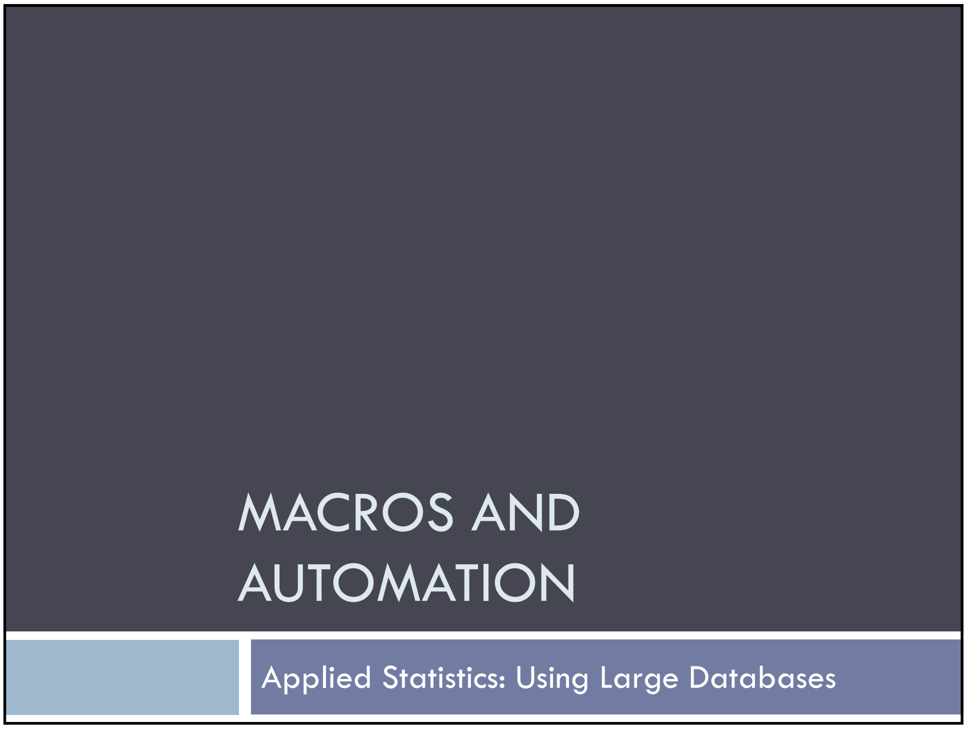
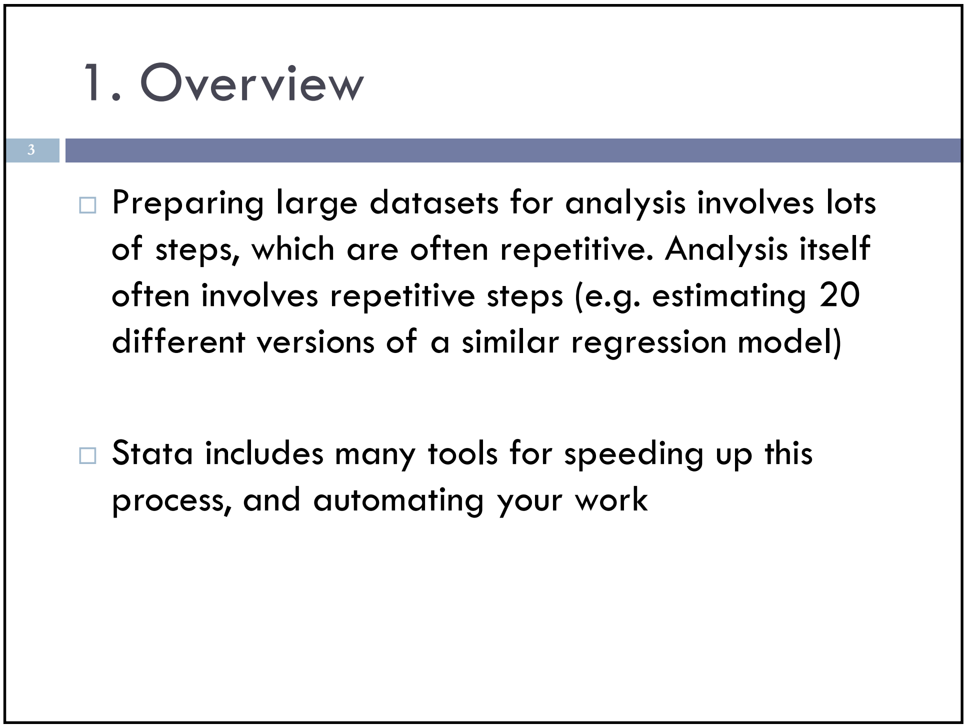


MACROS AND AUTOMATION

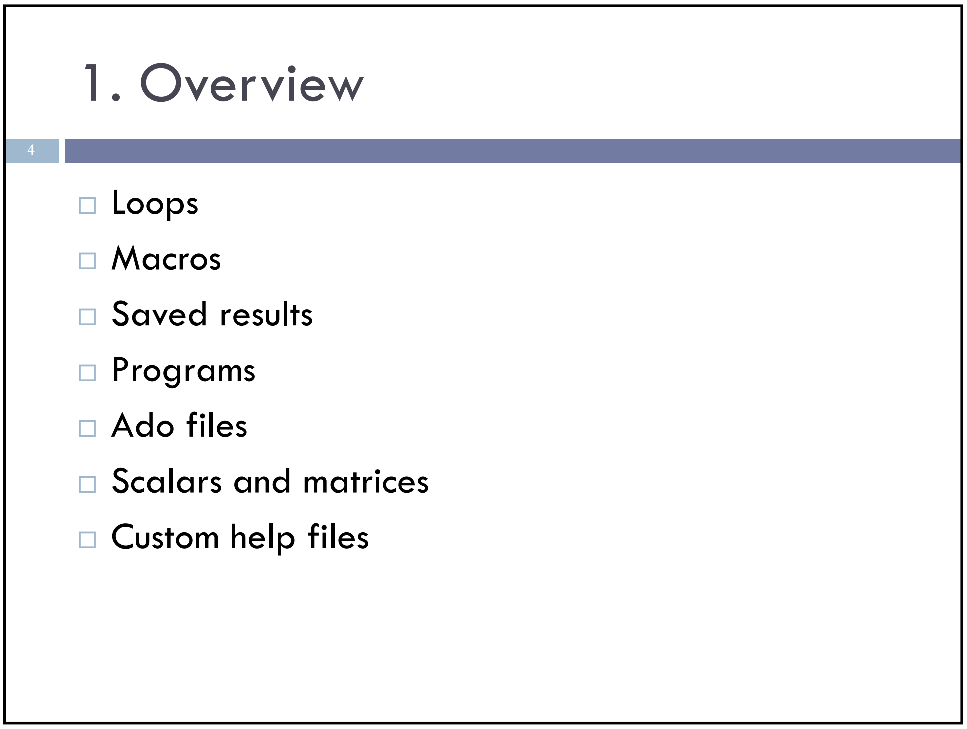
Applied Statistics: Using Large Databases



1. Overview

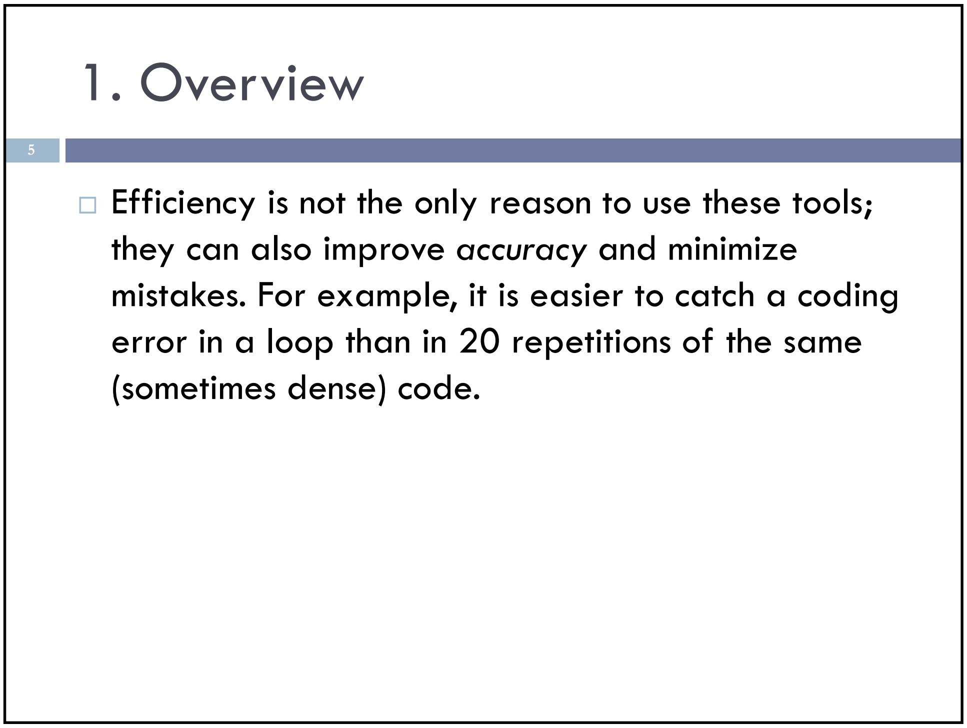
- Preparing large datasets for analysis involves lots of steps, which are often repetitive. Analysis itself often involves repetitive steps (e.g. estimating 20 different versions of a similar regression model)

- Stata includes many tools for speeding up this process, and automating your work



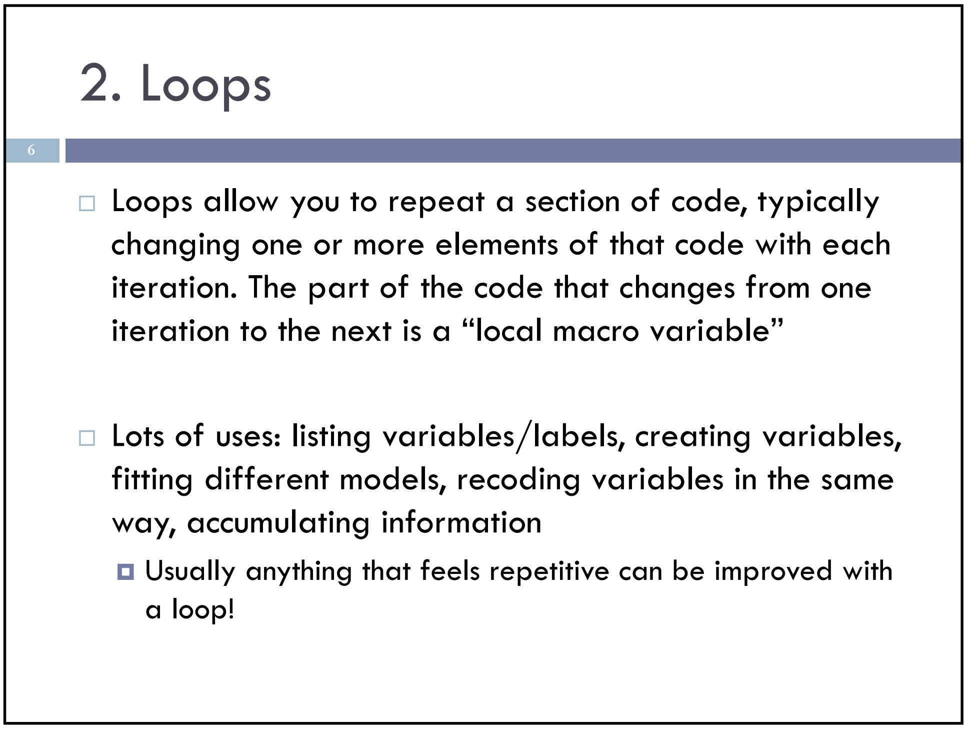
1. Overview

- □ Loops
- □ Macros
- □ Saved results
- □ Programs
- □ Ado files
- □ Scalars and matrices
- □ Custom help files



1. Overview

- □ Efficiency is not the only reason to use these tools; they can also improve *accuracy* and minimize mistakes. For example, it is easier to catch a coding error in a loop than in 20 repetitions of the same (sometimes dense) code.

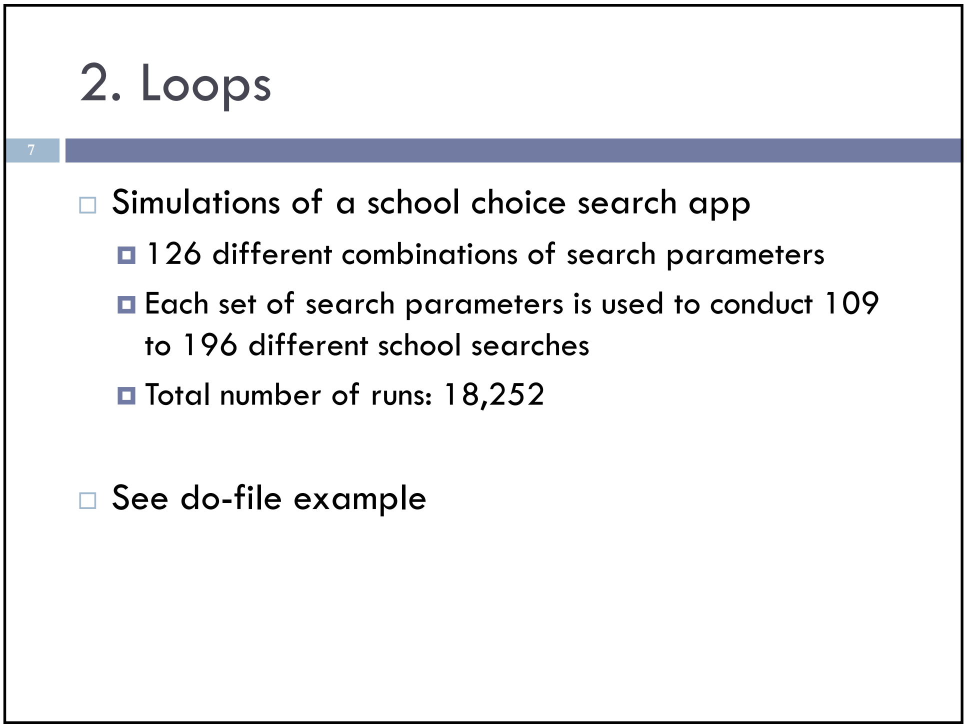


2. Loops

6

- Loops allow you to repeat a section of code, typically changing one or more elements of that code with each iteration. The part of the code that changes from one iteration to the next is a “local macro variable”

- Lots of uses: listing variables/labels, creating variables, fitting different models, recoding variables in the same way, accumulating information
  - Usually anything that feels repetitive can be improved with a loop!

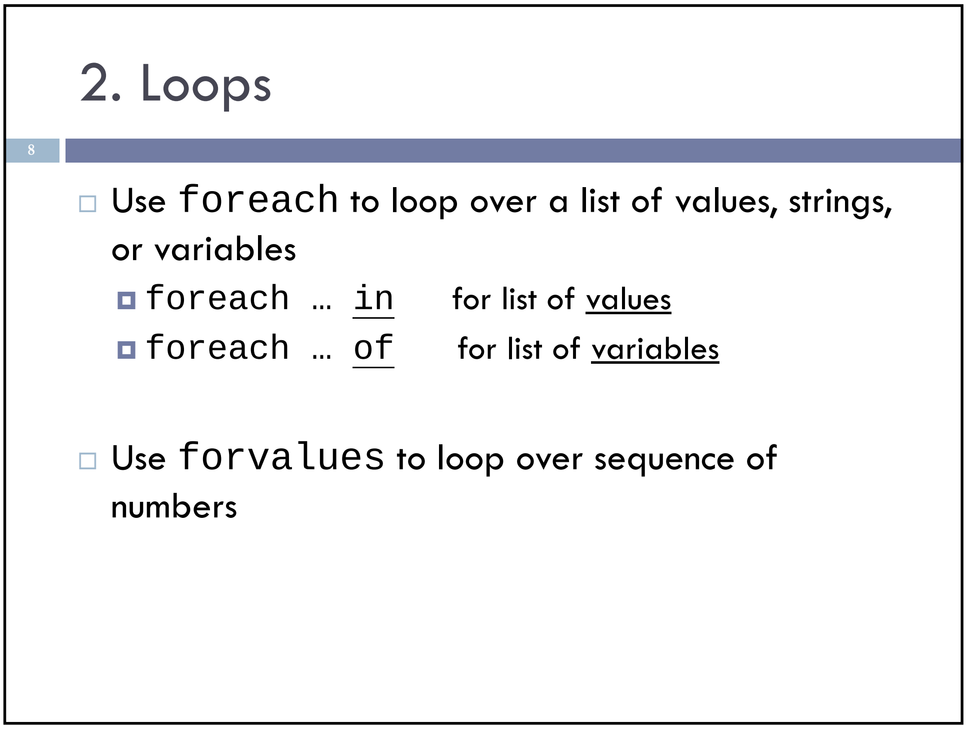


2. Loops

7

- Simulations of a school choice search app
  - 126 different combinations of search parameters
  - Each set of search parameters is used to conduct 109 to 196 different school searches
  - Total number of runs: 18,252

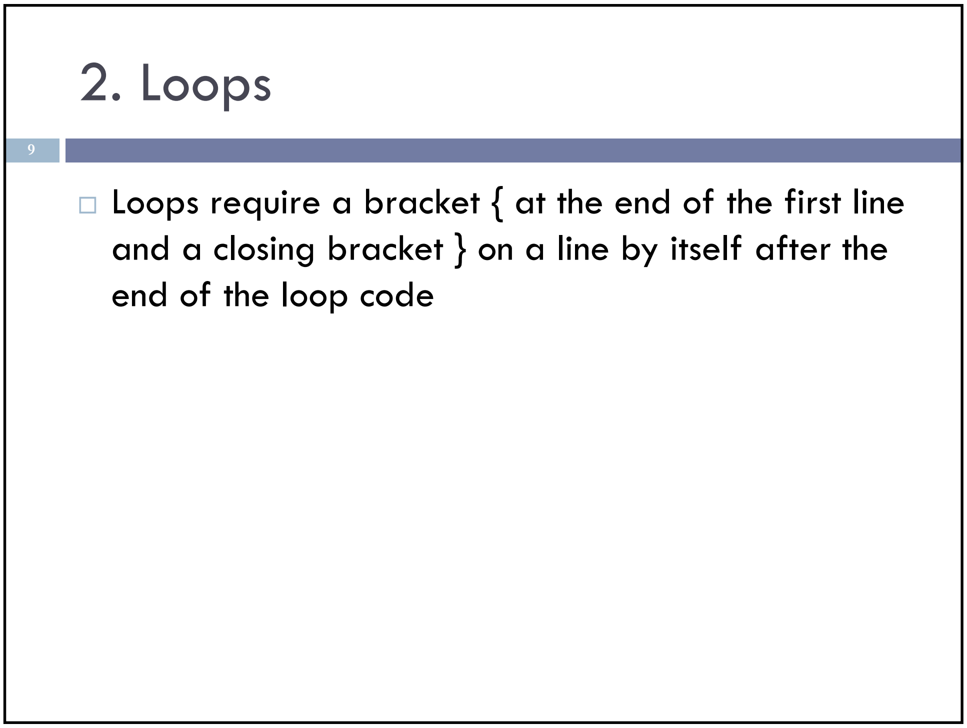
- See do-file example



2. Loops

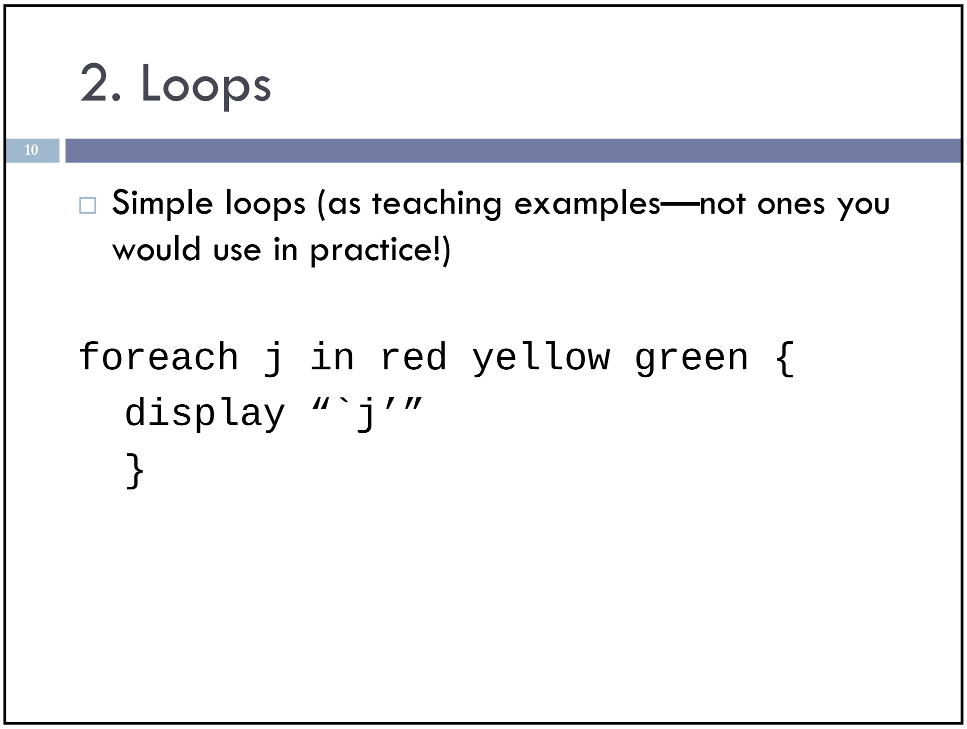
- Use `foreach` to loop over a list of values, strings, or variables
  - `foreach … in` for list of values
  - `foreach … of` for list of variables

- Use `forvalues` to loop over sequence of numbers



2. Loops

- Loops require a bracket { at the end of the first line and a closing bracket } on a line by itself after the end of the loop code

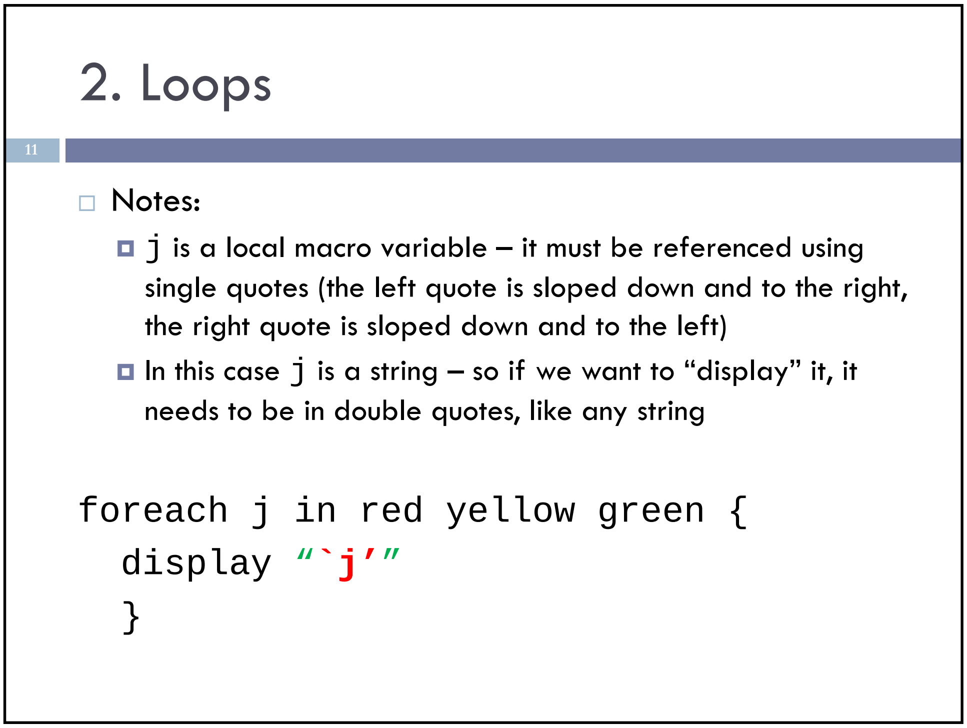


2. Loops

10

- □ Simple loops (as teaching examples—not ones you would use in practice!)

```
foreach j in red yellow green {
  display “`j’”
  }
```

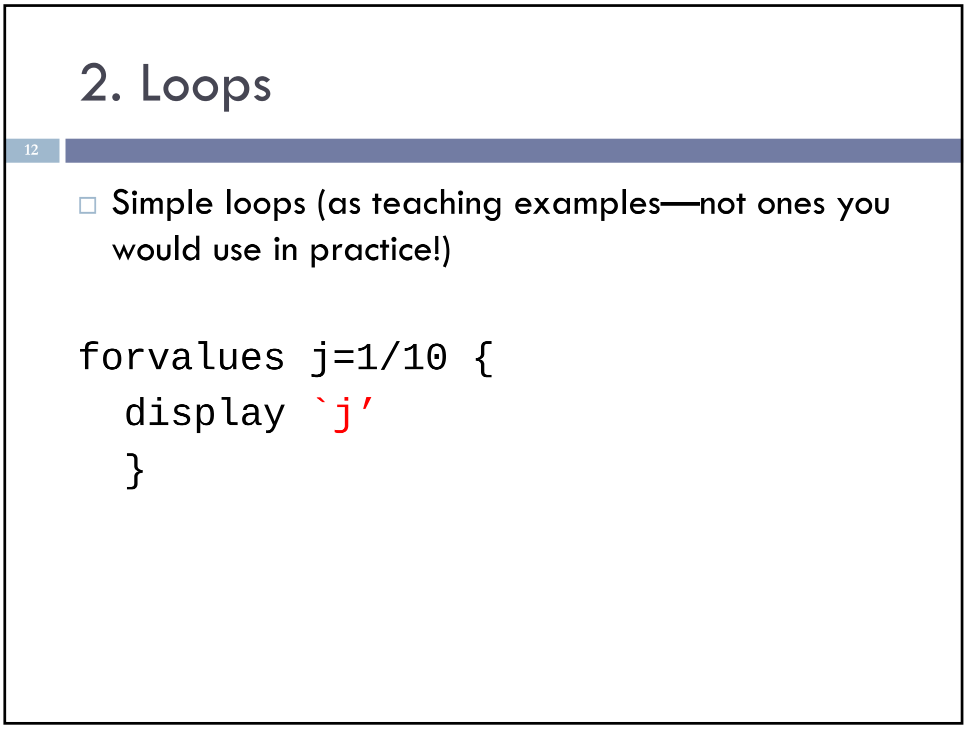


2. Loops

11

- □ Notes:
  - `j` is a local macro variable – it must be referenced using single quotes (the left quote is sloped down and to the right, the right quote is sloped down and to the left)
  - In this case `j` is a string – so if we want to “display” it, it needs to be in double quotes, like any string

```
foreach j in red yellow green {
  display “`j’”
  }
```

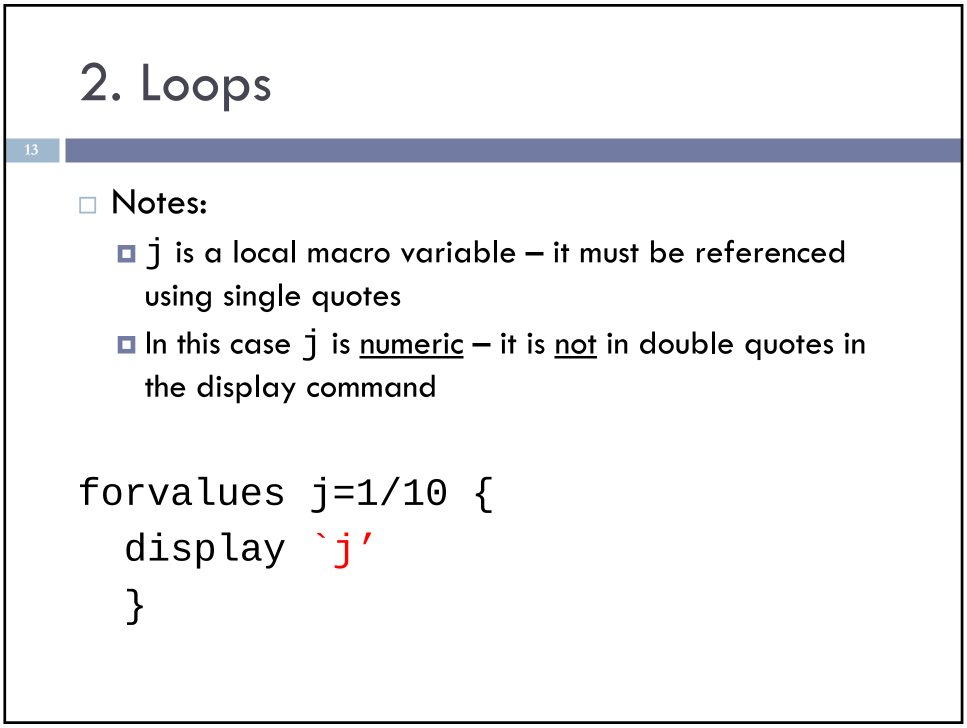


2. Loops

12

- □ Simple loops (as teaching examples—not ones you would use in practice!)

```
forvalues j=1/10 {
  display `j’
  }
```

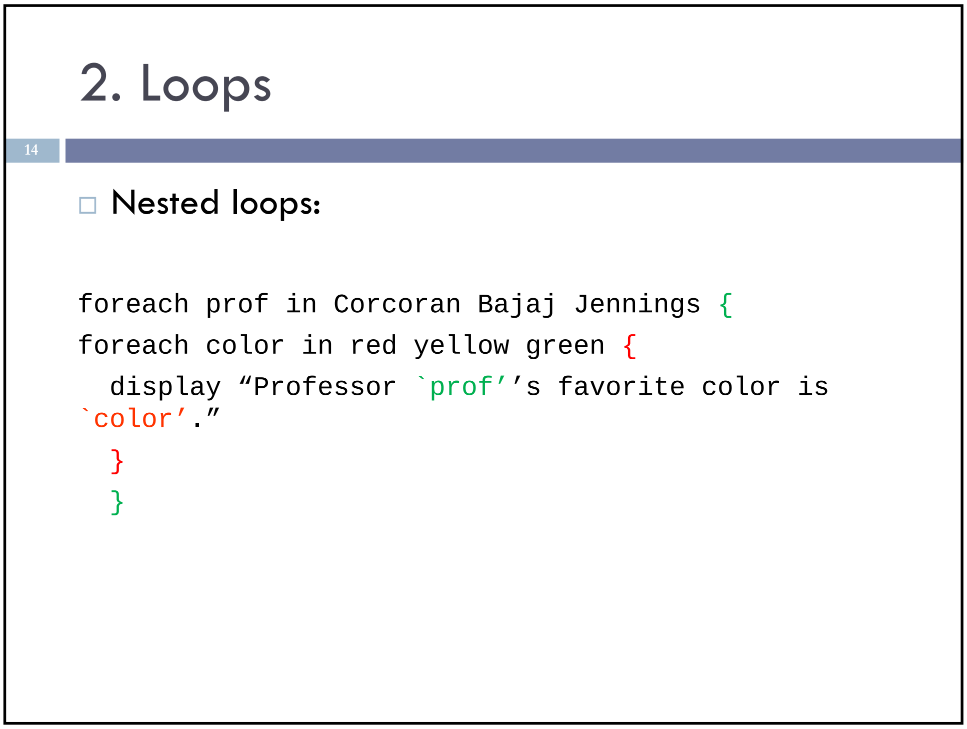


2. Loops

13

- □ Notes:
  - `j` is a local macro variable – it must be referenced using single quotes
  - In this case `j` is numeric – it is not in double quotes in the display command

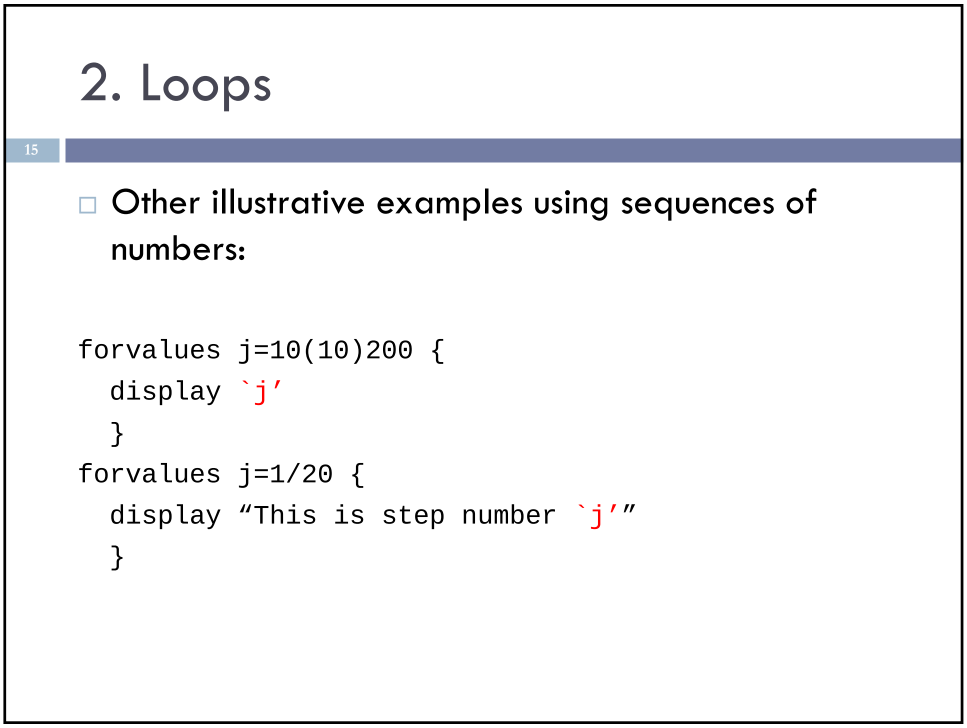
```
forvalues j=1/10 {
  display `j’
  }
```



2. Loops

- Nested loops:

```
foreach prof in Corcoran Bajaj Jennings {
foreach color in red yellow green {
  display “Professor `prof’’s favorite color is
`color’.”
  }
  }
```



2. Loops

- Other illustrative examples using sequences of numbers:

```
forvalues j=10(10)200 {
  display `j’
  }
forvalues j=1/20 {
  display “This is step number `j’”
  }
```

2. Loops

- □ Using variables from your dataset in loops:

```
foreach j of varlist x1mthid x1mthuti x1mtheff {
  sum `j’
  }
```

- □ Good reasons to use numeric suffixes in var names:

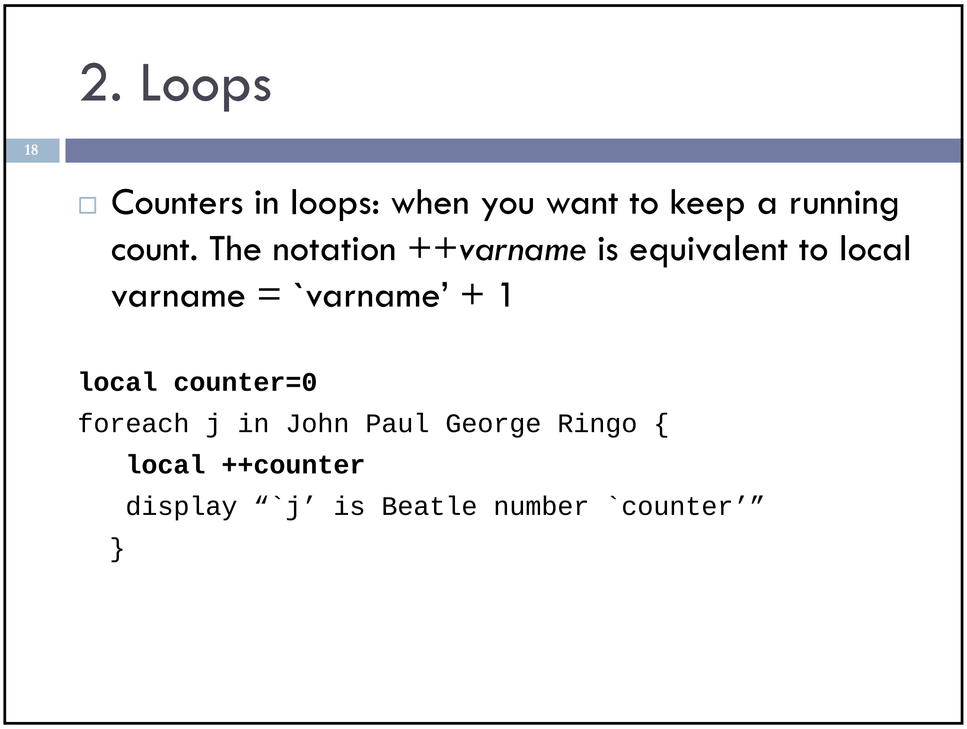
```
forvalues j=1989/2010 {
  replace exp_pupil`j’ = exp_`j’ / enroll`j’
  }
```

2. Loops

- □ Think ahead about how you might use your suffixes in loops:

```
forvalues j=1/10 {
  replace exp_pupil200`j’ = exp_200`j’ /
enroll200`j’
  }
```

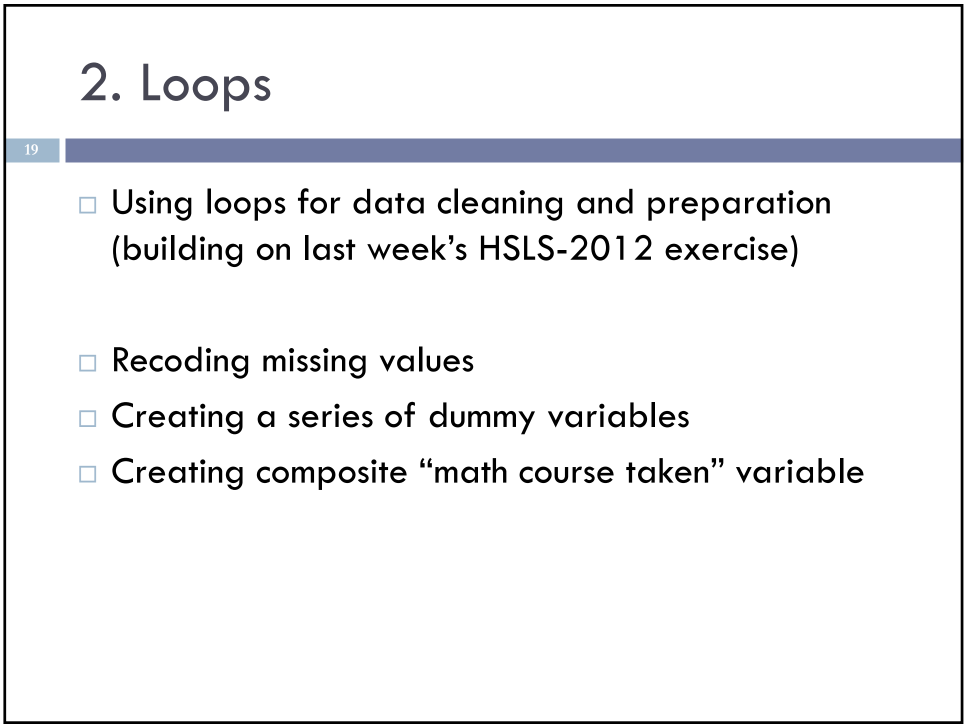
* This will be problematic



2. Loops

- Counters in loops: when you want to keep a running count. The notation ++*varname* is equivalent to local varname = `varname’ + 1

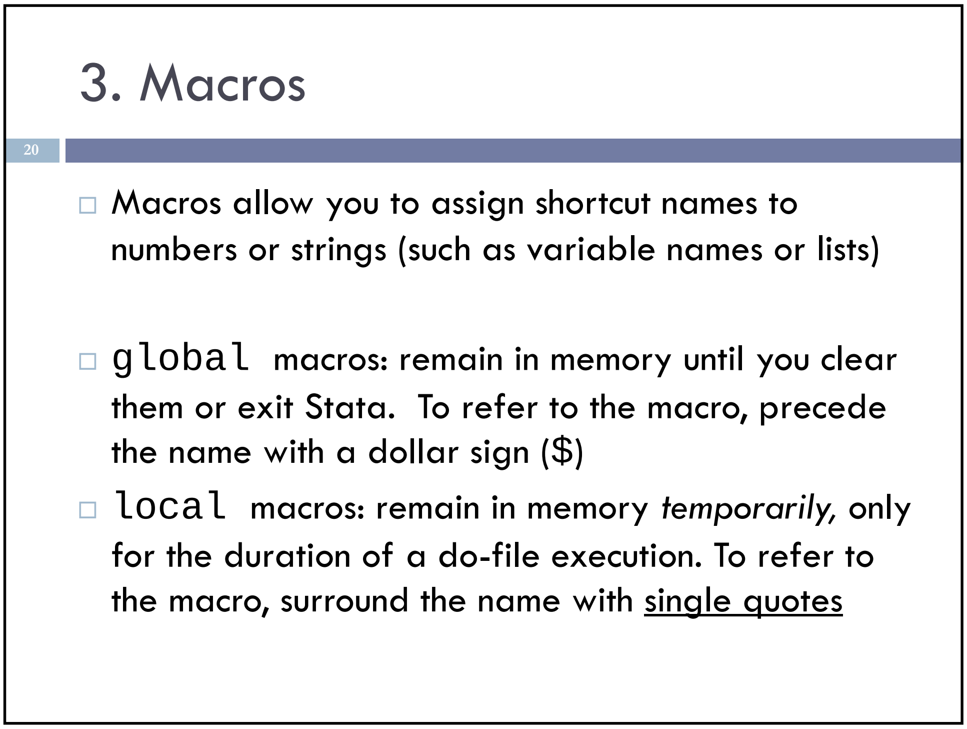
```
local counter=0
foreach j in John Paul George Ringo {
   local ++counter
   display “`j’ is Beatle number `counter’”
  }
```



2. Loops

- Using loops for data cleaning and preparation (building on last week’s HSLS-2012 exercise)

- Recoding missing values
- Creating a series of dummy variables
- Creating composite “math course taken” variable

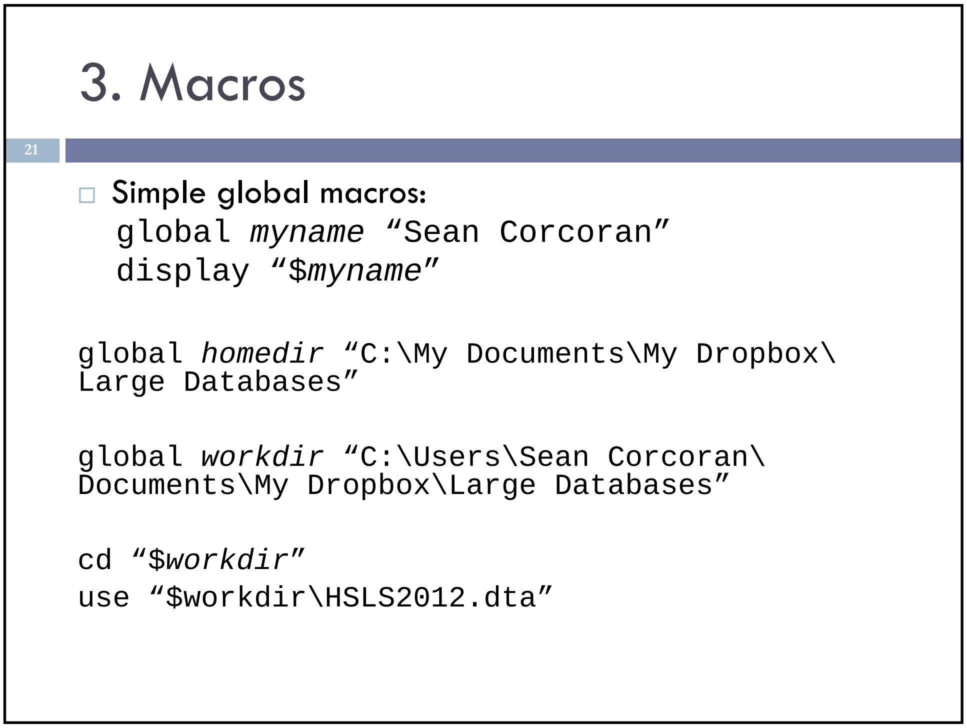


3. Macros

20

- Macros allow you to assign shortcut names to numbers or strings (such as variable names or lists)

- `global` macros: remain in memory until you clear them or exit Stata. To refer to the macro, precede the name with a dollar sign ($)
- `local` macros: remain in memory *temporarily,* only for the duration of a do-file execution. To refer to the macro, surround the name with single quotes



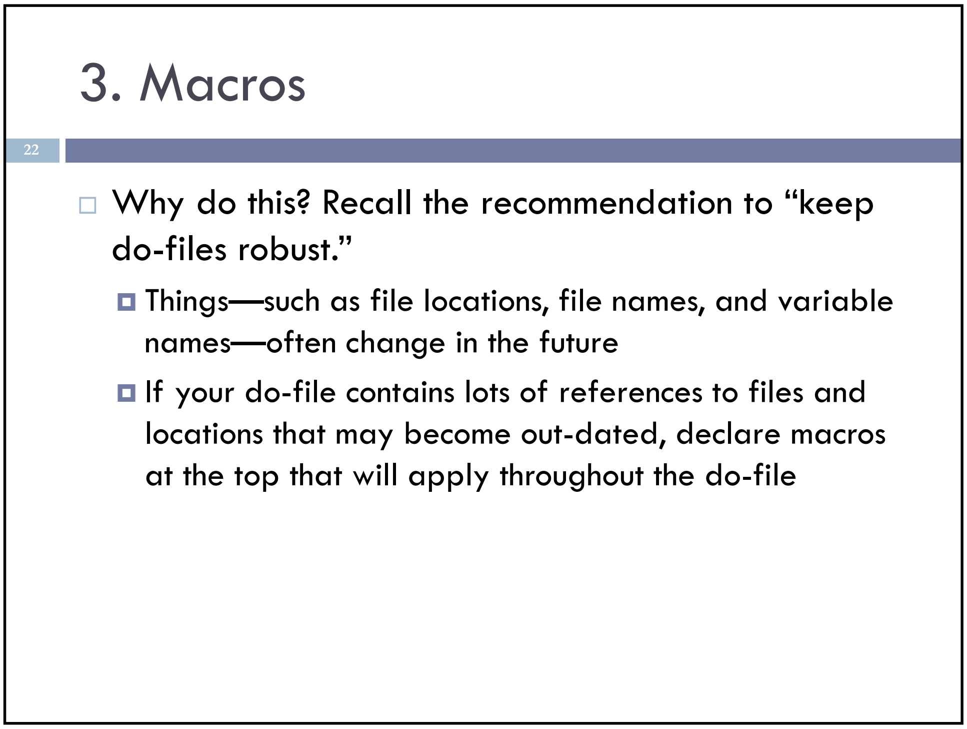
3. Macros

21

- Simple global macros:

```
global myname “Sean Corcoran”
display “$myname”
```

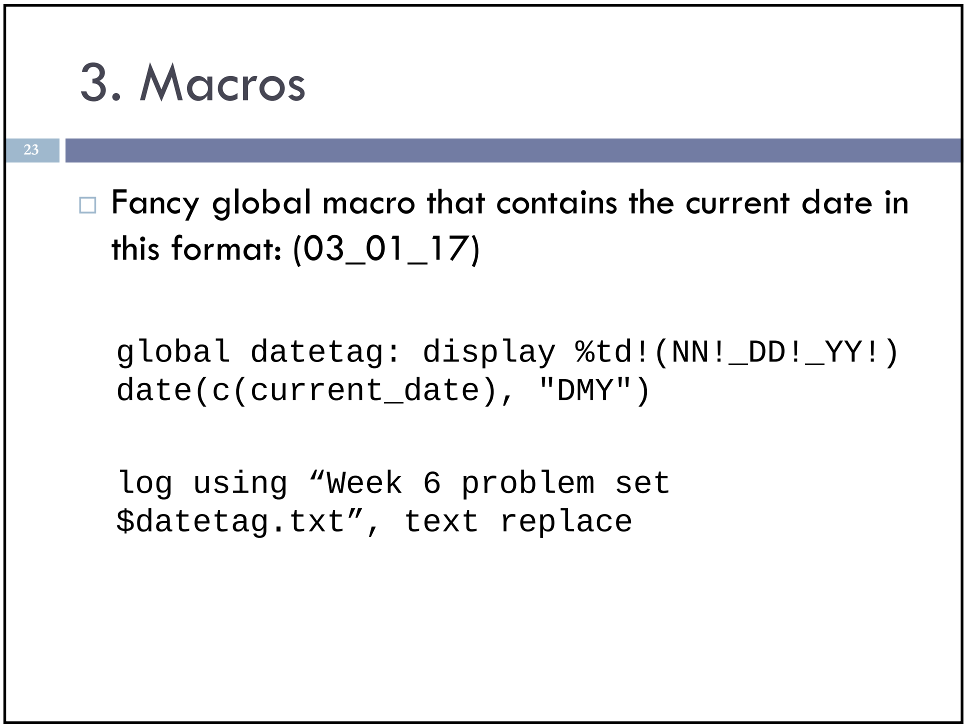
```
global homedir “C:\My Documents\My Dropbox\
Large Databases”

global workdir “C:\Users\Sean Corcoran\
Documents\My Dropbox\Large Databases”

cd “$workdir”
use “$workdir\HSLS2012.dta”
```



3. Macros

22

- Why do this? Recall the recommendation to “keep do-files robust.”
  - Things—such as file locations, file names, and variable names—often change in the future
  - If your do-file contains lots of references to files and locations that may become out-dated, declare macros at the top that will apply throughout the do-file

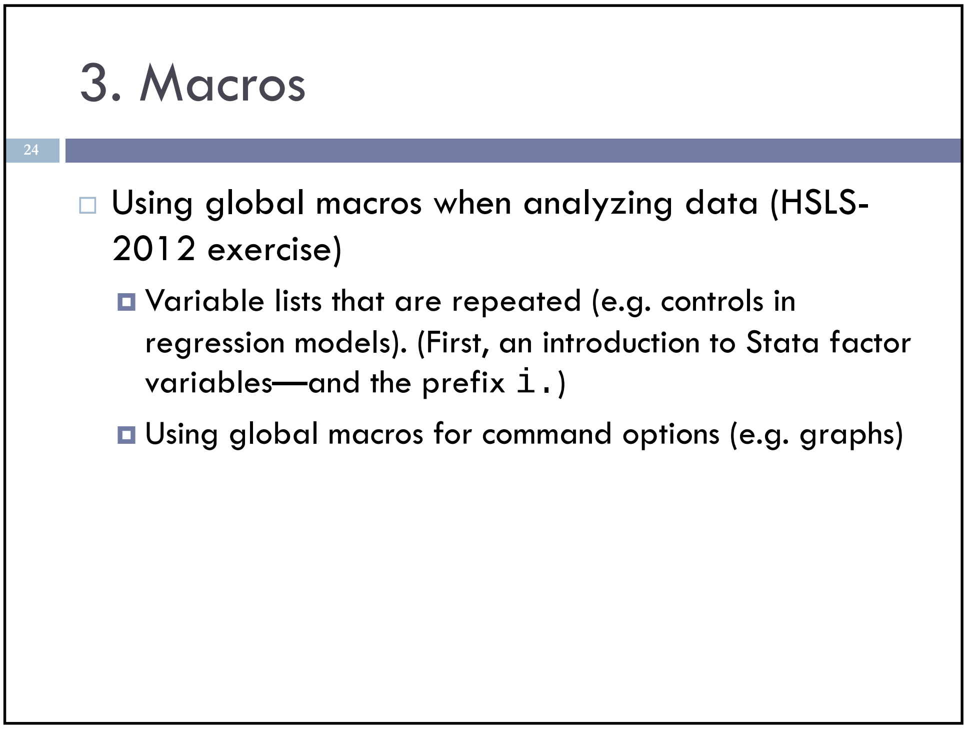


3. Macros

23

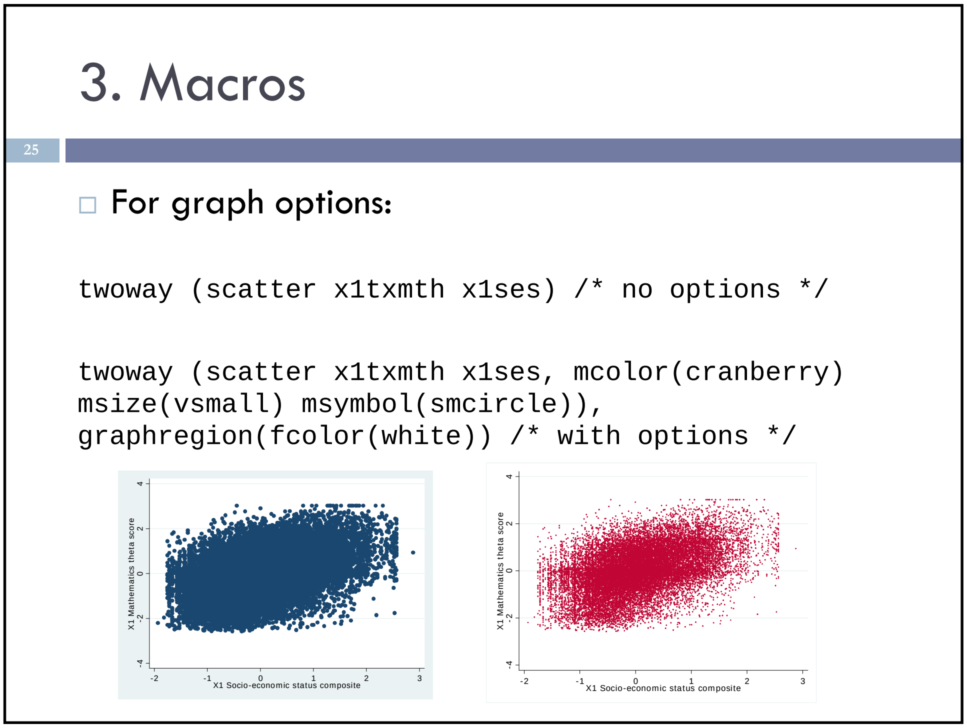
- Fancy global macro that contains the current date in this format: (03_01_17)

```
global datetag: display %td!(NN!_DD!_YY!)
date(c(current_date), "DMY")

log using “Week 6 problem set
$datetag.txt”, text replace
```



3. Macros

- Using global macros when analyzing data (HSLS-2012 exercise)
  - Variable lists that are repeated (e.g. controls in regression models). (First, an introduction to Stata factor variables—and the prefix `i.`)
  - Using global macros for command options (e.g. graphs)



3. Macros

- For graph options:

```
twoway (scatter x1txmth x1ses) /* no options */

twoway (scatter x1txmth x1ses, mcolor(cranberry)
msize(vsmall) msymbol(smcircle)),
graphregion(fcolor(white)) /* with options */
```

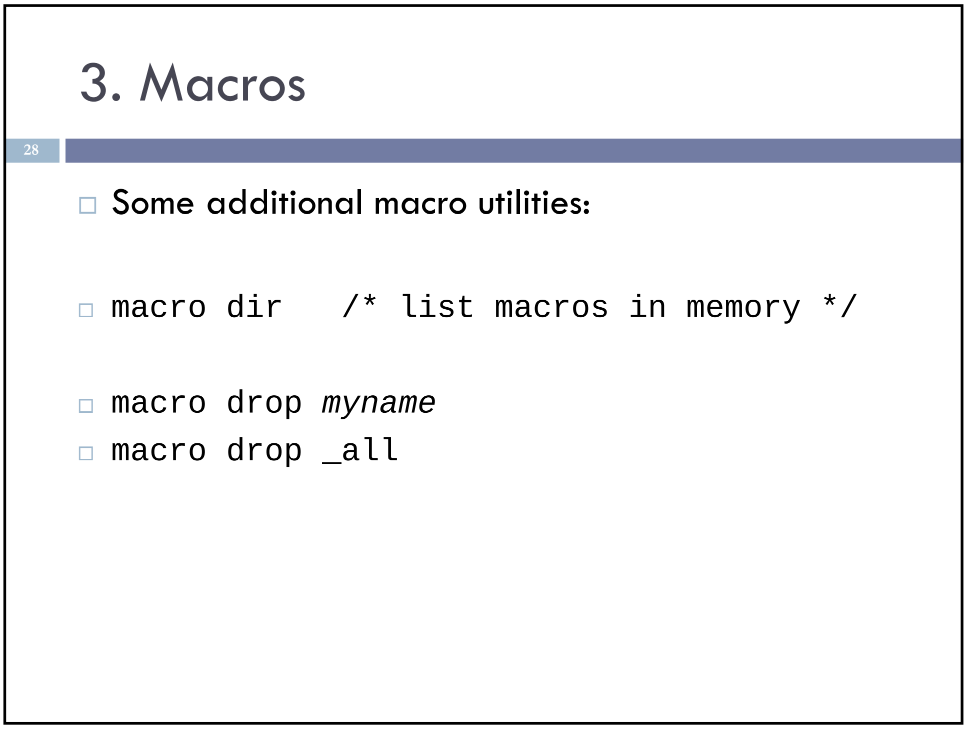
3. Macros

```
global scatterfmt “mcolor(cranberry)
msize(vsmall) msymbol(smcircle))”


global regionfmt
“graphregion(fcolor(white))”



twoway (scatter x1txmth x1ses,
$scatterfmt), $regionfmt
```

3. Macros

- Simple local macros – examples (for use in a do-file). We already saw examples of how `foreach` and `forvalues` uses local macros

```
local myname “Sean Corcoran”
display “`myname’”

local xlist x1ses x1txmth
summarize `xlist’
```



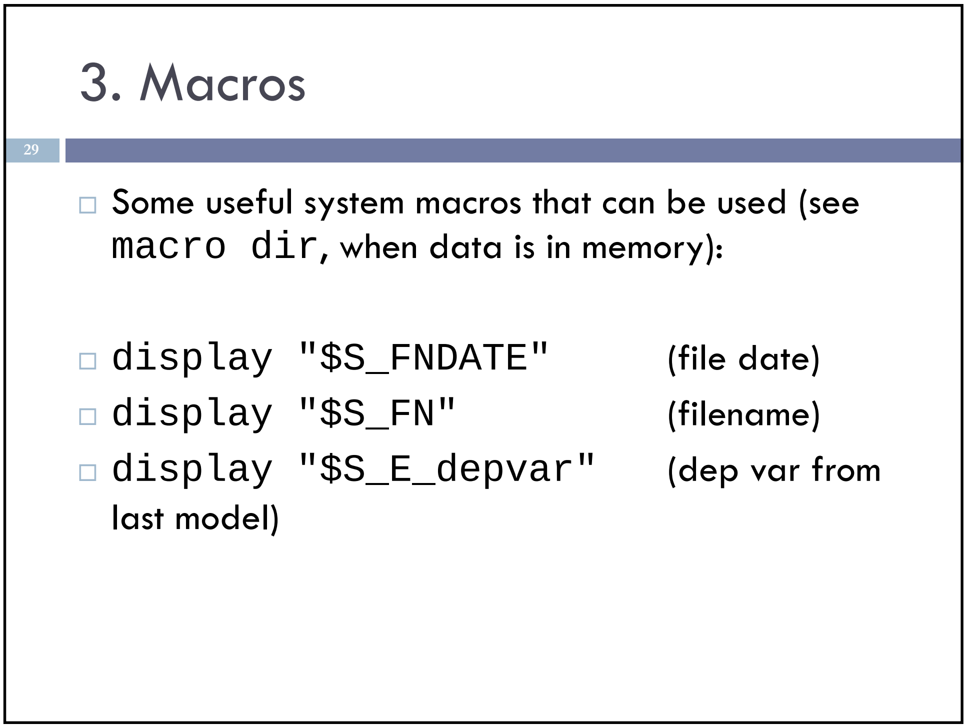
3. Macros

28

- Some additional macro utilities:

- macro dir   /* list macros in memory */

- macro drop *myname*
- macro drop _all

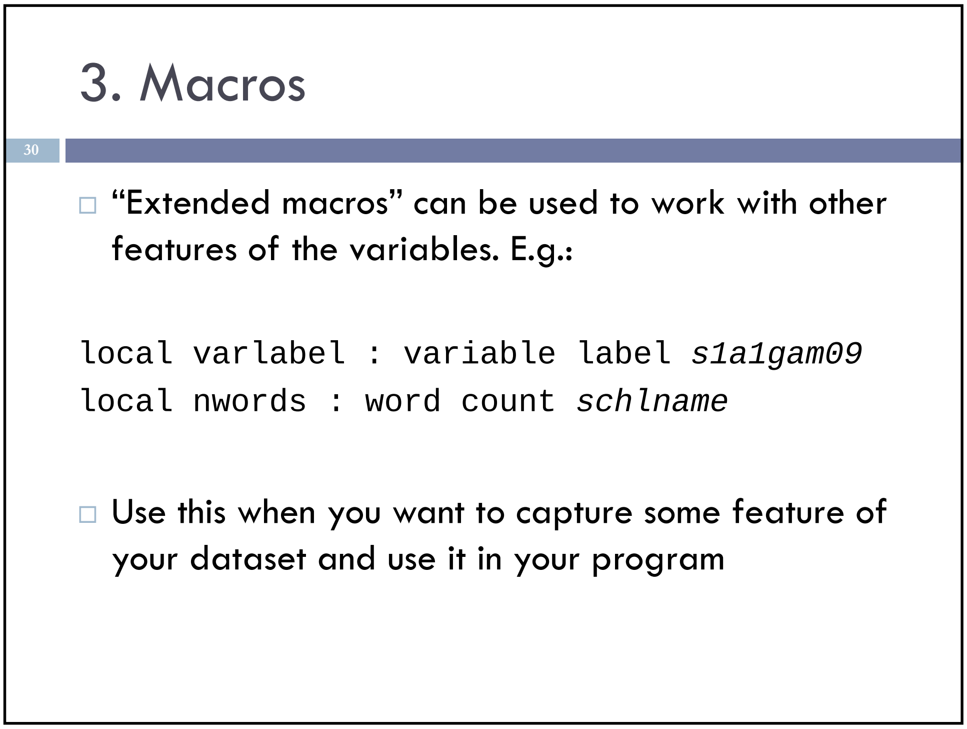


3. Macros

29

- Some useful system macros that can be used (see macro dir, when data is in memory):

- display "$S_FNDATE"   (file date)
- display "$S_FN"   (filename)
- display "$S_E_depvar"   (dep var from last model)

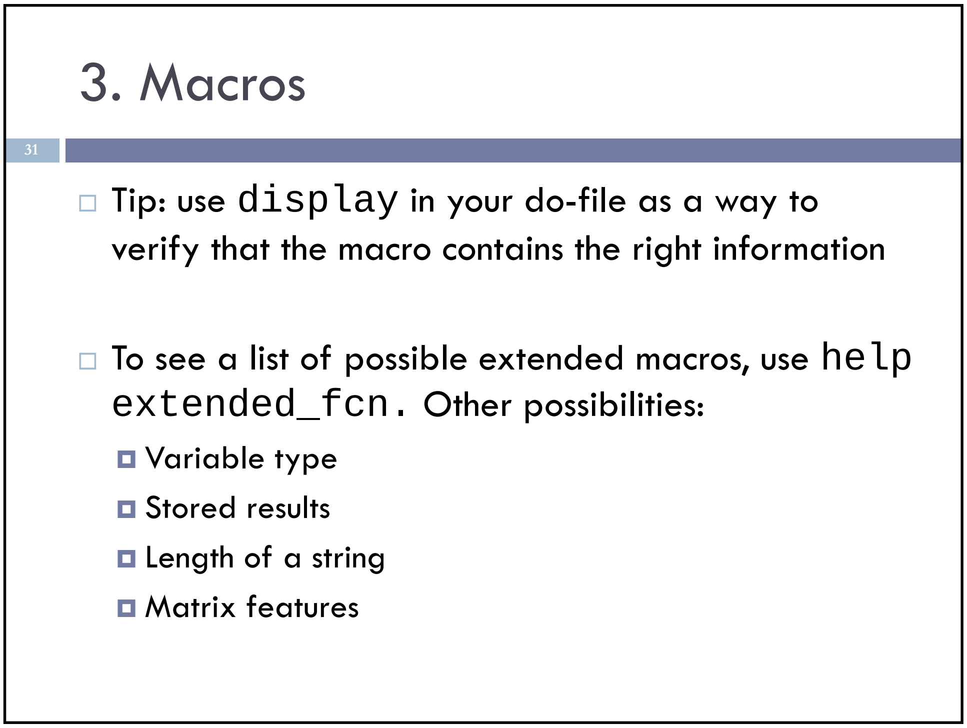


3. Macros

- “Extended macros” can be used to work with other features of the variables. E.g.:

```
local varlabel : variable label s1a1gam09
local nwords : word count schlname
```

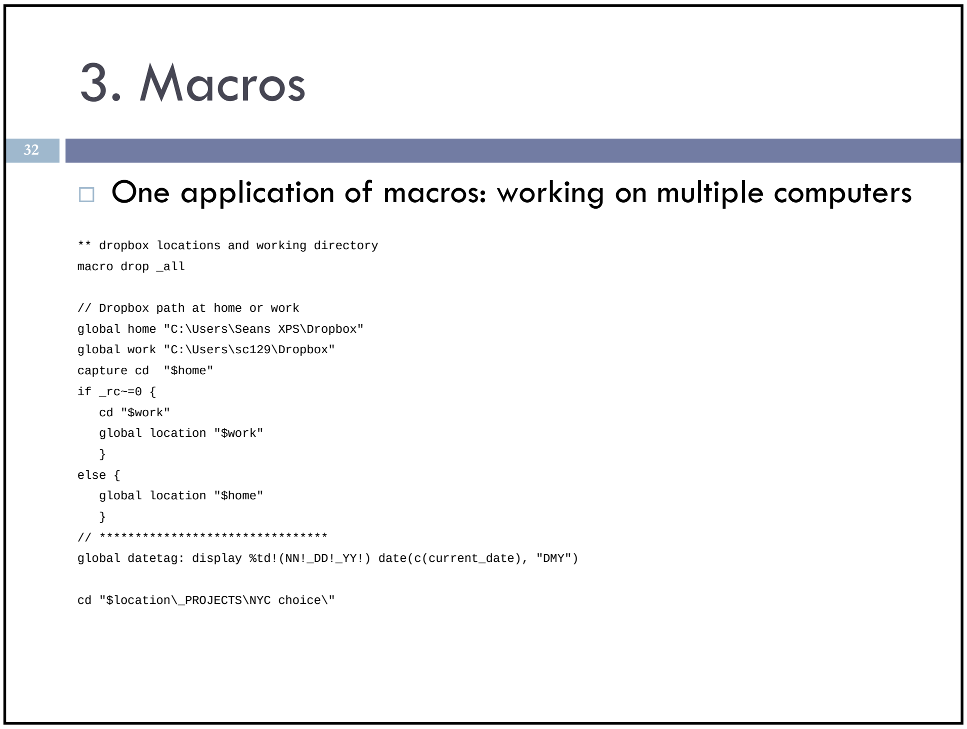
- Use this when you want to capture some feature of your dataset and use it in your program



3. Macros

- Tip: use `display` in your do-file as a way to verify that the macro contains the right information

- To see a list of possible extended macros, use `help extended_fcn`. Other possibilities:
  - Variable type
  - Stored results
  - Length of a string
  - Matrix features

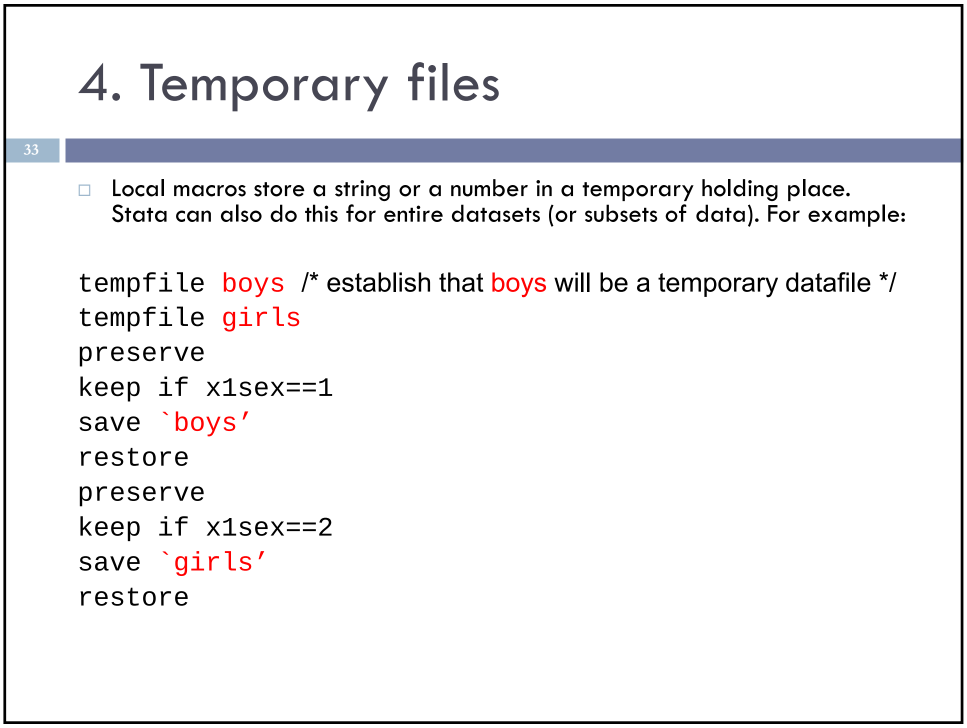


3. Macros

32

- □ One application of macros: working on multiple computers

```
** dropbox locations and working directory
macro drop _all

// Dropbox path at home or work
global home "C:\Users\Seans XPS\Dropbox"
global work "C:\Users\sc129\Dropbox"
capture cd  "$home"
if _rc~=0 {
   cd "$work"
   global location "$work"
   }
else {
   global location "$home"
   }
// ********************************
global datetag: display %td!(NN!_DD!_YY!) date(c(current_date), "DMY")

cd "$location\_PROJECTS\NYC choice\"
```



4. Temporary files

33

- □ Local macros store a string or a number in a temporary holding place. Stata can also do this for entire datasets (or subsets of data). For example:

```
tempfile boys  /* establish that boys will be a temporary datafile */
tempfile girls
preserve
keep if x1sex==1
save `boys’
restore
preserve
keep if x1sex==2
save `girls’
restore
```

5. Scalars

□ A scalar is a name assigned to a number or string (usually a number). When invoking a scalar, no $ or quotes are required. Example:

□ `scalar cpi = 1/1.3256`

□ `display cpi`

□ `gen realgdp = gdp * cpi`

6. Returned results

□ After most commands, Stata retains key pieces of information in memory for later use. See:

□ `return list -` after an analysis command (e.g. `summarize` or `tabstat`)

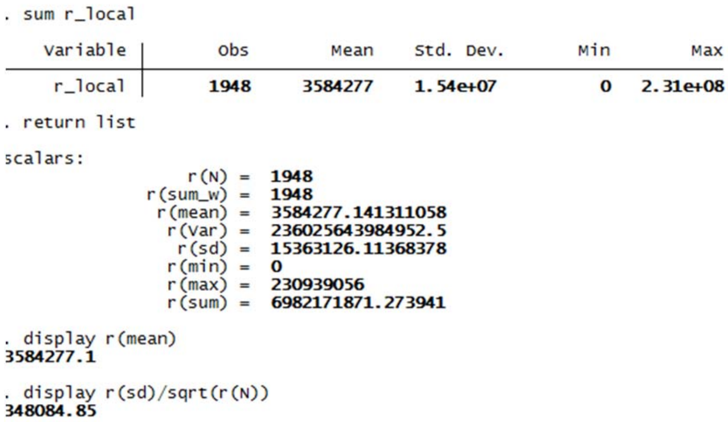
□ `ereturn list –` after an estimation command (e.g. `regress` or `logit`)

6. Returned results

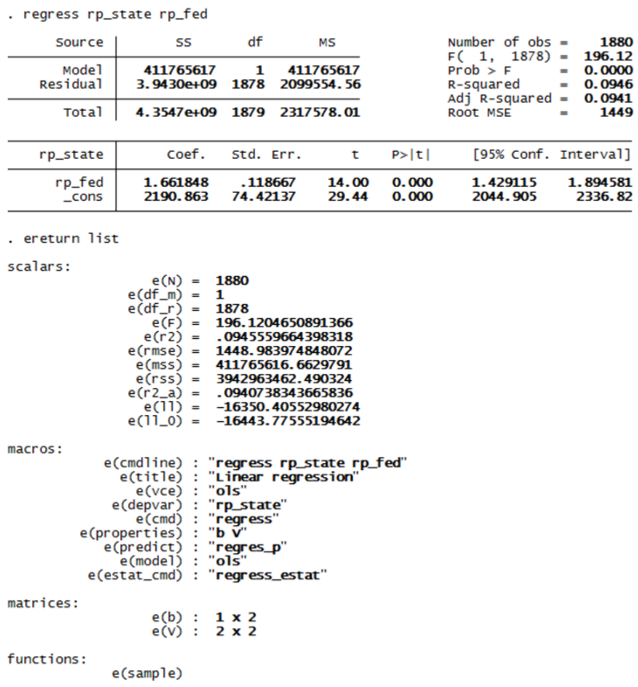
- Saved values in Stata have *more precision* than what is usually displayed in the output (e.g. with the `summarize` command)

- Take advantage of this to avoid the loss of precision that comes from rounding

6. Returned results



```
. sum r_local

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     r_local |      1948     3584277    1.54e+07          0   2.31e+08

. return list

scalars:
                  r(N) =  1948
              r(sum_w) =  1948
               r(mean) =  3584277.141311058
                r(Var) =  236025643984952.5
                 r(sd) =  15363126.11368378
                r(min) =  0
                r(max) =  230939056
                r(sum) =  6982171871.273941

. display r(mean)
3584277.1

. display r(sd)/sqrt(r(N))
348084.85
```



```
. regress rp_state rp_fed

      Source |       SS       df       MS              Number of obs =    1880
-------------+------------------------------           F(  1,  1878) =  196.12
       Model |   411765617     1   411765617           Prob > F      =  0.0000
    Residual |  3.9430e+09  1878  2099554.56           R-squared     =  0.0946
-------------+------------------------------           Adj R-squared =  0.0941
       Total |  4.3547e+09  1879  2317578.01           Root MSE      =    1449

------------------------------------------------------------------------------
    rp_state |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      rp_fed |   1.661848    .118667    14.00   0.000     1.429115    1.894581
       _cons |   2190.863   74.42137    29.44   0.000     2044.905     2336.82
------------------------------------------------------------------------------

. ereturn list

scalars:
                  e(N) =  1880
               e(df_m) =  1
               e(df_r) =  1878
                  e(F) =  196.1204650891366
                 e(r2) =  .0945559664398318
               e(rmse) =  1448.983974848072
                e(mss) =  411765616.6629791
                e(rss) =  3942963462.490324
               e(r2_a) =  .0940738343665836
                 e(ll) =  -16350.40552980274
               e(ll_0) =  -16443.77555194642

macros:
            e(cmdline) : "regress rp_state rp_fed"
              e(title) : "Linear regression"
                e(vce) : "ols"
             e(depvar) : "rp_state"
                e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
              e(model) : "ols"
          e(estat_cmd) : "regress_estat"

matrices:
                  e(b) :  1 x 2
                  e(V) :  2 x 2

functions:
             e(sample)
```

7. Matrices

39

- `matrix list e(b)` – to display a matrix (in this case, the coefficient matrix after a regression)

- Matrices are objects with `r` rows and `c` columns. The notation `a[r, c]` describes the matrix `a` with `r` rows and `c` columns. E.g. `a[16,16]`

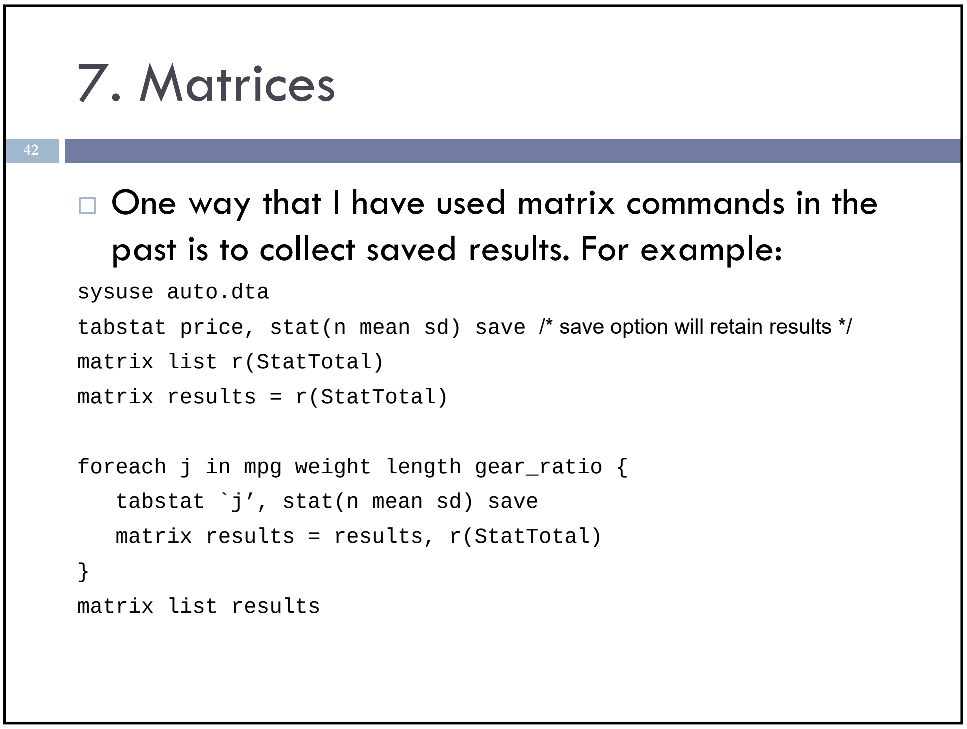
- You can perform many operations with matrices (we won't cover many in this class). Stata's matrix language is called "Mata"

7. Matrices

- Matrix commands:
  - `matrix A = (1, 2 \ 3, 4)` – example of inputting a matrix by hand
  - `matrix list` – to display a matrix
  - `matrix dir` – to see a list of matrices in memory

7. Matrices

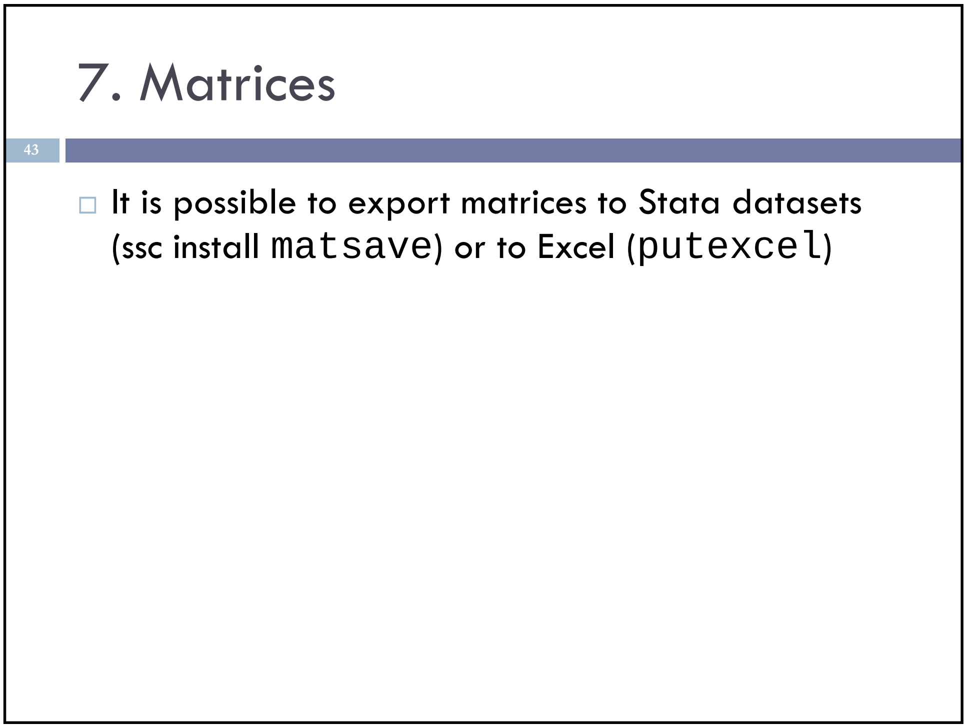
- Matrix commands:
  - `matrix rename` *`m1 m2`* – to rename matrix m1 as m2
  - `matrix drop` *`m1`* – to drop matrix m1
  - `matrix drop _all` – to drop all matrices from memory
  - `matrix` *`A`* `=` *`A`* `\` *`B`* – to add matrix B to the rows of A
  - `matrix` *`A`* `=` *`A`* `,` *`B`* – to add matrix B to the columns of A



7. Matrices

- One way that I have used matrix commands in the past is to collect saved results. For example:

```
sysuse auto.dta
tabstat price, stat(n mean sd) save  /* save option will retain results */
matrix list r(StatTotal)
matrix results = r(StatTotal)

foreach j in mpg weight length gear_ratio {
   tabstat `j’, stat(n mean sd) save
   matrix results = results, r(StatTotal)
}
matrix list results
```



7. Matrices

- It is possible to export matrices to Stata datasets (ssc install `matsave`) or to Excel (`putexcel`)

8. Include command

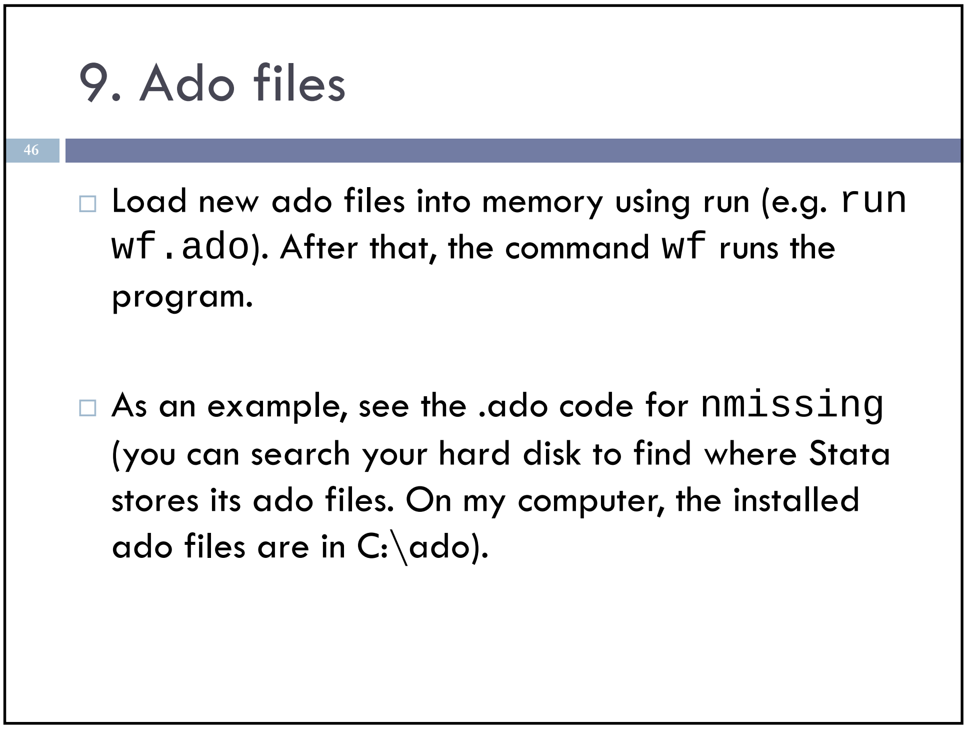
- The include command in a do-file allows you to insert and run code from another file (which Long names `.doi` files, but they could be `.do` files)

- This is helpful when you have some routine procedure (e.g. data cleaning) that is used repeatedly across programs
  - Example: simulation of school choice app

9. Ado files

- `.ado` files are user-written commands that anyone can create
  - The ado file contains a *program*, for example (wf.ado):

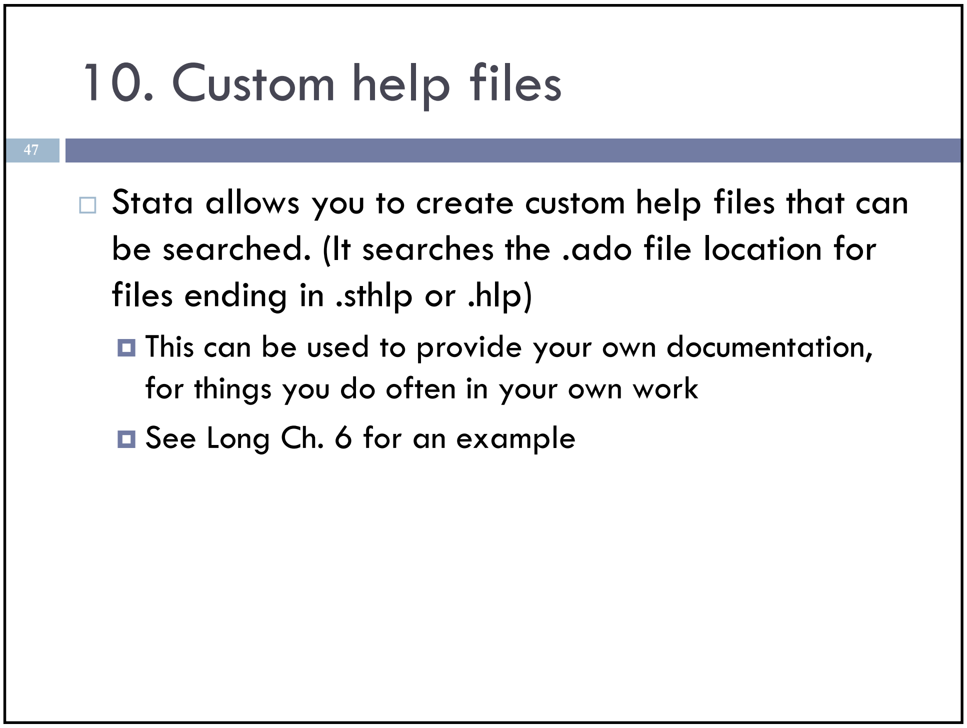
```
program define wf
   cd “C:\My Documents\My Dropbox\Large
Databases”
   end
```



9. Ado files

46

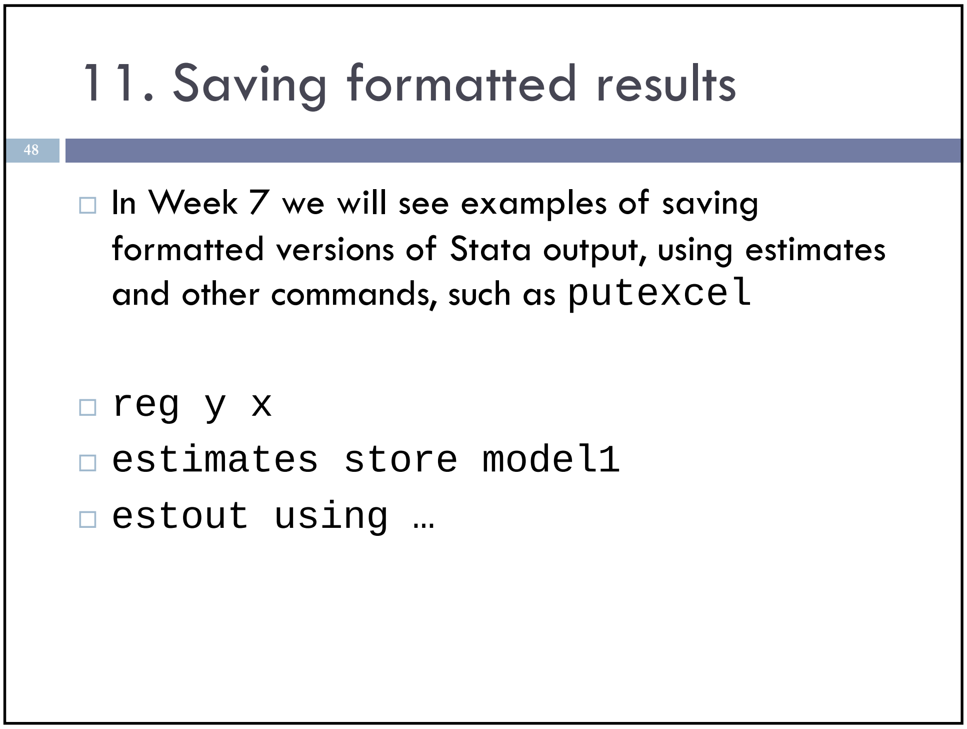
- Load new ado files into memory using run (e.g. `run wf.ado`). After that, the command `wf` runs the program.
- As an example, see the .ado code for `nmissing` (you can search your hard disk to find where Stata stores its ado files. On my computer, the installed ado files are in C:\ado).



10. Custom help files

47

- Stata allows you to create custom help files that can be searched. (It searches the .ado file location for files ending in .sthlp or .hlp)
  - This can be used to provide your own documentation, for things you do often in your own work
  - See Long Ch. 6 for an example



11. Saving formatted results

- In Week 7 we will see examples of saving formatted versions of Stata output, using estimates and other commands, such as `putexcel`

```
reg y x
estimates store model1
estout using …
```