures, the layout of the data is often in what is called "wide" format. In this format, the repeated measurements are distinct variables across the horizontal of each record, as in the following example taken from UCLA Academic Technology Services. A long layout is needed for some kinds of repeated measures analyses and for many of the graphs that you will want to produce.

Example -

Consider the family income data file below.

```
use http://www.ats.ucla.edu/stat/stata/modules/faminc, clear

list
```

	famid	faminc96	faminc97	faminc98
1.	3	75000	76000	77000
2.	1	40000	40500	41000
3.	2	45000	45400	45800

Key to Using reshape long

- Be sure you have an **id variable** (eg – **famid**)
- Name your **repeated measures variables** using the convention name##
  as in the example above (eg – **faminc96 faminc97 faminc98**)
- Choose a new variable name to index the **occasion** of the repeated measure (eg – **year**)

```
. reshape long variableprefix, i(idvariable) j(occasionvariable)
```

Example - continued

```
reshape long faminc, i(famid) j(year)

(note:  j = 96 97 98)

Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                        3   ->      9
Number of variables                   4   ->      3
j variable (3 values)                     ->   year
xij variables:
             faminc96 faminc97 faminc98   ->   faminc
-----------------------------------------------------------------------------
```

The **list** command shows that the data are now in **long** form, where each **year** is represented as its own observation.

```
list

        famid       year     faminc
  1.        1         96      40000
  2.        1         97      40500
  3.        1         98      41000
  4.        2         96      45000
  5.        2         97      45400
  6.        2         98      45800
  7.        3         96      75000
  8.        3         97      76000
  9.        3         98      77000
```

Let's look at the **wide** format and contrast it with the **long** format.

6.6 For Repeated Measures: Long to Wide Using Reshape Wide

The wide format layout of repeated measures data is useful when doing multivariate analyses. Thus, you may want to reshape from long to wide. Consider the following example. The unit of measurement is the family and the id for family is the variable **famid**. The variable indexing the repeated measurements within each family is birth. The analysis variable is age.

Example -

	famid	birth	age
1.	1	1	9
2.	1	2	6
3.	1	3	3
4.	2	1	8
5.	2	2	6
6.	2	3	2
7.	3	1	6
8.	3	2	4
9.	3	3	2

Key to Using reshape wide

- Be sure you have an **id variable** (eg – **famid**)
- Be sure you have your **repeated measures variable** (eg –**birth**).
  This exists in the long format data set but not in the wide format data set
- Be sure you have your outcome variable (eg –**age**).
  This exists as just one variable in your long format data set but will exist as multiple (repeated measures) variables in your wide format data set
  (eg – **age1 age2 age3** etc).

. reshape wide *variable*, i(*idvariable*) j(*occasionvariable*)

Example - continued

```
. reshape wide age, i(famid)  j(birth)

(note:  j = 1 2 3)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                        9   ->       3
Number of variables                   3   ->       4
j variable (3 values)             birth   ->   (dropped)
xij variables:
                                    age   ->   age1 age2 age3
-----------------------------------------------------------------------------

. list

          famid       age1       age2       age3
  1.          1          9          6          3
  2.          2          8          6          2
  3.          3          6          4          2
```

Let's look at the pieces of the **reshape** command.

```
. reshape wide age, j(birth) i(famid)
```

wide tells reshape that we want to go from long to wide
age tells Stata that the variable to be converted from long to wide is **age**
i(famid) tells reshape that **famid** uniquely identifies observations in the wide form
j(birth) tells reshape that the suffix of **age** (1 2 3) should be taken from the variable **birth**

It is also possible to convert from long to wide repeated measurements on several variables.
In the example below, the data includes repeated measurements on each of **age, wt,** and **sex** in long format. In wide format, these become **age1 age2 age3 wt1 wt2 wt3 sex1 sex2** and **sex3**

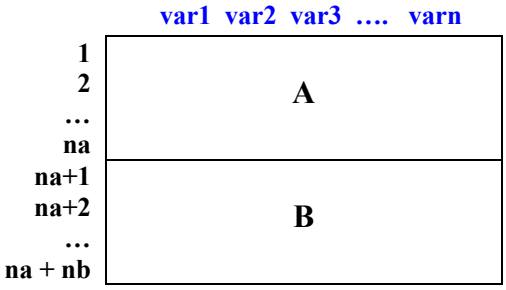
```
reshape wide kidname age wt sex, i(famid) j(birth)

 (note:  j = 1 2 3)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                        9   ->      3
Number of variables                   6   ->     13
j variable (3 values)             birth   ->   (dropped)
xij variables:
                                kidname   ->   kidname1 kidname2 kidname3
                                    age   ->   age1 age2 age3
                                     wt   ->   wt1 wt2 wt3
                                    sex   ->   sex1 sex2 sex3
-----------------------------------------------------------------------------
```

7. Working with Data Sets

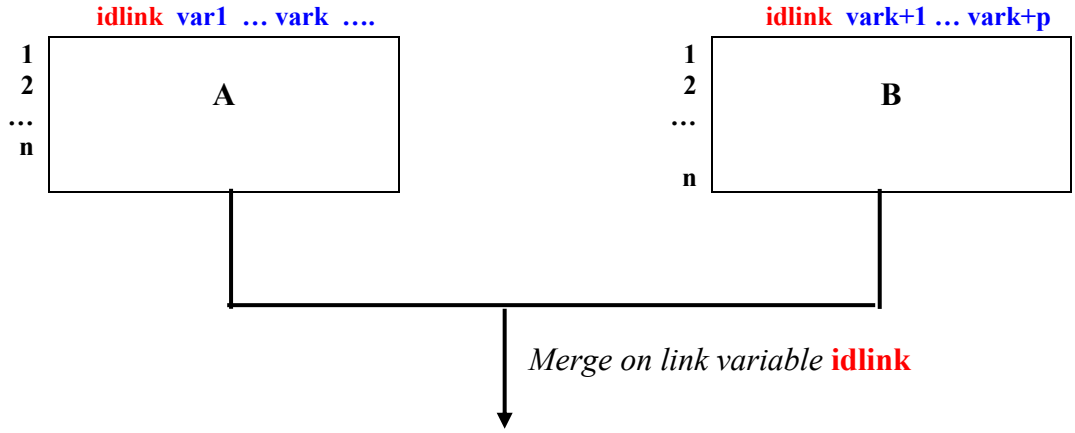
Append or Merge?

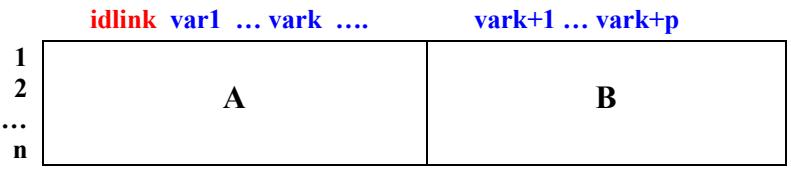
If you want …	Then use
More subjects **More observations** of the same variables, this is a concatenation	**APPEND**
More variables **More data** on the same subjects, this is a merge	**MERGE**

Schematic of Append (sample size increases from na to (na + nb), number of variables is the same)



Schematic of Merge (sample size is the same, number of variables increases from k to (k+p)





Design ..... Data Collection ..... Data Management ..... Data Summarization ..... Statistical Analysis ..... Reporting

7.1 Concatenating Data Sets Using Append

How to Concatenate Two Data Sets Using Append

Data sets ***dataA.dta*** and ***dataB.dta*** have the same variables and different observations. They are combined into a new data set called ***combined.dta***

```
. use dataA.dta
. append using dataB.dta
. save combined.dta
```

7.2 Merging Data Sets Using Merge

How to Merge Two Data Sets Using Merge

Data sets ***dataA.dta*** and ***dataB.dta*** have the different variables on the same subjects. To merge them requires linking, for each subject id, the information in dataA.dta with the information in dataB.dta.

Key to Merging

- The data sets to be merged MUST all contain at least one "linking" variable that has the same name in each data set. And, it must uniquely identify each subject id.
- The other variables should be different in the two data sets.
- You **MUST** sort each data set by the "linking" variable prior to their merge.

Stata Creates a Useful Variable For You Called _merge

You can use this to check for unmatched observations.

_merge = 1 if record exists in data set 1 ONLY
2 if record exists in data set 2 ONLY
3 if record exists in both data set 1 and 2

```
. use dataA.dta
. sort idlink
. merge idlink using dataB.dta, sort
. save combined.dta
```

How to Check for Unmatched Observations After A Merge.

Use the **assert** command to flag the unmatched observations. The command assert instructs Stata to report any instances where the claim is false.	**. assert _merge==3**
Obtain a frequency distribution of the values of _merge	**. tabulate _merge**
Obtain a listing of the id's for which there are not matches	**. list *idlink* if _merge < 3** *Or* **. list *idlink* if _merge ==1** **. list *idlink* if _merge ==2**

8. Data Screening and Documentation

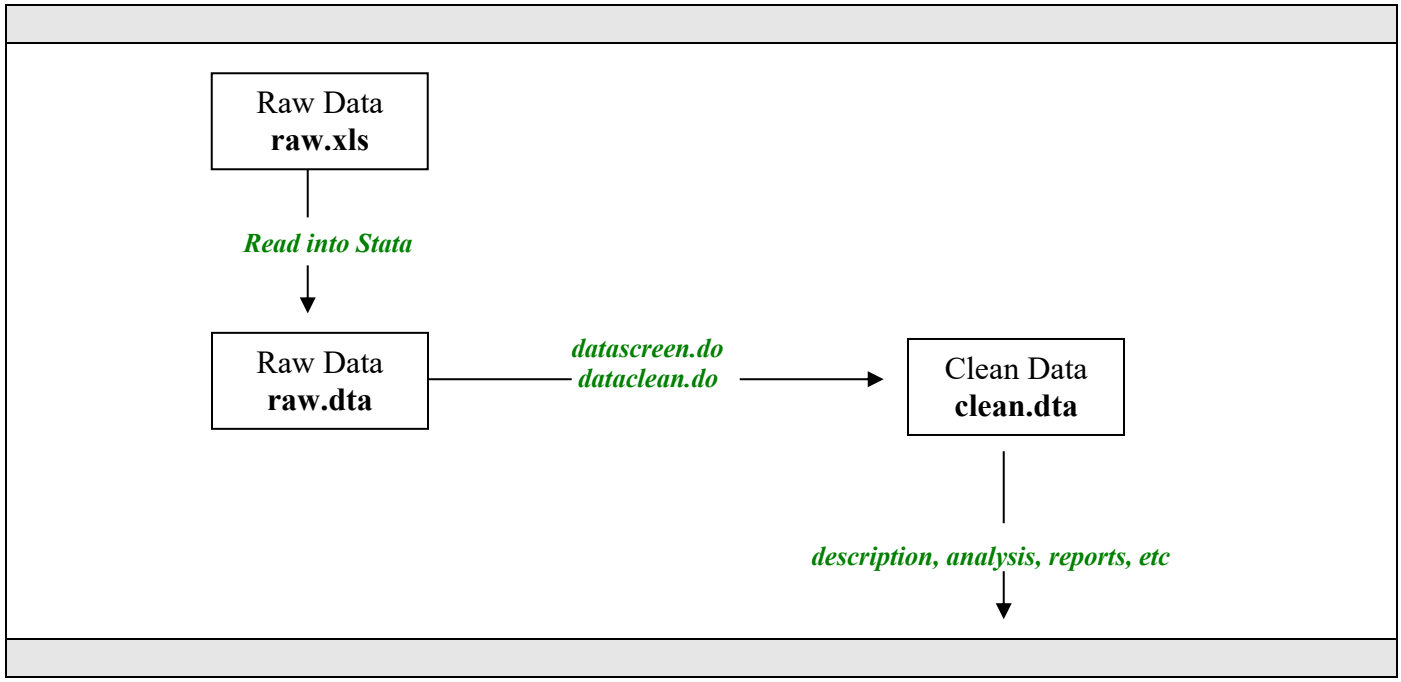
Important Tip!

Never change the raw data!

Data screening, cleaning and **"readying for analysis"** are the first things you do with a data set.

- The raw data file is preserved unchanged and a copy is placed elsewhere for safekeeping.
- The cleaned and "readied" data set is a new file with its own name
- **NEVER** make changes to the raw data.

Schematic!



Design ..... **Data Collection** ..... **Data Management** ..... **Data Summarization** ..... **Statistical Analysis** ..... **Reporting**

Data can be "not clean" for several reasons

- Incompleteness
- Errors in data entry
- Outliers
- Duplicates
- "string" type instead of "numeric" type

Data Review Suggestions

- Review the data codebook to check on data type
- List the data and do range checks
- Identify duplicates
- Screen for Errors
- For discrete variables – produce tabulations
- For continuous variables – obtain selected descriptive statistics
- For continuous variables – produce dot plot or histogram summary

Data Documentation Suggestions

- Label your data set
- Label variables
- Label variable values

8.1 Data Screening

(a) Review the data codebook

	Stata Command
describe This command provides overall information about the dataset including – data set name, creation date, number of variables, samplesize, storage type for each variable, etc	**Example** . **describe**
codebook This command will produce basic information about every variable in your data set: type, variable label, variable value labels. It will also produce a limited set of descriptive statistics that are helpful – n, range, mean, sd, **codebook, compact** This option compact will provide a more limited set of information.	**Examples** . **codebook** . **codebook, compact** . **codebook *matage hyp***

(b) List the data and do range checks

	Stata Command
list This command will produce a listing of every variable value for every subject in the dataset **list *variable1 variable2*** This command will produce a listing of the values of the two variables (variable1 and variable 2) for every subject in the dataset **list *variable1 variable2* in 1/5** This command will produce a listing of the values of the two variables (variable1 and variable 2) for the first 5 subjects in the dataset **list *variable1 variable2* in -5/l** This command will produce a listing of the values of the two variables (variable1 and variable 2) for the last 5 subjects in the dataset **Note – The character after the slash is the letter "el" not the number 1.**	**Examples** . **list** . **list *matage hyp*** . **list *matage hyp* in *1/5*** . **list *matage hyp* in *-5/l*** **This is the letter "el"**
How to do a range check **Discrete** **tabulate *variable*, missing** This command will produce a tabulation of the frequencies of each value. Be sure to include the "missing" option **Continuous (suppose your n=100)** **sort *variable*** **list *variable* in 1/5** **list *variable* in 96/100** These commands will produce a listing of the smallest 5 values and the largest 5 values.	**Example** . **tabulate *hyp*, missing** **Example** . **sort *matage*** . **list *matage* in 1/5** . **list *matage* in 96/100**

(c) Identify Duplicates

There are 3 kinds of duplicates. ***BEWARE*** …depending on your study, these may or may not be errors.

(i) Duplication of study id and duplication of all data *(eg – same record entered twice by mistake)*

(ii) Duplication of study id only; data varies *Tip! This is ONLY an error if there should be just one record/id!*

(iii) Duplication of all data; study id varies *Tip! This is NOT necessarily an error*

(i) – Duplication of study id and all data *(note – these suggestions are in the recommended order)*

Stata Command	Example
duplicates report This two word command will give you a summary report of the number of instances of duplicate observations on every variable	Copies observations surplus 1 24 0 2 4 2 3 3 2 **Translation:** **1st row: In this data set, there are 24 unique observations (1 copy of each).** **2nd row: There 4/2=2 groups of duplicates of size 2. Each group of 2 duplicates contributes 1 surplus. So the total number in surplus in this row is 2.** **3rd row: Finally there is 3/3=1 group of duplicates of size 3. This group of 3 duplicates contributes 2 surplus.** **Thus, the total number of surplus in this row is also 2.**
duplicates report *variable* This two word command will give you a summary report of the number of instances of duplicate observations of the specified variable ***Tip!*** Use this to search for duplicate study id.	**Example** **. duplicates report *studyid***
duplicates tag, generate(*variable*) Very handy … This two word command followed by the option "generate(*variable*)" will create a new variable called "*variable*" that will indicate for every observation the number of duplications.	**Example** **. duplicates tag, generate(*dup*)** **Trnaslation:** **Unique observations will have dup=0** **Observations appearing twice in the data set will have dup=1. Observations appearing 3x in the data set will have dup=2 … and so on.**
sort *variable variable* **list *variable variable* if *dup* > 0, sepby(*variable*)** These two steps accomplish, first, a grouping together of the observations in each group of duplicates. It then provides a listing of the requested variables for ***only*** those observations that are non-unique.	**Example** **. duplicates tag, generate(*dup*)** **. sort *studyid age gender*** **. list *studyid age gender* if *dup>0*, sepby*(studyid)*** **Note - The option "sepby" will produce nice horizontal lines that separate each group of duplicates.**

(ii) – Duplication of study id only; data varies *(again – suggestions are in the recommended order)*

Stata Command	Example
duplicates tag *studyidvariable*, generate(*variable*) This tags duplications based on study id regardless of values of any other data	**Example** **. duplicates tag *studyid*, generate(*id_dup*)**
duplicates tag, generate(*variable*) This identifies occurrences of the duplication of everything – study id AND all the data.	**Example** **. duplicates tag, generate(*all_dup*)**
Count if *id_dup== 1 & all_dup==0* This produces a count of the number of instances where study id is duplicated but data are different	**Note** **I supplied the variable names *id_dup* and *all_dup*** **You can choose your own variable names.** ***REMINDER*** ***This kind of duplication is NOT an error when there are actually multiple records of data for each id.***
sort *studyid* **list *studyid variable* if *id_dup==1 & all_dup==0***	**Note** **I supplied the variable names *id_dup* and *all_dup*** **You can choose your own variable names.**

(iii) – Duplication of data only; study id varies

There is not a straightforward solution for identifying these kinds of duplicates in Stata.

(d) Screen for Errors

Stata Command	Example
Error – Incorrect specification of type **describe, fullnames detail** Use this command with the two options "fullnames" and "detail" to obtain complete information on each variable including its width and type	
Error – **Discrete** variable has an "out of range value **assert inlist(*variablename, value, value value*)** Stata will report to you the number of occrrences of the stated assertion being false. Translation: You provide what the correct rule is; Stata then tells you how many errors you have. **list *variable variable* if ! *inlist*(*variablename, value, value value*)**	**Example** **. assert inlist(*race, 1, 2, 3*)** To be clear: The values 1, 2, 3 are the allowable values. The assert command then prompts Stata to tell you how many are INCORRECT **Example** **. list *studyid race* if ! inlist(*race, 1, 2, 3*)** To be clear: With this command, Stata will list the actual observations that are INCORRECT.
Error – **Discrete** variable has an "out of range value **assert *variablename truexpression*** Stata will report to you the number of occrrences of the stated assertion being false. **list *variable variable* if *variablename falseexpression***	**Example** **. assert *bp >=100 & bp <= 200*** **Example** **. list *studyid bp if bp < 100 | bp > 200***

(e) Screen Using Tabulations (discrete variables)

	Stata Command
tabulate *variable*, missing This command will produce a tabulation of the frequencies of each value. Be sure to include the "missing" option	**Example** . **tabulate *hyp*, missing**
bysort *stratavariable*: tabulate *variable*, missing This command will produce a tabulation of the frequencies of each value of ***variable*** separately across strata defined by ***stratavariable***.	**Examples** . **bysort *age*: tabulate *hyp*, missing**
tabulate *variable* if *variable2==expression*, missing This command will produce a tabulation of the frequencies of variable only for those observations for which the variable2 expression is true	**Example** . **tabulate *hyp* if *sex==2*, missing**

(f) Screen Using Histograms (continuous variables)

	Stata Command
inspect *variable* This is a nice data screening command! IT will produce some descriptive statistics plus a little histogram	**Example** . **inspect *matage***

8.2 Data Documentation

(a) Label Dataset

	Stata Command
label data "*labelstring* **"** This command will attach a label to your data set	**Example** . **label data** ***"statins analysis file 2018"***
To attach a note to a dataset **note:** ***little note*** To list all notes **notes**	**Example** . **note: Created from premier download of 6-1-2018** **Example** . **notes**

(b)Label variables *Previously detailed on p 50*

	Stata Command
label variable ***variable "variablelabel "***	**Example** . **label variable** ***mpg "mpg: miles per gallon"***

(c) Label variable values *Previously detailed on p 51*

	Stata Command
Step 1 - Define the dictionary **label define** ***dictionaryname value "valuelabel" value "valuelabel"*** Step 2 – Link the dictionary to the variable **label values** ***variablename variablename dictionaryname***	**Example** . **label define** ***hypf 0 "normotensive" 1 "hypertensive"*** . **label values** ***hyp hypf*** **Example** . **label define** ***yesnof 0 "no" 1 "yes"*** . **label values** ***exposure treatment mortality yesnof***

(d) Annotate with notes

Stata command	Example
DATA SET *How to record a note* **notes:** Use the command **notes** *followed by a colon* to record notations about your data set that you want to retain. *DATA SET* *How to read/display notes* **notes** Use the command **notes** without *a colon* to read your notes	**. notes: corrections made by cb in p01**
VARIABLE *How to record a note* **notes *variablename*:** Use the command **notes** *followed by a colon* to record notations about a particular variable that you want to retain. *VARIABLE* *How to read/display notes* **notes *variablename*:** Use the command **notes** without *a colon* to read your notes	**. notes *age*: age>120 and age < 0 are set to missing**

9. Introduction to DO Files

What is a DO File?

A **do file** is a "batch" file that contains a set of commands that can be re-executed

- Do files have the extension **".do"** **(Example: bigelow_hw7.do)**
- A **do file** can be created within Stata and saved for use in a later session.
- Or, a **do file** can be opened within Stata and then executed.

Tip! Do Files have many advantages. Consider the following uses of do files:

1. **Record keeping**: Consider creating a do file of your work at the end of your Stata session.
2. **"How to"**: Consider using someone else's do file to perform an analysis of interest.
3. **Time saving/Best Practices**: There are many data management and analysis tasks that are done repeatedly. Consider creating "standard" do-files to do standard things such as data set creation, cleaning, descriptive statistics, graphs, etc.

There are at least 3 Ways to Create a DO File

1. *** **(Best choice!)**
   Record Keeping: At the end of your Stata session, as a record of your work **(this is what I do most often)**

2. As an edit of someone else's do file

3. From Scratch

Method 1 – "Record Keeping"
Create a do file record of your Stata session

Step 1: **Launch Stata. A good practice is to begin your sessions with some comments that are useful for record keeping**

Example - The following is a suggestion for preliminary comments. You might have a better system

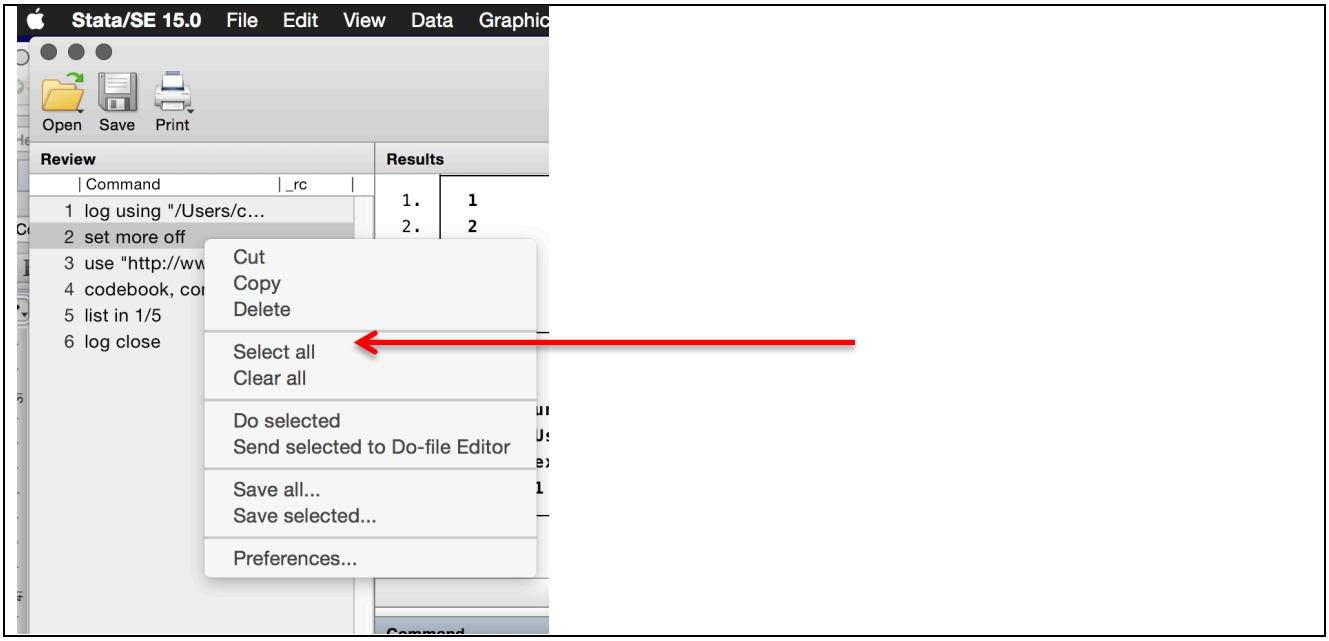
```
. log using "users/cbigelow/Desktop/october1p01.log", replace
. *  BIOSTATS 690C -  Fall 2020
. *  prog:  Carol Bigelow
. *  date:  September 16, 2020
. *  input:  http://www.pauldickman.com/survival/ivf.dta
. *  title:  Illustration of creating a DO FILE record of session
```

Step 2: **Do your Stata session.**

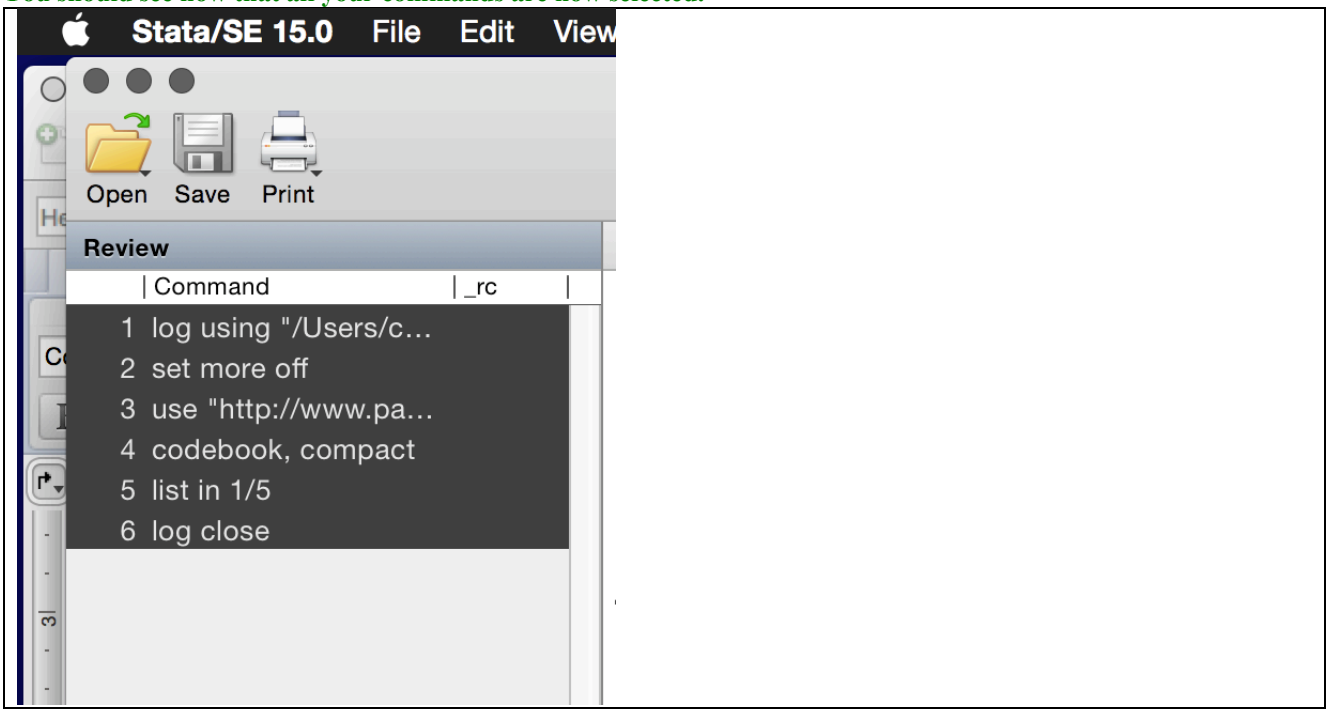
Example:

```
. set more off
. use "http://www.pauldickman.com/survival/ivf.dta", clear
(In Vitro Fertilization data)

. codebook, compact

Variable   Obs Unique      Mean    Min    Max  Label
------------------------------------------------------------------------------
id         641    641       321      1    641  identity number
matage     641     21  33.97192     23     43  maternal age (years)
hyp        639      2  .1392801      0      1  hypertension (1=yes, 0=no)
gestwks    641    177  38.68725  24.69  42.35  gestational age (weeks)
sex        641      2   1.49142      1      2  sex of the baby
bweight    641    295  3129.137    630   4650  birthweight (g)
------------------------------------------------------------------------------
. list in 1/5

     +-------------------------------------------------+
     | id   matage   hyp   gestwks      sex   bweight |
     |-------------------------------------------------|
  1. |  1       33     0     37.74   female      2410 |
  2. |  2       34     0     39.15   female      2977 |
  3. |  3       34     0     35.72   female      2100 |
  4. |  4       30     0     39.29     male      3270 |
  5. |  5       35     0     38.38   female      2620 |
     +-------------------------------------------------+

. log close
      name:  <unnamed>
       log:  /Users/cbigelow/Desktop/October1p01.log
```

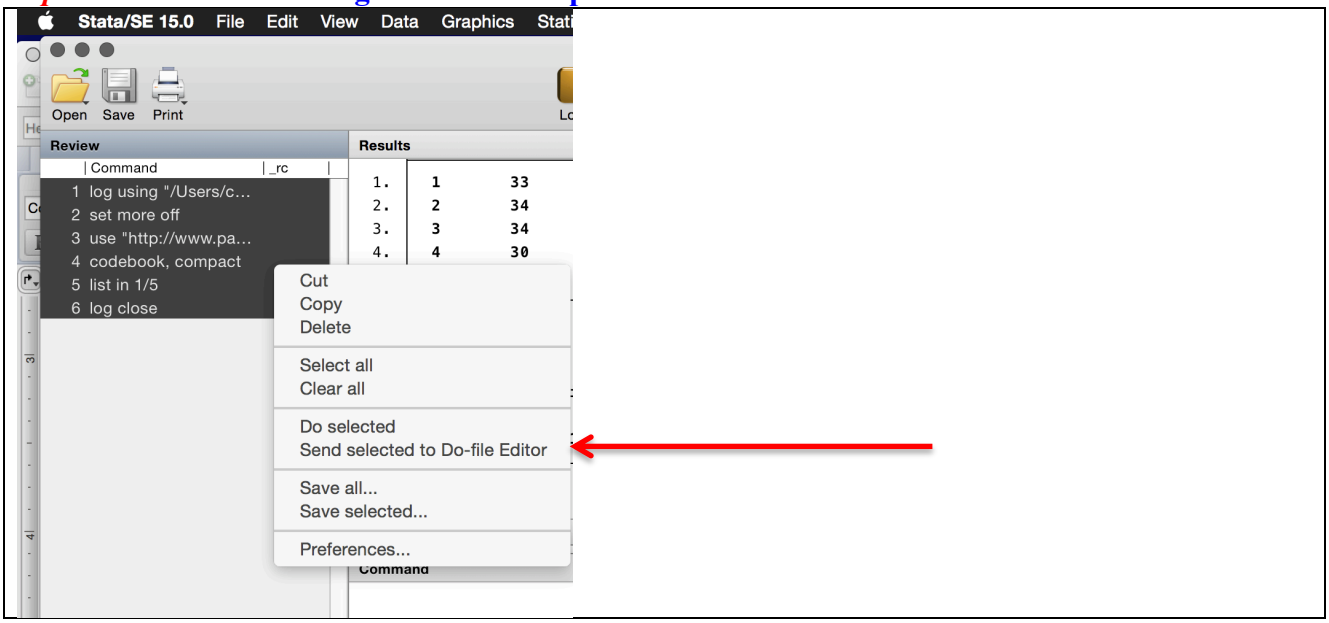
Step 3: **In the HISTORY window, RIGHT CLICK anywhere. From dropdown: SELECT ALL**
Note – If you are using an older version of Stata this will be the REVIEW window, such as my older version here…



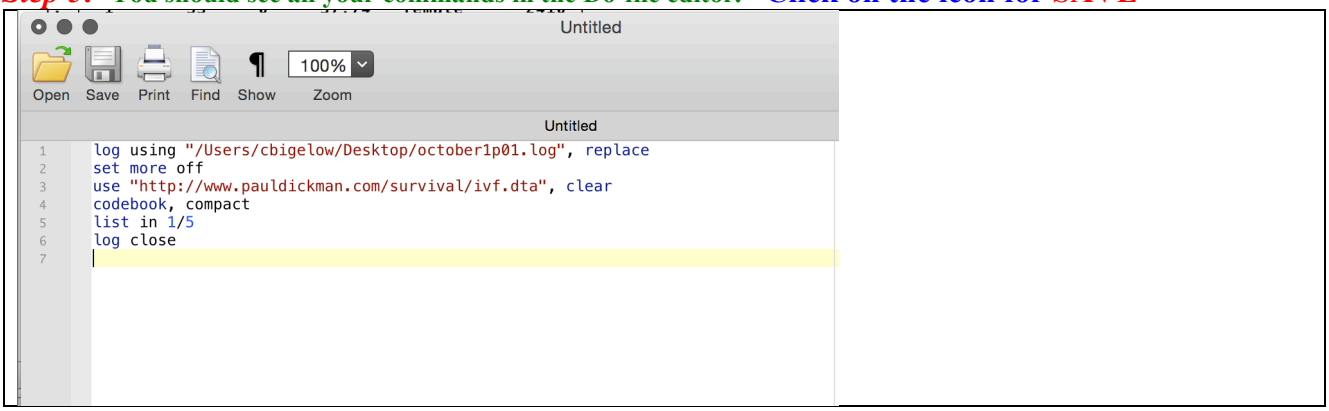
You should see now that all your commands are now selected.



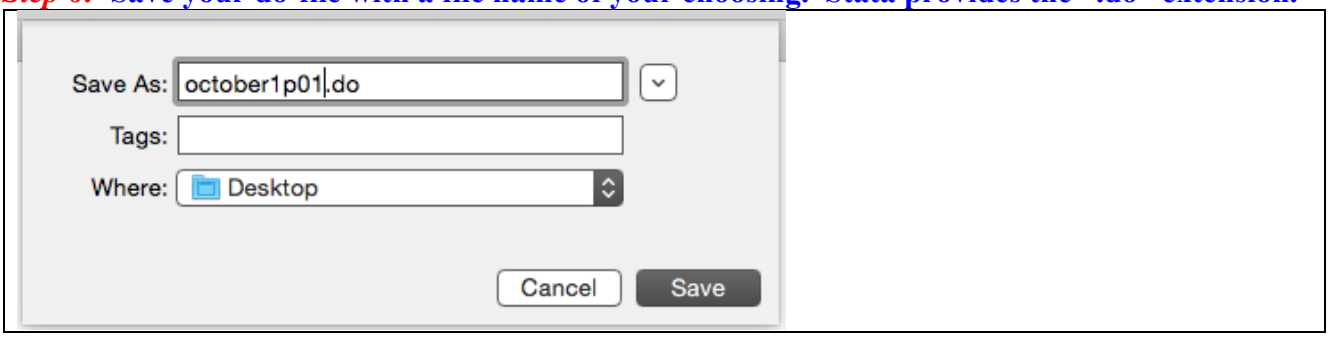
Step 4: **RIGHT CLICK again. From dropdown: SEND to Do-file Editor**



Step 5: **You should see all your commands in the Do-file editor. Click on the icon for SAVE**



Step 6: **Save your do-file with a file name of your choosing. Stata provides the ".do" extension.**



Method 2 – "As an Edit"

Create a do file from someone else's do file

Tip! ***This is helpful when you are learning something new!***

1. Have ready (translation: know the location) the DO file you want to use.
2. **Launch Stata**
3. From the main menu at top, at right: Launch the **DO FILE EDITOR**.
4. From the main menu at top, at left: OPEN
5. Browse to locate and open the DO file you want to use.
6. IMPORTANT: SAVE under a new name of your choosing.
7. Edit, fix, execute commands as needed
8. **SAVE** under the new name before exiting Stata
9. **Exit Stata**

Method 3 – "From Scratch"

In general, I do not recommend this approach unless you are very expert at Stata

From Scratch, Within STATA

1. Click on the do-file editor **– I don't recommend this, actually**
2. Type in a set of commands
3. Save the do-file with a name. Stata will provide the extension ".do"

From Scratch, Using a Text Editor

1. Create your do-file using a text editor of your choice.
2. When naming your do-file, use the extension ".do"
3. When saving your do-file, save it to your working directory.
4. Launch Stata.
5. At the command window, type **.do** ***dofilename*****.do**

Suggestion – Time Saving/Best Practices
Develop your own standard do file Stata "batch" programs

Step 1: **shell.do**

Create and save a boiler DO file that will be the starting file for all future DO files.

Example - I named my boiler do file *shell.do*. You can name yours whatever you want.

Note – Note the use of // in the syntax. These are comments. The // work ONLY in a do file.

Dear class – User edits the highlighted yellow.
The code highlighted in yellow is specific to my computer and my session.
You would need to change this code according to your computer and your Stata session.

```
***   shell.do

***   Opening Commands (do not edit)
version 16                  //  Ensures future Stata will execute
capture clear               //  Clear data space
capture clear matrix        //  Clear data matrix space
set more off                //  Turn off screen pausing of output
capture log close           //  Close any open logs

***   Opening Commands (User Edits)
log using "users/cbigelow/Desktop/carol_p01.log", replace
cd/users/cbigelow/Desktop
use "/users/cbigelow/Desktop/Sepsis.dta"

***  Opening Commands (User Edits)
* do file:  descriptives.do
*  author: carol bigelow
*    date: October 1, 2019
*   input: sepsis.dta
*  output: none
* summary: Descriptives for general use

***  Analyses
USER SUPPLIES

***   Closing Commands (do not edit)
log close
exit
```

Step 2: **Perform your Stata work**

Now that you have shell.do, you can make use of it to create log files and do files of every STATA session. Of course, you won't always want to go to such lengths. But you will if you are doing analyses that will ultimately be the source of tables and figures for any sort of publication. Here is how to proceed.

#1. **Launch STATA**

#2. From the main menu at top, at right: Launch **Do-file Editor**

#3. From the main menu at top, at left: **OPEN**
Browse to OPEN shell.do (or whatever your shell DO file is)

#4. From the main menu at top, at left: **FILE > SAVE AS**
Save under a new name of your choosing.
Take care to notice the path/directory where your new file is being saved.

#5. EXECUTE the opening commands. To do this.
Use your cursor to highlight the commands to be executed
From the main menu at top, at right: click on the icon **DO**

#6. TYPE and EXECUTE new commands as you like.

#7. FIX and RE-EXECUTE all the commands that do not work. Invariably, you will execute some commands that are incorrect. You might also re-run some commands with changes that you like.

#8. CLEAN UP your new DO file by deleting all the commands that did not work and all the commands that you decide you do not want to keep.

#9. Final Save
From the main menu at top, at left: **FILE > SAVE AS**
Save under the same new name that you used in #4 above.
Again, take care to notice the path/directory where your new file is being saved.

Always embed a log of your session in your .do file