MPP-C6: Statistics II

Programming with Stata

Max Callaghan

Hertie School of Governance

?? February 2015

Outline

- Why Program?
- Reproducible Research
- Programming in Stata to accomplish difficult tasks and simplify repetitive tasks

Why Program?

- Computers can perform calculations faster and more accurately than humans
- When a task is difficult or repetitive, if often makes sense to instruct the computer to do it
- When those instructions are written down, we and others can see exactly what has been done
  - It's easier to repeat the work
  - It's easier to spot errors
  - It's easier to repeat the work, changing just one detail, without doing every subsequent step again.

Reproducible Research

- “The standard of reproducibility calls for the data and the computer code used to analyse the data to be made available to others” [**?**]
- Literate programming ties together data, code and the actual research output, enhancing reproducibility

Reproducible Research with Stata

- What do we already do that helps to keep our research reproducible?
- How do we ensure that what's in our research output reflects the calculations we report making?

Reproducible Research with Stata

- The ideal for perfect reproducibility would be to have a single document that contains instructions for performing calculations as well as for producing the research output we present.
- This is not possible in Stata without LaTeX but we can at least make the way we include Stata output in Word documents more systematic

Reproducible Research with Stata

[Include instructions to do this, with a simple example of output that changes to reflect a change in a do file]

Programming with Stata

Outline

- Stata basics
- Directory structure
- Reading data
- Transforming and processing data
- Presenting results with Stata

Stata Basics

- Pointing and clicking is fine for exploring data
- The command line is fine for trying out commands
- Anything you want to be able to reproduce, you should put in the do file

Stata Basics

Writing a good do file

A good do file should be readable by humans as well as computers.

- Use comments to explain what each line is doing
- Empty lines are free, space makes your code easier to read
- Use meaningful names when you create them, and write them consistently (variable_name, variableName or VariableName)
- Follow indentation conventions e.g.

```
forvalues i in 1/5 {
  display `i'
}
```

Stata Basics

Understanding Stata Commands

Stata commands are preprogrammed functions that take information we give to them, do something with the information, then output something. We pass information to commands with *arguments* and *options*.
If you are not sure how to use a command you can get help by typing "help" and then the name of the command. For example,

```
help regress
```

will take you to the regress command's manual

Stata Basics

Reading the Stata manual

The Syntax `regress` *depvar [indepvars] [if] [in] [weight] [,options]* describes the basic use of the command

- Pay attention to the order of the arguments. This is how Stata knows which arguments are which
- Items in square brackets are optional
- Options come after a comma. Possible options are described in the help file
- You don't always need to set a lot of options, but you should pay atention to what the defaults are

If you don't know the command you want to use, you will have try and describe your problem to google.

Directory Structure

File paths tell the computer where to read and write information. They differ between Windows and Mac/Linux. (\or /)

- Absolute paths start from the top of the tree and specify each subdirectory until the file e.g. "C:bla\bla\data.dta" (or "/bla/bla/data.dta", or even "http://bla.com/data.dta")
- Relative paths start from the current working directory; use ../ or ..\to move to the directory above the current one

Directory Structure

If you refer to more than one resource, it makes sense to set the working directory at the top of your do file and use relative paths. You'll want to think about how you structure your directory so that you can access items easily.

- If your do file produces output, think about where you want to save it so that you can access it automatically with another program
- This also allows you to change computers easily. Dropbox is an easy way to carry entire directories between computers. Git/Github is even better as it incorporates version control.

Reading Data

Data doesn't always come in nicely formatted dta files. Sometimes you have a data source or sources in files that aren't set up for stata to read. You often have to do a bit of work to get things into the format you want: the more of that work is recorded the better.

Reading Data

Some things to pay attention to when reading data

- Keep an original copy of the data exactly as you found it, if you make changes, save to a new name
- Try and make changes with Stata in your do file. If you have to change in Excel, write down what you did
- Check the data has been imported properly before you use it
  - You may need to specify what character signifies missing values in your data
  - You might need to specify the delimiter in csv or txt files
  -

Data types

Stata stores data in various different data types. Each variable can only be one data type. Some operations can only be done on data of certain types

- Numeric data can be stored in various degrees of precision: check -help data types- for more information
- Anything with non-numeric characters will be saved as a string (text)

It's easy for data to arrive in the wrong format when we read from other sources.

We can use -tostring- and -destring- to convert between string and numeric data, as well as -encode- to create a numeric variable out of non-numeric string data.

Processing Data

- -keep- and -drop- can remove observations we don't want
- -gen- and -egen- create new variables
- -replace- can change the values of a variable

All can be applied selectively with if conditions

Stata already helps us to perform calculations quickly, but we can speed up how we interact with Stata by using some simple programming to avoid repetition. The most simple concept is storing something as a macro.

- Macros can tie a name to some text
- `local controls age gender incCat` ties the word ”controls” to “age gender inc_cat”
- Now, everytime we type `` `controls' ``, stata understands “age gender inc_cat” (note the backtick, which is under the tilde)
- If we type “age gender inc_cat” a lot, then we save ourselves time by defining it once and referring to the definition the other times
- This also reduces the risk of errors. Why?

Programming with Stata

Loops

Loops can speed up our work by repeating tasks while changing one thing.

```
foreach control of local controls {
  display "`control'"
}
```

Will loop through our list of controls, and perform the command -display- on each of them.

As with macros, we initialise the iterator without quotes, and insert its value into our commands using the backtick and single quote

Programming with Stata

Loops

We can perform any commands we want inside the loop, including using more loops and if conditions. What would be the outcome of this loop?

```
forvalues i = 1/20 {
  if mod(`i',2)==0 {
    di "`i' is even"
  } else {
    di "`i' is odd"
  }
}
```

Programming with Stata

Exercise - are you smarter than a 10 year old?

Fizz-Buzz

Count up to 100, replacing any number divisible by three with the word "fizz", and any number divisible by five with the word "buzz", and any number divisible by both with "fizz-buzz".

Can you write a loop in Stata that plays fizzbuzz correctly? (2 minutes)

Programming with Stata

Exercise - are you smarter than a 10 year old?

```
forvalues i = 1/100 {
  if mod(`i',3)==0 & mod(`i',5)==0{
    di "fizzbuzz"
  }
  else if mod(`i',5)==0{
    di "buzz"
  }
  else if mod(`i',3)==0{
    di "fizz"
  }
  else {
    di "`i'"
  }
}
```

Programming with Stata

The -by- command

Another way to repeat calcuations is using -by-.

-by- temporarily splits your data into subsets for every value of a variable and performs the command on that subset.

To use the by command you need to -sort- your data

Programming with Stata

The -by- command

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. sort foreign

. by foreign: reg price mpg

-----------------------------------------------------------------------------------------------------------------------
-> foreign = Domestic

      Source |       SS           df       MS      Number of obs   =        52
-------------+----------------------------------   F(1, 50)        =     17.05
       Model |   124392956         1   124392956   Prob > F        =    0.0001
    Residual |   364801844        50  7296036.89   R-squared       =    0.2543
-------------+----------------------------------   Adj R-squared   =    0.2394
       Total |   489194801        51  9592054.92   Root MSE        =    2701.1

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -329.2551   79.74034    -4.13   0.000    -489.4183   -169.0919
       _cons |   12600.54   1624.773     7.76   0.000     9337.085    15863.99
------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------
-> foreign = Foreign

      Source |       SS           df       MS      Number of obs   =        22
-------------+----------------------------------   F(1, 20)        =     13.25
       Model |  57534941.7         1  57534941.7   Prob > F        =    0.0016
    Residual |  86828271.1        20  4341413.55   R-squared       =    0.3985
-------------+----------------------------------   Adj R-squared   =    0.3685
       Total |   144363213        21   6874438.7   Root MSE        =    2083.6

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -250.3668   68.77435    -3.64   0.002    -393.8276    -106.906
       _cons |   12586.95   1760.689     7.15   0.000     8914.217    16259.68
```

Example 1

Reading Data

Now we want to apply some of this to a real world example.

We've found an interesting data source on the web (link).

It's an excel sheet containing various crime related data in different boroughs over different time periods.

There's an interesting panel dataset in there but we have to work to get it.

Example 1

Reading Data

We start by copying the data it onto our computer and use -import excel- to read into stata the sheet "Fear of Crime-Borough".

Interactive Stata Example

```
. capture confirm file data/crime.xls

. if _rc==601 {
.         copy https://files.datapress.com/london/dataset/metropolitan-police-service-recorded-crime-figures-an
> -figures.xls ///
>                 data/crime.xls, replace
. }

. import excel data/crime.xls, ///
>         sheet("Fear of Crime-Borough") /// Tell stata which sheet to import
>         cellrange(A3:AG31) /// Specify the cells we want to import
>         firstrow // tell stata that variable names are in the first row

.
. cap rename (BarkingandDagenham HammersmithandFulham KensingtonandChelsea) ///
>    (BarkingDagenham HammersmithFulham KensingtonChelsea) // Inconsistent names

. cap rename A MonthYear // Merged cell caused problem

.
```

Example 1

Cleaning data

Sometimes data is input incorrectly - in this case we have 2 records for September 2008. Since we don't know which is correct, it's probably safer to remove them both.

Interactive Stata Example

```
. sort MonthYear

. by MonthYear: gen dup = cond(_N==1,0,_n)

. drop if dup > 0
(2 observations deleted)

.
```

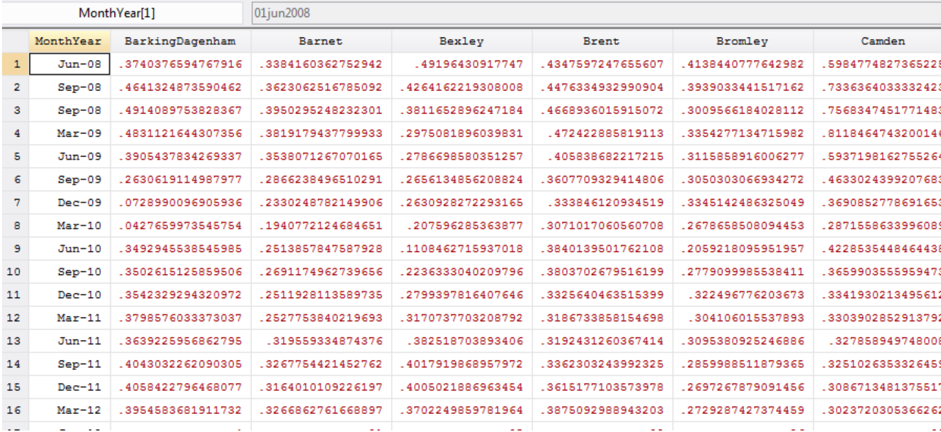
For each value of the date, we set dup to 1 if the number of observations (_N) with that value is 1, otherwise we set it to the number of each observation (_n)

Example 1

Transforming data

In stata, columns are called variables and rows are observations. We don't always receive data like that.

How would you reformat this data? What variables do we have?



MonthYear[1] 01jun2008

	MonthYear	BarkingDagenham	Barnet	Bexley	Brent	Bromley	Camden
1	Jun-08	.3740376594767916	.3384160362752942	.49196430917747	.4347597247655607	.4138440777642982	.5984774827365225
2	Sep-08	.4641324873590462	.3623062516785092	.4264162219308008	.4476334932990904	.3939033441517162	.7336364033332423
3	Sep-08	.4914089753828367	.3950295248232301	.3811652896247184	.4668936015915072	.3009566184028112	.7568347451771483
4	Mar-09	.4831121644307356	.3819179437799933	.2975081896039831	.472422885819113	.3354277134715982	.8118464743200146
5	Jun-09	.3905437834269337	.3538071267070165	.2786698580351257	.405838682217215	.3115858916006277	.5937198162755264
6	Sep-09	.2630619114987977	.2866238496510291	.2656134856208824	.3607709329414806	.3050303066934272	.4633024399207683
7	Dec-09	.0728990096905936	.2330248782149906	.2630928272293165	.333846120934519	.3345142486325049	.3690852778691653
8	Mar-10	.0427659973545754	.1940772124684651	.207596285363877	.3071017060560708	.2678658508094453	.2871558633996089
9	Jun-10	.3492945538545985	.2513857847587928	.1108462715937018	.3840139501762108	.2059218095951957	.4228535448464438
10	Sep-10	.3502615125859506	.2691174962739656	.2236333040209796	.3803702679516199	.2779099985538411	.3659903555959473
11	Dec-10	.3542329294320972	.2511928113589735	.2799397816407646	.3325640463515399	.322496776203673	.3341930213495612
12	Mar-11	.3798576033373037	.2527753840219693	.3170737703208792	.3186733858154698	.304106015537893	.3303902852913792
13	Jun-11	.3639225956862795	.319559334874376	.382518703893406	.3192431260367414	.3095380925246886	.327858949748008
14	Sep-11	.4043032262090305	.3267754421452762	.4017919868957972	.3362303243992325	.2859988511879365	.3251026353326459
15	Dec-11	.4058422796468077	.3164010109226197	.4005021886963454	.3615177103573978	.2697267879091456	.3086713481375517
16	Mar-12	.3954583681911732	.3266862761668897	.3702249859781964	.3875092988943203	.2729287427374459	.3023720305366262

Example 1

Transforming data

Our data is too wide: we can use the -reshape- command to switch between “wide” and “long” formats. First we need to give the borough coloums a common prefix, then we can tell reshape to create a new variable. While we’re at it, we can label the new variable we create

Interactive Stata Example

```
. rename(BarkingDagenham-Westminster) FoC=

. reshape long FoC, i(MonthYear) j(Borough) string
(note: j = BarkingDagenham Barnet Bexley Brent Bromley Camden Croydon Ealing Enfield Greenwich Hackney Hammersm
> unslow Islington KensingtonChelsea KingstonuponThames Lambeth Lewisham Merton Newham Redbridge RichmonduponTh
>  Wandsworth Westminster)

Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                       26   ->     832
Number of variables                  34   ->       4
j variable (32 values)                    ->   Borough
xij variables:
FoCBarkingDagenham FoCBarnet ... FoCWestminster->FoC
-----------------------------------------------------------------------------

. label variable FoC "Fear of Crime-Borough"

.
```

Example 1

Using a loop

Now we know how to read in one sheet, we need to do the same for the others. We could write the below 7 times (=77 lines) and change all the relevant parts...

```
import excel data/crime.xls, ///
  sheet("Fear of Crime-Borough") /// Tell stata which sheet to import
  cellrange(A3:AG31) /// Specify the cells we want to import
  firstrow // tell stata that variable names are in the first row

cap rename BarkingandDagenham BarkingDagenham // Inconsistent name
cap rename A MonthYear // Merged cell caused problem

rename(BarkingDagenham-Westminster) foCrime=
keep if MonthYear > td(2sep2008) | MonthYear < td(2aug2008) // There were 2 values
reshape long foCrime, i(MonthYear) j(Borough) string
label variable FoC "Fear of Crime-Borough"
save foCrime.dta
```

Example 1

Reading Data

Instead, we can write 3 lists, for the 3 parts of our code that change (Sheet name, cell range, variable name) and write a loop (20 lines).

```
local sheets `" "Fear of Crime-Borough" "MOPAC Priority-Borough" "Officer Strength-Borough" "Sergeant Strength-
local cranges A3:AG31 A3:AH58 A5:AG97 A4:AG36 A5:AG97 A5:AG97 A5:AG97
local varnames FoC MOPAC OffStrength SgntStrength SpclStrength PCSOStrength StaffStrength
local N : word count `sheets'

forvalues i = 1/`N' {
  local sheet : word `i' of `sheets'
  local crange : word `i' of `cranges'
  local varname : word `i' of `varnames'
  clear
  import excel data/crime.xls, ///
    sheet("`sheet'") ///
    cellrange("`crange'") ///
    firstrow
  cap rename BarkingandDagenham BarkingDagenham // Inconsistent name
  cap rename A MonthYear // Merged cell caused problem
  sort MonthYear
  by MonthYear: gen dup = cond(_N==1,0,_n)
  drop if dup > 0
  rename (BarkingDagenham-Westminster) `varname'=
  reshape long "`varname'", i(MonthYear) j(Borough) string
  save data/`varname'.dta, replace
}
```

Example 1

Using a loop

Now we can use the same list to loop through the saved datasets and merge them with each other

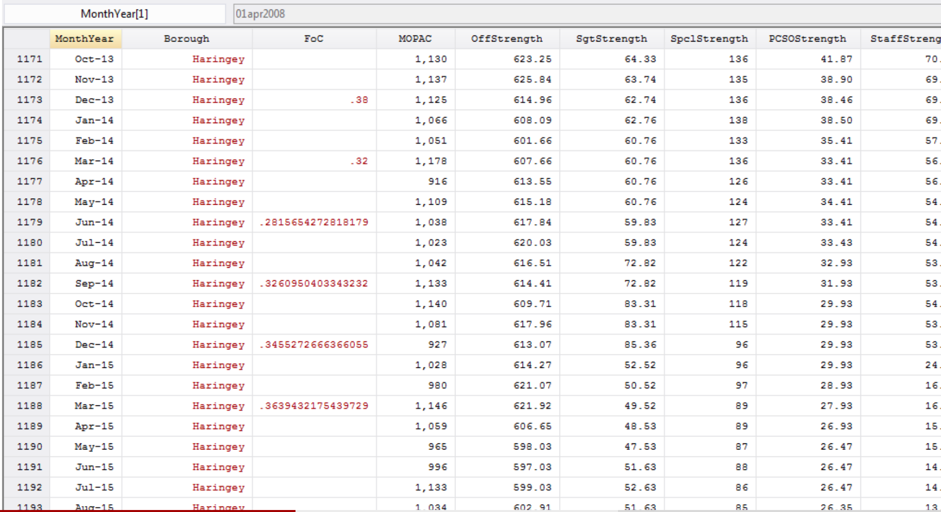
Interactive Stata Example

```
. clear

. forvalues i = 1/`N' {
  2.    local varname : word `i' of `varnames'
  3.    if `i'==1 {
  4.      use data/`varname'
  5.    }
  6.    else {
  7.      merge m:m MonthYear Borough using data/`varname'
  8.          drop _merge
  9.    }
 10. }

    Result                           # of obs.
    -----------------------------------------
    not matched                         1,568
        from master                       320  (_merge==1)
        from using                      1,248  (_merge==2)

    matched                               512  (_merge==3)
    -----------------------------------------

    Result                           # of obs.
    -----------------------------------------
    not matched                           864
        from master                         0  (_merge==1)
        from using                        864  (_merge==2)

    matched                             2,080  (_merge==3)
    -----------------------------------------

    Result                           # of obs.
    -----------------------------------------
    not matched                         1,920
```

Example 1

Cleaning Data

We now have all of our data in one place, but it hasn't been imported correctly



MonthYear[1] 01apr2008

	MonthYear	Borough	FoC	MOPAC	OffStrength	SgtStrength	SpclStrength	PCSOStrength	StaffStreng
1171	Oct-13	Haringey		1,130	623.25	64.33	136	41.87	70.
1172	Nov-13	Haringey		1,137	625.84	63.74	135	38.90	69.
1173	Dec-13	Haringey	.38	1,125	614.96	62.74	136	38.46	69.
1174	Jan-14	Haringey		1,066	608.09	62.76	138	38.50	69.
1175	Feb-14	Haringey		1,051	601.66	60.76	133	35.41	57.
1176	Mar-14	Haringey	.32	1,178	607.66	60.76	136	33.41	56.
1177	Apr-14	Haringey		916	613.55	60.76	126	33.41	56.
1178	May-14	Haringey		1,109	615.18	60.76	124	34.41	54.
1179	Jun-14	Haringey	.2815654272818179	1,038	617.84	59.83	127	33.41	54.
1180	Jul-14	Haringey		1,023	620.03	59.83	124	33.43	54.
1181	Aug-14	Haringey		1,042	616.51	72.82	122	32.93	53.
1182	Sep-14	Haringey	.3260950403343232	1,133	614.41	72.82	119	31.93	53.
1183	Oct-14	Haringey		1,140	609.71	83.31	118	29.93	54.
1184	Nov-14	Haringey		1,081	617.96	83.31	115	29.93	53.
1185	Dec-14	Haringey	.3455272666366055	927	613.07	85.36	96	29.93	53.
1186	Jan-15	Haringey		1,028	614.27	52.52	96	29.93	24.
1187	Feb-15	Haringey		980	621.07	50.52	97	28.93	16.
1188	Mar-15	Haringey	.3639432175439729	1,146	621.92	49.52	89	27.93	16.
1189	Apr-15	Haringey		1,059	606.65	48.53	89	26.93	15.
1190	May-15	Haringey		965	598.03	47.53	87	26.47	15.
1191	Jun-15	Haringey		996	597.03	51.63	88	26.47	14.
1192	Jul-15	Haringey		1,133	599.03	52.63	86	26.47	14.
1193	Aug-15	Haringey		1,034	602.91	51.63	85	26.35	13.

Example 1

Cleaning Data

The reason why Stata thinks our fear of crime variable is a string is that some of the values have % characters (probably an artefact of inconsistent Excel cell formatting).

We can remove these using the -subinstr- command

Interactive Stata Example

```
. gen FoC2 = subinstr(FoC,"%","",.)
(2,112 missing values generated)

. destring FoC2, replace
FoC2 has all characters numeric; replaced as double
(2112 missing values generated)

. replace FoC2 = FoC2/100 if FoC2 > 1
(128 real changes made)

.
```

Example 1

Cleaning Data

We also need to encode a numerical version of our Borough variable in order to finish setting up the data

Interactive Stata Example

```
. encode Borough, generate(nBorough)

. xtset nBorough MonthYear
       panel variable:  nBorough (strongly balanced)
        time variable:  MonthYear, Apr-08 to Nov-15, but with gaps
                delta:  1 day

.
```

Presenting Results with Stata

Tables

The -estout- program is a good way to present professional looking regression tables.

You usually want to present the results of several models side by side

- After each regression, save the results of the model using -eststo-
- Create a table using -esttab-
- You can save the table as an rtf file, and include a link to it in a word document that can be refreshed if you change your analysis

Presenting Results with Stata

Regression results

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. eststo A: quietly reg price mpg

. eststo B: quietly reg price mpg rep78

. esttab A B using ../word/reg_table_1.rtf, replace
(output written to ../word/reg_table_1.rtf)

.
```

	(1) price	(2) price
mpg	-238.9***	-271.6***
	(-4.50)	(-4.70)
rep78		667.0
		(1.95)
_cons	11253.1***	9657.8***
	(9.61)	(7.17)
N	74	69

t statistics in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Presenting Results with Stata

Regression results

After running a regression, you can access post-estimation statistics with -ereturn-.

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. quietly reg price mpg rep78

. ereturn list

scalars:
                  e(N) =  69
               e(df_m) =  2
               e(df_r) =  66
                  e(F) =  11.05650420530028
                 e(r2) =  .250961904598189
               e(rmse) =  2558.53561189829
                e(mss) =  144754063.3643494
                e(rss) =  432042895.5052159
               e(r2_a) =  .2282637804951038
                 e(ll) =  -637.8293069393488
               e(ll_0) =  -647.7986144493904
               e(rank) =  3

macros:
            e(cmdline) : "regress price mpg rep78"
              e(title) : "Linear regression"
         e(marginsok) : "XB default"
                e(vce) : "ols"
             e(depvar) : "price"
                e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
              e(model) : "ols"
          e(estat_cmd) : "regress_estat"

matrices:
```

Presenting Results with Stata

Regression results

You can access these and use them just like any other number with e([statistic])

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. reg price mpg rep78

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(2, 66)        =     11.06
       Model |   144754063         2  72377031.7   Prob > F        =    0.0001
    Residual |   432042896        66  6546104.48   R-squared       =    0.2510
-------------+----------------------------------   Adj R-squared   =    0.2283
       Total |   576796959        68  8482308.22   Root MSE        =    2558.5

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -271.6425   57.77115    -4.70   0.000    -386.9864   -156.2987
       rep78 |   666.9568   342.3559     1.95   0.056     -16.5789    1350.492
       _cons |   9657.754    1346.54     7.17   0.000       6969.3    12346.21
------------------------------------------------------------------------------

. di e(r2)
.2509619

.
```

Presenting Results with Stata

Regression results

You can also view a matrix of the coefficients

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. reg price mpg rep78

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(2, 66)        =     11.06
       Model |   144754063         2  72377031.7   Prob > F        =    0.0001
    Residual |   432042896        66  6546104.48   R-squared       =    0.2510
-------------+----------------------------------   Adj R-squared   =    0.2283
       Total |   576796959        68  8482308.22   Root MSE        =    2558.5

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -271.6425   57.77115    -4.70   0.000    -386.9864   -156.2987
       rep78 |   666.9568   342.3559     1.95   0.056    -16.5789    1350.492
       _cons |   9657.754    1346.54     7.17   0.000      6969.3    12346.21
------------------------------------------------------------------------------

. matrix list e(b)

e(b)[1,3]
           mpg        rep78        _cons
y1  -271.64254    666.95676    9657.7544

.
```

Presenting Results with Stata

Regression results

And the variance covariance matrix

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. reg price mpg rep78

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(2, 66)        =     11.06
       Model |   144754063         2  72377031.7   Prob > F        =    0.0001
    Residual |   432042896        66  6546104.48   R-squared       =    0.2510
-------------+----------------------------------   Adj R-squared   =    0.2283
       Total |   576796959        68  8482308.22   Root MSE        =    2558.5

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -271.6425   57.77115    -4.70   0.000    -386.9864   -156.2987
       rep78 |   666.9568   342.3559     1.95   0.056    -16.5789    1350.492
       _cons |   9657.754    1346.54     7.17   0.000      6969.3    12346.21
------------------------------------------------------------------------------

. matrix list e(V)

symmetric e(V)[3,3]
              mpg        rep78        _cons
  mpg   3337.5061
rep78  -7957.6075    117207.58
_cons  -43953.025   -229768.93      1813171

.
```

Presenting Results with Stata

Regression results

As well as individual coefficients and standard errors

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. reg price mpg rep78

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(2, 66)        =     11.06
       Model |   144754063         2  72377031.7   Prob > F        =    0.0001
    Residual |   432042896        66  6546104.48   R-squared       =    0.2510
-------------+----------------------------------   Adj R-squared   =    0.2283
       Total |   576796959        68  8482308.22   Root MSE        =    2558.5

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -271.6425   57.77115    -4.70   0.000    -386.9864   -156.2987
       rep78 |   666.9568   342.3559     1.95   0.056    -16.5789    1350.492
       _cons |   9657.754    1346.54     7.17   0.000      6969.3    12346.21
------------------------------------------------------------------------------

. di _b[mpg]
-271.64254

. di _se[mpg]
57.771153

.
```

Presenting Results with Stata

Regression results

This is useful because we can specify the statistics we want to include in our regression table with the stats option.

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. eststo A: quietly reg price mpg

. eststo B: quietly reg price mpg rep78

. esttab A B using ../word/reg_table_2.rtf, ///
>          stats(N r2 F) ///
>          replace
(output written to ../word/reg_table_2.rtf)

.
```

	(1)	(2)
	price	price
mpg	-238.9***	-271.6***
	(-4.50)	(-4.70)
rep78		667.0
		(1.95)
_cons	11253.1***	9657.8***
	(9.61)	(7.17)
N	74	69
r2	0.220	0.251
F	20.26	11.06

t statistics in parentheses

$^{*}\ p < 0.05$, $^{**}\ p < 0.01$, $^{***}\ p < 0.001$

Presenting Results with Stata

Regression results

We can also calculate our own statistics, and include them in the model. Here we compute the turning point for a squared term

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. cap gen sqweight = weight^2

. eststo A: quietly reg price weight sqweight

. estadd scalar tp = -_b[weight]/(2*_b[sqweight])

added scalar:
                  e(tp) =  2401.6552

. eststo B: quietly reg price weight sqweight mpg

. estadd scalar tp = -_b[weight]/(2*_b[sqweight])

added scalar:
                  e(tp) =  2691.2649

. esttab A B using ../word/reg_table_tp.rtf, ///
>          stats(tp) ///
>          replace
(output written to ../word/reg_table_tp.rtf)

.
```

	(1)	(2)
	price	price
weight	-7.273**	-9.040**
	(-2.70)	(-3.11)
sqweight	0.00151***	0.00168***
	(3.49)	(3.79)
mpg		-124.8
		(-1.53)
_cons	13418.8**	19804.8***
	(3.36)	(3.44)
tp	2401.7	2691.3

t statistics in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Presenting Results with Stata

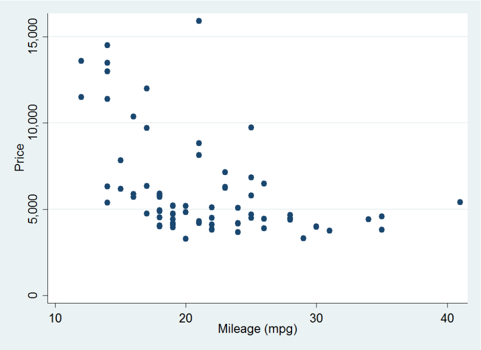
Graphs

-graph twoway- has many different ways to show relationships between variables

Here we show a basic scatter plot

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. twoway scatter price mpg

. graph export ../word/scatter.png, replace
(file ../word/scatter.png written in PNG format)

.
```



After saving, we can insert a picture as link into word

Presenting Results with Stata

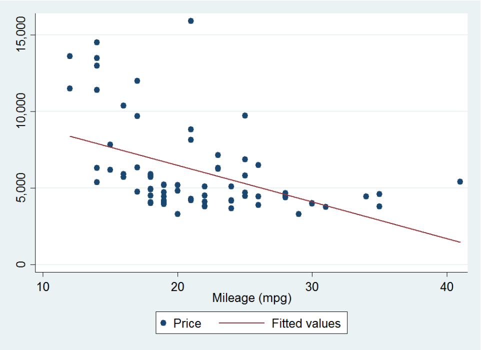
Graphs

-twoway- is extendable - we can overlay lots of plots one on top of the other

We just add each plot in a set of brackets

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. twoway (scatter price mpg) (lfit price mpg)

. graph export ../word/overlay.png, replace
(file ../word/overlay.png written in PNG format)

.
```



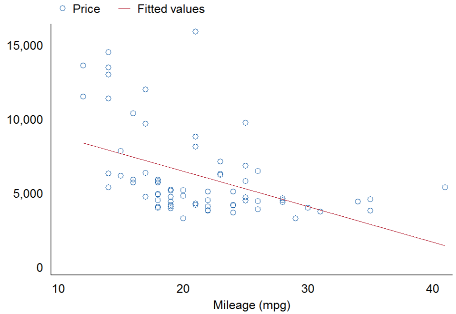
Presenting Results with Stata

Graphs

Various schemes are available - a lighter option than built schemes can be downloaded with -ssc install burd-

Interactive Stata Example

```
. graph query, schemes

Available schemes are

    s2color        see help scheme_s2color
    s2mono         see help scheme_s2mono
    s2manual       see help scheme_s2manual
    s2gmanual      see help scheme_s2gmanual
    s2gcolor       see help scheme_s2gcolor
    s1color        see help scheme_s1color
    s1mono         see help scheme_s1mono
    s1rcolor       see help scheme_s1rcolor
    s1manual       see help scheme_s1manual
    sj             see help scheme_sj
    economist      see help scheme_economist
    s2color8       see help scheme_s2color8
    burd           see help scheme_burd
    burd10
    burd11
    burd3
    burd4
    burd5
    burd6
    burd7
    burd8
    burd9

. set scheme burd

. sysuse auto
(1978 Automobile Data)

. twoway (scatter price mpg) (lfit price mpg)
```



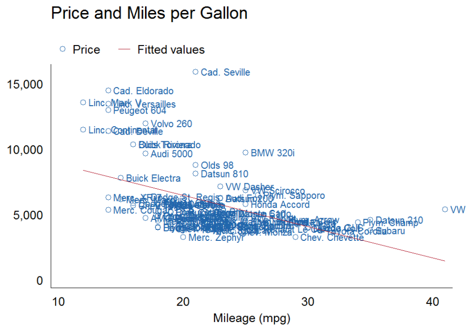
Presenting Results with Stata

Graphs

There are many more options we can set for the graph. They can be applied universally, or to individual layers

Interactive Stata Example

```
. sysuse auto
(1978 Automobile Data)

. twoway (scatter price mpg, mlabel(make)) ///
>         (lfit price mpg), ///
>         title("Price and Miles per Gallon")

.
. graph export ../word/graph_options.png, replace
(file ../word/graph_options.png written in PNG format)

.
```



Presenting Results with Stata

Graphs

-graph matrix- draws a scatterplot matrix showing the relationship between pairs of variables in a list

You can apply it to all numeric variables

-

-graph matrix-

-

References