Stata Summer Series

Stata 301 – Automating Tasks and Exporting Output

Previous Stata sessions laid out the fundamentals of interacting with Stata; setting up DO and LOG files to easily save, replicate, modify, and share your analysis and its results; and how to detect and deal with missing and special values as you check your work.

Stata 301 will help you automate your work by using global and local macros, loops, programs, and automated ways to output summary/descriptive statistics.

What you will get out of this session:

- Learn about global and local macros and their differences
- Learn about looping and the different looping syntax
- Learn about writing your own programs
- Ways to automate outputting descriptive/summary statistics (time permitting)

Basic command structure

```
command   objects    conditions          , options
use       file.dta                       , clear
generate  age = 15   if AGE2 == 15
tabulate  state      if country == "US" , missing
```

Helpful resources

- Stata manual: access by typing "help command" in the stata console
- Statalist: https://www.statalist.org/forums/forum/general-stata-discussion/general
  - Often will come up if you google a question that isn't covered by the documentation
- UCLA IDRE: https://stats.idre.ucla.edu/stata/
  - Provides helpful tips on how to use Stata as well as the statistics behind the programming
- UNC CPC: http://www.cpc.unc.edu/research/tools/data_analysis/statatutorial
  - Guide to working with and analyzing data in Stata

Remember: Getting errors is a normal part of programming! The best way to debug is to read through every line carefully.

Next steps:

- Come to our Stata Office Hours!

```
1   *The usual commands at the top to clear anything already open and establish an initial
    setup:
2       *(You can setup a template/starter DO file that has your preferred initial
    setup/parameters)
3   clear
4   capture log close
5   set more off, permanently
6
7   *Set a working directory where all your project files will be located (replace my
    username with your own):
8   cd "C:\Users\urbanmeet\Desktop\"
9
10  log using "StataClass_$S_DATE.log", replace
11
12  use "data_wide.dta", clear
13
14  *------------------------------
15  **MACROS: LOCALS vs. GLOBALS
16  *------------------------------
17
18      *Macros are a kind of temporary holding space akin to "variables" in R, Python,
19      *and other general program languages (remember: think of your data set as a
    spreadsheet,
20      *or in R/Python, etc. terms, a "data frame" and its variables as columns). These
    are also
21      *different from "macros" in Excel, Macs, etc. which are a set of step automated via
    short code scripts.
22  *There are 2 flavors of Macros in Stat: LOCALS and GLOBALS.
23      *LOCALS last only as long as your commands are continuously running within a session
24          *and have limited (local/private) SCOPE, meaning their values are independent
    and cannot be accessed
25          *at different levels (i.e. within loops) in your code or other DO files you
    might run in the same session.
26      *GLOBALS persist for your entire Stata session and have global (public) SCOPE,
27          *meaning their values are accesible and changeable at any point in your session.
28  *Both types of macros can be used to storre strings or numeric values/expressions:
29  local my1stlocalmacro "Hello, world!" //define LOCALS with the "local" command followed
    by what you want to name it.
30  di `my1stlocalmacro'  //refer to locals with an open single quote "`" (shift+ ~), the
    name, and then a close single quote/apostrophe "'"
31      *Uh oh, note that although the macro stored a string, when trying to display the
    value of the LOCAL,
32      *Stata first evaluates it/looks for it as a variable name.
33  *If we enclose the reference in double quotes, that may help us get closer to the
    desired output:
34  di "`my1stlocalmacro'"
35      *On no again, the LOCAL macro is now empty since it is temporary/ephemoral,
36      *we have to define it again and run it together with our display command:
37  local my1stlocalmacro "Hello, world!"
38  di "`my1stlocalmacro'"
39
40  *Globals on the other hand will persist and do not require us to redefine
41      *after the interactive session or DO file (or set of commands in a DO file if we
    only run a section) is finished:
42  global my1stglobalmacro "Hi, friend" //define LOCALS with the "local" command followed
    by what you want to name it.
43  di $my1stglobalmacro //refer to GLOBALS with the dollars sign "$" followed by the
    global name
44  di  "$my1stglobalmacro"
45  di  "$my1stglobalmacroly guy" // if I try to add some text right after the global name,
    Stata thinks that is part of the global's name
46  di  "${my1stglobalmacro}ly guy" // enclose the global's name in curly brackets to
    remove this amgiguity
47
48  *You can use macros to store lists of variables that you want to use over and over
    again,
49  local myvarlist "age gender"
50  sum `myvarlist'
```

```
51  *Or for filepaths to use/save data and results from/to:
52  pwd
53  local workingdir `c(pwd)'
54  di "`workingdir'"
55  global saved_data "`workingdir'\Stata 301\"
56  capture noisily mkdir "${saved_data}" //create the directory, use capture in case it
    already exists...
57  save "${saved_data}data_wide_copy.dta", replace //save the file there
58  *You can also set them to the value of some expression:
59  local my2ndlocalmacro = 2+2
60  di "`my2ndlocalmacro'"
61  *Local macros are also used as part of loops (see below)
62
63  *IN practical terms, I find GLOBALS easier to work with as they persist, and I tend not
    to do anything too too fancy things like
64      *running many PROGRAMS (see below) or DO/ADO (those downloaded user written
    commands) files  within other DO files
65      *where scoping would be a bigger issue. However, it is technically bad form to use
    GLOBALS generally due to potential conflicts.
66      *Strictly speaking, good programming form is to use LOCALS rather than GLOBALS
    unless you have a program that must remember something
67      *from one run to the next. Even with DO files, you can actually pass values from
    one DO file to another run within it with LOCAL arguments (see below)
68
69  *------------------
70  **LOOPING
71  *------------------
72
73  /*Use forvalues for looping over numeric values with consistent intervals   */
74  forvalues i=1(1)10 {        /*syntax is i=min(interval)max                  */
75      display "`i'"               /*call the values with `i'                  */
76      }                           /*close the bracket                         */
77  forvalues i=1(2)10 {        /*loop over odd values                          */
78      display "`i'"
79      }
80  forvalues i=2(2)10 {        /*loop over even values                         */
81      display "`i'"
82      }
83
84  /*Want to generate a variable that's the number of theft arrests per person
85  could write out arr_theft_tot=arr_theft1 + arr_theft2 +...+arr_theft39
86  or use a loop to do this all more easily in one step*/
87  tab arr_theft4, m
88  tab arr_theft4, m nolabel
89
90  /*If any of the values are missing the sum will be
91  missing, so this command sets the original variables to 0 if missing*/
92  forvalues i=1(1)39 {
93      replace arr_theft`i' = 0 if arr_theft`i' == .
94  }
95
96  gen arr_theft=0         /*Can only generate once so do it before the loop*/
97  forvalues i=1(1)39 {
98      replace arr_theft=arr_theft + arr_theft`i'
99      }
100 list id arr_theft arr_theft1 arr_theft2 arr_theft3 arr_theft4  in 1/10

101
102
103 /*Want to generate a variable that's the number of arrests per year per person*/
104 /*Can loop within loops*/
105 forvalues j=2012(1)2014 {
106     gen arr_`j'=0
107     forvalues i=1(1)39  {
108         replace arr_`j'=arr_`j'+1 if arr_y`i'==`j'
109         }
110     }
111 list id arr_2012 arr_y1 arr_y2 arr_y3 arr_y4  in 1/10
```

```
112
113
114   /*Use foreach for looping over non-numeric values or numeric values with
115   inconsistent intervals                                                    */
116   /*Want to create a similar variable as the number of theft arrests for alcohol and
      other type arrests*/
117   foreach k in arr_alcohol arr_other { /*Syntax is "k in" some list strings or numbers
118                                        separted by spaced*/
119       display "`k'"
120       }
121
122   * replace missings
123   forvalues i=1(1)39 {
124       replace arr_alcohol`i' = 0 if arr_alcohol`i' == .
125       replace arr_other`i' = 0 if arr_other`i' == .
126   }
127
128   foreach k in arr_alcohol arr_other {
129       gen `k'=0           /*Can only generate once so do it before the loop*/
130       forvalues i=1(1)39 {
131           replace `k'=`k' + `k'`i'
132           }
133       list id `k' `k'1 `k'2 `k'3 `k'4  in 1/10
134   }
135
136   /*Want to do the same thing for the regression output for the entire sample using a
      varlist*/
137   local varlist "arr_theft arr_alcohol arr_other"
138   *Install outreg2, a user written command to output regression results in a pretty way,
139       *more on this in future weeks...
140   capture ssc install outreg2
141   foreach var of varlist `varlist' { /*Note syntax here is different, "of varlist"
142                                   instead of "in"                           */
143       reg `var' age gender
144       outreg2 using "regressions.doc",  label ctitle(`var') append
145   }
146
147   /*Want to do the same thing for a subpopulation*/
148   local varlist "arr_theft arr_alcohol arr_other"
149   forvalues i=0(1)1 {
150       foreach var of varlist `varlist' {
151           reg `var' age if gender==`i'
152           outreg2 using "regressions_bygender.doc",  label ///
153           ctitle(`var', "gender==`i'") append /*added to ctitle to show subgroup*/
154           }
155   }
156
157   *You can combine the "of varlist" syntax with the flexible ways Stata allows you to
      define a VARLIST
158       *in order loop through a group of variables that start, end, or have some other
      common section in their names.
159   *Let's try only looking at whether age and gender are related to if individuals had
      specifically a 4th arrest of anytype:
160       *first I'll rename ALL the day and year variables to a slightly different pattern
      since we don't want to run our regression on them...
161   rename  arr_d* arrest_d*
162   rename  arr_y* arrest_y*
163   *Now loop through just the arrest "type" indicator variables (theft, alcohol, other)
164       *but just for if each individual had a 4TH arrest of that type:
165   foreach var of varlist arr_*4 {
166       di "`var'"
167       reg `var' age gender
168       outreg2 using "regressions_type.doc",  label ctitle(`var') append
169   }
170   *We could have also used this "foreach... of varlist..." syntax instead of the
      "forvalues" contstruction above
171       *to loop through the 39 different versions of the various arrest type variables.
```

```
172
173
174   *The "forvalues" looping syntax assumes a regular pattern numeric pattern
175       *for values of a variable you might want to step through in your loop.
176       *However, this is not always the case, and trying to refer to a value
177       *that does not exist can lead to errors.
178   *We can use the "levelsof" syntax to loop through just the valid values of
179       *either a numeric or even a string variable:
180
181   *Let's say we don't know what years individuals' 1st arrests occured in or are
182       *working with multiple data sets where the years covered may vary.
183   *If we wanted to run our regressions on whether individuals' 1st arrest
184       *was for each of the 3 potential reasons by year, we would need to manually
185       *check the arrest years in this data set and then loop through only those values.
186   *Or we could just use "levelof" to show and return the different levels of the
      "arrest_y1" variable
187       *and then just loop through those values:
188   levelsof arr*_y1, local(levels)
189   display "`levels'"
190   foreach l of local levels {
191   foreach var of varlist arr_*4 {
192       di "`var' `l'"
193       reg `var' age gender if arrest_y1 == `l'
194       outreg2 using "regressions.doc" ,  label ctitle("`var'_`l'") append
195   }
196    }
197   *rename the variables back:
198   rename   arrest_d* arr_d*
199   rename   arrest_y* arr_y*
200
201   /*If and else statements let you control the
202   flow of your program to execute some commands and/or
203   skip others depending on some condition:*/
204   forvalues i=1(1)39 {
205       if `i'<10 {
206           display "Less than 10"
207           }
208       if `i'>=10 & `i'<20 {
209           display "10-20"
210           }
211       else if `i'>=20     {
212           display "Greater than 20"
213           }
214   }
215
216   /*Suppose you want to run regressions for both binomial and continuous variables,
217   but you don't want to go through the trouble of specifying separate lists for each
      type:*/
218   local varlist "arr_theft arr_theft1"
219       sum arr_theft
220       return list
221       sum arr_theft1
222       return list
223
224   foreach x of varlist `varlist' {
225       qui sum `x'
226       if r(max)==1 {              /*This won't always work in every context make
227                                   sure you know the distribution of your varlist
228                                   before using this syntax to separate out binomials for
229                                   this*/
230           probit `x' age gender
231       }
232       else if r(max)>1  {
233           regress `x' age gender
234       }
235   }
236
237   *For more on returning stored results see the help file for the command you are using
```

```
      or for "return"
238   help summarize
239   help return
240   help program //you can also return results in the own programs (see below) and commands
      (ADO) files you create
241
242   *----------------
243   **PROGRAMMING
244   *----------------
245
246   *Analogous to defining a "function" in R, Python, etc.
247       *Allows you to take one or more pieces of varying input and
248       *execute the same set of commands for the varying input.
249   capture program drop test    /*Drops any previous definition of this program*/
250   program define test          /*Defines the name of the program*/
251       args x                   /*Defines the input(s) in the program*/
252           display "`x'"        /*Tells stata what to do with that input*/
253       end
254                          /*Tells stata this is the end of your program*/
255   test "Stata is great" /*To run the program type the name and the input(s)*/
256   test 45
257
258   /*Suppose you want to make a bunch of different histograms*/

259   histogram arr_theft, percent                    ///
260       title("Arrests for Theft")
      ///
261       color("22 150 210") lcolor(black)           ///
262       width(1)                                    ///
263       ylabel(0(25)100,nogrid)                     ///
264       xlabel(0(5)35,nogrid)                       ///
265       xtitle("Number of Arrests")                 ///
266       graphregion(fcolor(white))                  ///
267       plotregion(style(none))
268
269   capture program drop pretty_histogram
270   program define pretty_histogram
271                   args var title color num
272                       histogram `var', percent
      ///
273                                    title("`title'")
      ///
274                                    color("`color'") lcolor(black)          ///
275                                    width(1)                                ///
276                                    ylabel(0(25)100,nogrid)                 ///
277                                    xlabel(0(5)35,nogrid)                   ///
278                                    xtitle("Number of Arrests")             ///
279                                    graphregion(fcolor(white))              ///
280                                    plotregion(style(none))
281                   graph export "AppendixA_Figure`num'.pdf", replace
282   end
283
284
285   pretty_histogram arr_theft      "Arrests for Theft"     "207 232 243"   1
286   pretty_histogram arr_alcohol    "Arrests for Alcohol"   "70 171 219"    2
287   pretty_histogram arr_2012       "Arrests in 2012"       "253 216 112"   3
288   pretty_histogram arr_2013       "Arrests in 2013"       "253 191 17"    4
289   pretty_histogram arr_2014       "Arrests in 2014"       "202 88 0"      5
290
291   *PROGRAMS are a good illustration of scoping issues between GLOBALS and LOCALS.
292       *What do you think the value of Y will be at the end in each of the situations below?
293   *1. LOCAL example:
294   capture program drop localexample
295   program define localexample
296       args x
297   local y = `x' +1
298   di "`y'"
299   end
```

```
300
301   *2. GLOBAL example:
302   capture program drop globalexample
303   program define globalexample
304       args x
305   global y = `x' +1
306   di "`y'"
307   end
308   *What happened in the program, stays in the program for the local,
309   localexample 10
310   di "`y'"
311
312   globalexample 2
313   di "$y"
314   *which is not the same case for the global, it is updated within the program.
315
316   *Programmers NOTE: this can also be important for running DO/ADO files within
317   *another DO file. If you update a global in the child DO file you are running from
318   *your Master/Parent DO file, it will also be updated for the Master DO file and any
319   *other subsequen child DO files you run in it after the original child DO file is done
320   *excecuting. This is not the case with local macros. Good programming form would be to
321   *generally use local arguments and returned values passed explcitily between DO files and
322   *programs rather than globals, generally. However, that is beyond the SCOPE of this
      session.
323
324   *-------------------------------------
325   **OUTPUTTING DESCRIPTIVE STATISTICS
326   *-------------------------------------
327
328   *This will help you with speeding up your work by automating the outputting of your
      results from Stata to external files
329   use "FINRA_01.dta", clear  // use the old FINRA data file from the intro trainings
330   **Three (3) frequently used ways of outputting results to external files, each with
      pros and cons**
331       * 1) POSTFILE:
332           *Pros:
333               *Native to Stata
334               *Fairly flexible on what data you can put and in what order
335           *Cons:
336               *Less easy to format and include context like variables names and labels
337               *Less easy to use and output lots of information as it is based on looping
      and writes 1 line at a time
338
339       * 2) TABOUT:
340           *Pros:
341               *Easy to use and output lots of information
342               *Easy to format and include context like variables names and labels
343           *Cons:
344               *Not built into Stata-- have to download and install from SSC
345               *Less flexible on what statistics you can run and in what order
346
347       * 3) PUTEXCEL:
348           *Pros:
349               *Similar to POSTFILE in that it is built into Stata
350               *Easier to edit and manipulate specific cells/data points and add
      labels/text unlike POSTFILE,
351           *Cons:
352               *Less automated and more tedious as you must specify which cells and ranges
      to write to in Excel with each command
353
354   ** 1) POSTFILE: first way of outputting descriptive statistics
355
356   *Postfile creates a second temporary data set in memory, separate from the one you have
      open
357   *You will run your summary statistics on the active data set, store the results in
      local macros,
358   *write the results in the locals to the temporary data set,
359   *and, finally, open the temporary data set and export it to an external Excel file
```

```
360
361   *First create a temporary data set in memory with a temporary name assigned to the
      local macro "memhold"
362   tempname memhold
363   *In this line we actually create the temporary dataset in memory (but which is not
      active) in the namespace "memhold"
364   *The different variables/columns we want in our temporary data set are listed between
      "memhold" and using
365   *Using specifies where the data set actually gets saved/stored eventually
366   postfile `memhold' str20(varname) meanmale meanfemale pvaloftest using
      "${saved_data}FINRA_02_postfile.dta", replace
367   *Next we will specify the different variables that we want statistics on in a loop we
      can iterate through
368   *In this case we want to see if the mean/average value of these various variables
      differ significanlty by gender (A3)
369   foreach var in A3A    r_white r_black r_hisp r_asian r_other reg_ne reg_mw reg_south
      reg_west {
370           *First find the mean value of the varible of interest for males (A3==1)
371           su `var' if A3 == 1
372           return list     //Return list displays the statistics stata saved and can be
      stored in your locals
373           *Store this average termporarily in a local macro called "meanmale"; we will
      write this vlaue to our temporary data file below in the "post" command line
374           loc meanmale = r(mean)
375
376           *Next, find the mean value of the varible of interest for females (A3==2)
377           su `var' if A3==2
378           *store the result to a local macro called "meanfemale"
379           loc meanfemale = r(mean)
380
381           *Then, we want to see if the average for males and females differs
      statistically,
382           *so we will use the mean estimation and the postestimatation test command to see.
383           *We could also use "ttest" to do this in one line, but "mean" is more flexible
      and allows us to use weights if we want
384           mean `var', over (A3)
385           test [`var']Male = [`var']Female
386           *Save the 2-sided p-value of this difference of means test to a third local macro
387           local pvaloftest = r(p)
388       *Finally, for the variable we are actively looping through,
389       *write the mean value for males, mean value for females, and 2-sided p-value of the
      difference
390       *as a new observation (row) in our temporary data file (not the one we have open
      actively and are analyzing)
391       post `memhold' ("`var'") (`meanmale') (`meanfemale') (`pvaloftest')
392       }
393   *Postclose tells Stata we are done writing to the temporary data file we created
394   *and to finalize/save it where we specified in the "postfile" command
395   postclose `memhold'
396
397   *Then we just have to open up and export that new data file and export the results
398   *First we may want to preserve our active data file, though, so we can restore it
      quickly to conduct additional analyses
399       *This part is optional, though, since you could just reopen your original data file,
400       *or save a working version here and reopen it after the export instead of using
      preserve/restore
401   preserve //optional
402   *Open up your results file data set:
403   use "${saved_data}FINRA_02_postfile.dta", clear
404       *Export to Excel:
405       export excel using "${saved_data}FINRA_02_postfile.xls", replace
406   restore //optional
407
408   *Take a look at the resulting Excel file for the output--its gives you the information
      you need,
409   *but the numbers are not formatted and there are no column headings to tell you what
      metric is in each column
410   *Sometimes, the better option is... TABOUT
```

```
411  *(though you would not be able to pull out and write the p-value for the significance
     test in TABOUT easily)
412
413
414  * 2) TABOUT: second way of outputting descriptive statistics that is pretty
     automated/easy to use
415  *Download command:
416  capture ssc install tabout
417  *Can use to show tabulations
418  tabout A3 using "${saved_data}FINRA_02.xls", cells(col) stats(chi2) replace
419
420  *Or crosstabs (by gender in this case):
421  foreach characteristic of varlist A4A A6 A5 {
422      tabout `characteristic' A3 using "${saved_data}FINRA_02.xls", cells(col) stats(chi2)
      append
423  }
424
425  *Can also use to show summary statistics with the "sum" option
426  *You need to specify which statistic to show for each variable whose mean, median, etc.
     you want to show
427  *Tabout then runs this by the differing variables listed before "using"
428  tabout A3 A4A using "${saved_data}FINRA_02.xls", sum c(mean A3A  mean emp_full) f(4)
     append
429  *f(4) indicates four digits after the decimal
430
431  *Again, you can use loops and loops within loops to automate and quicken your work...
432  foreach characteristic of varlist A4A A6 A5 {
433  foreach outcome of varlist A3A   emp_full {
434  tabout `characteristic' using "${saved_data}FINRA_02.xls", sum c(N `outcome' mean
     `outcome') f(4) append
435  }
436  }
437  *Can also download and use the OUTSUM command from SSC to export summary statistics,
438  *but usually I find tabout along with the sum option to be adequate
439
440
441  * 3) PUTEXCEL: another way of outputting results from Stata to external file (the most
     tedious),
442  *but also the most flexible in terms of what to place in each individual cell and how
     to format it.
443  *First specify the file that you want to write your output to:
444  putexcel set "${saved_data}test.xls", replace
445  *Can write stored results/contents of local macros similar to postfile:
446  sum A3A
447  local avg = r(mean)
448  putexcel B2= `avg'
449  local stdDev  = r(sd)
450  putexcel B3  = `stdDev'
451  *Can also add labeling more easily than in postfile:
452  putexcel B1="Age" A2="Average" A3="Standard Deviation"
453
454  capture log close
455
456  *-------------------------
457  **EXERCISES
458  **DATASET: data_wide.dta
459  *-------------------------
460
461  use "data_wide.dta", clear
462
463  **LOOPING
464  /* 1. Using a loop, generate a variable equal to the total number of arrests each
465      person had had*/
466  gen arr_total=0
467  forvalues i=1(1)39 {
468      replace arr_total=arr_total+1 if arr_d`i'!=.
469  }
470
```

```
471  /* 2. Use a loop to generate a variable to equal their age at arrest, e.g. age1 for
472  age at arr_d1 and age2 for age at arr_d2, etc*/
473  forvalues i=1(1)39 {
474      gen age`i'=floor((arr_d`i'-birth_d)/365)
475  }
476
477  /*3. Using a loop, create a variable for the date of their first arrest and the
478      date of their last arrest*/
479
480  *Answer version 1
481  gen first_arr=.
482  gen last_arr=.
483  format first_arr last_arr %td
484  forvalues i=1(1)39 {
485      replace first_arr=arr_d`i' if arr_d`i'<first_arr
486      replace last_arr=arr_d`i' if (arr_d`i'>last_arr | last_arr==.) & arr_d`i'!=.
487  }
488
489  *Answer version 2 - based on the idea that we know these are orderd*/
490  gen first_arr2=arr_d1
491  gen last_arr2=.
492  format first_arr2 last_arr2 %td
493
494  forvalues i=2(1)39 {
495      local j=`i'-1
496      display "`i'"
497      display "`j'"
498      assert arr_d`j'<=arr_d`i'
499      replace last_arr2=arr_d`j' if arr_d`i'==. & arr_d`j'!=.
500      replace last_arr2=arr_d`i' if `i'==39 & arr_d`i'!=.
501  }
502
503  /*4. Using a loop, tab all of arr_theft1, arr_alcohol1, arr_other1, arr_total*/
504
505  foreach x in arr_theft1 arr_alcohol1 arr_other1 arr_total {
506      tab `x'
507  }
508
509  /*Write a program that says "Hello, Name" and replaces a name that you input*/
510  capture program drop hello
511  program define hello
512      args name
513      display "Hello `name'"
514      end
515
516  hello "Matt"
517
518  *-------------------------------------------------------------------------
519  **EXERCISES for automating Output of summary/descriptive statistics:
520  **DATASET: FINRA_01.dta
521  *-------------------------------------------------------------------------
522
523  *1. Start a new log file, open the FINRA_01 data file again, and save a working version
     under a different filename right away.
524  capture log close
525  log using "Stata_Summary output_EXERCISES_$S_DATE.log", replace
526  use "FINRA_01.dta", clear
527  save "${saved_data}FINRA_02.dta", replace
528  *2. Output descriptive tables that show the share of observations by their age group,
     race/ethnicity, gender/sex, income category, and census region.
529  tabout  A4A a_2029 a_3039 a_4049 a_5059 a_60plus A3 i_catvar censusreg using
     "${saved_data}FINRA_one-way_tabs.xls", cells(col)  replace oneway //the "oneway" option
     produces a one-way tabulation (like the tab1 command) rather than a crosstab
530  *3. Output the distribution (crosstab) of race/ethnicity by gender/sex
531  tabout  A4A A3  using "${saved_data}FINRA_x-tab_race-sex.xls", cells(col)   replace
532
533  *4. Output the number of observations at each education level and the median value of
     age for each education level.
```

```
534   tabout A5 using "${saved_data}FINRA_ageN-p50.xls", sum c(N A3A  p50 A3A) f(4) replace
535
536   *5. Repeat question 4, but use a different method for exporting your output (TABOUT,
      POSTFILE, or PUTEXCEL)
537   *USING POSTFILE:
538   tempname memhold
539   postfile `memhold' str20(varname) Nsample medianAge  using
      "${saved_data}FINRA_q5postfile.dta", replace
540   forvalues i=1/6{
541           sum A3A if A5 == `i'
542           local N =  r(N)
543           local med =  r(p50)
544       post `memhold' ("A5 (eduation level) == `i'") (`N') (`med')
545       }
546   postclose `memhold'
547   preserve
548   use "${saved_data}FINRA_q5postfile.dta", clear
549
550       export excel using "${saved_data}FINRA_q5postfile.xls", replace
551   restore
552
553   log close
554   clear all
555
556
```

```
1   **--------------------------------------------------------------------------------
2   * Stata Users Group
3   * Descriptives_Example.do
4   * Created       5/29/2019   by M Gerken
5   **-------------------------------------/------------------------------------------
6   * This program gives a few examples of quickly generating descriptive statistics
7   **--------------------------------------------------------------------------------
8
9   **--------------------------------------------------------------------------------
10  * Directories & Locals
11  **--------------------------------------------------------------------------------
12
13  local data "E:\Stata help\descriptives_example.dta"
14  local output "filepath"
15
16  cd "`output'"
17
18  **--------------------------------------------------------------------------------
19  * Method 1: Using collapse
20  *           Example below calculates the mean, median, min, max across 5 vars
21  **--------------------------------------------------------------------------------
22
23      use "E:\Stata help\descriptives_example.dta", clear
24
25      * create holder variables
26      foreach x of varlist var1-var5 {
27          gen `x'_mean = `x'
28          gen `x'_median = `x'
29          gen `x'_min = `x'
30          gen `x'_max = `x'
31      }
32
33      * Asterisk used as wild card,
34      collapse (mean) *_mean (median) *_median (min) *_min (max) *_max
35
36      * Order variables
37      order var1_* var2_* var3_* var4_* var5_*
38
39      export excel using "E:\Stata help\name.xls", sheet("Des") ///
40          sheetmodify firstrow(variables)
41
42  **--------------------------------------------------------------------------------
43  * Method 2: Using tabstat
44  *           Example below calculates the mean, median, min, max across 5 vars
45  **--------------------------------------------------------------------------------
46
47      use "E:\Stata help\descriptives_example.dta", clear
48
49      tabstat var1-var5, stat(mean median min max)
50
51      mrtab q3_1-q3_4 // good for dummy variables
52
53  **--------------------------------------------------------------------------------
54  * Method 3: Using postfile
55  *           Example below uses postfile to create a temporary dataset where
56  *           you decide what descriptives to store in the columns. This example
57  *           stores the mean and number of observations for a variable, as
58  *           well as the mean and number of observations of the variable by
59  *           another variable, in this case, region.
60  **--------------------------------------------------------------------------------
61
62      use "E:\Stata help\descriptives_example.dta", clear
63
64      * create a temporary dataset named "memhold"
65      tempname memhold
66
67      /* create the structure of the postfile dataset, with each column labeled
68      according to descriptives of interest */
```

```
69      postfile `memhold' str20(varname) meantot ntot mean1 n1 mean2 n2 mean3 ///
70          n3 mean4 n4 nomiss ///
71          using "E:\Stata help\descriptives_postfile.dta", replace
72
73      loc varlist var1 var2 var3 var4 var5
74
75      * loop through variables, generating means, and storing them as locals
76      foreach var of local varlist {
77
78          su `var' // summarize command
79          loc mean = r(mean) // store mean of variable as local
80          loc n = r(N) // store num obs of variable as local
81
82          forvalues i=1/4 {
83              su `var' if region == `i' // summarize variable for each region
84              loc mean`i' = r(mean) // store mean for that region
85              loc n`i' = r(N) // store num obs for that region
86          }
87              mdesc `var' // summarize missingness of variable
88              loc nomiss = r(miss) // store missingness in local
89
90      * "post" the estimates to the temporary file
91          post `memhold' ("`var'") (`n') (`mean') (`n1') (`mean1') (`n2') ///
92          (`mean2') (`n3') (`mean3') (`n4') (`mean4') (`nomiss')
93      }
94      * close the temporary file
95      postclose `memhold'
96
97      preserve
98      * open the temporary file
99          use "E:\Stata help\descriptives_postfile.dta", clear
100
101     * save the temporary file as an excel spreadsheet
102         export excel using "E:\Stata help\descriptives.xlsx", replace
103     restore
104
```