Programming with Stata

IAP Skills Workshops

Raymond Kluender and Benjamin Marx

January 26, 2016

Thanks and References

- Special thanks to Sally Hudson for letting us borrow and build on her slides from two years ago for this year's IAP workshop
- Hunt, Andrew and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley: New York (2000).
- Gentzkow, Matthew and Jesse M. Shapiro. "RA Manual: Notes on Writing Code". Chicago Booth Manuscript (June 25, 2012).
- Social Science Computing Cooperative. "Working with Groups." *University of Wisconsin-Madison*. Find more at: `https://www.ssc.wisc.edu/sscc/pubs/sfr-groups.htm`

I'd highly recommend reading the Gentzkow and Shapiro RA manual to inform your own practices, you can find it at the link here:
`http://web.stanford.edu/~gentzkow/research/CodeAndData.pdf`
Many of the habits we'll advocate here are grounded in their philosophy.

Thanks for coming!

It's pretty obvious to us that we do not have a monopoly on Stata knowledge in this room (and are almost certainly less knowledgeable than a number of you...). We also know that there are highly varying levels of Stata experience in the room.

A few requests for the talk:

- If you think our advice is misguided or there's a better way to do something, speak up!
- If we are confusing, ask questions or let us know!

This workshop is entirely drawn from our own personal (self-taught) experiences with Stata, and our goal was to collect solutions to the biggest headaches we've run into in our combined ∼10 years of programming with Stata and to share our favorite tips/tricks.

Features of Good Data Work

Managing a big data project is hard! Here are some principles that can make it easier.

Practices that are...	make it easy to...
flexible	adapt as the project evolves.
divisible	share work with a team.
transparent	correct mistakes when they happen.
well-documented	maintain the project over time.

Workshop Outline

- Principles for well-documented and replicable coding
- Avoiding mistakes
- Working with groups
- Assorted useful commands
- Presentation: Making professional tables and figures

Principles for Clean Coding

General Principles

- Keep clean directories
- Know your data (there is no shame in using `browse`!)
- Comment everything you do
- Don't hard code
- Insert breaks in the code
- Version control (Come to Michael's Git Workshop tomorrow!)

Directories

When you start a project, you want to create directories according to the following rules

- Separate directories by function
- Separate files into inputs and outputs
- Make directories reflect the workflow of the project

As an example, here's the directory for a current project I'm working on with the Health and Retirement Survey:

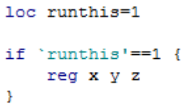
- `HRS/Build`
  - `/input` (Includes the raw data extract)
  - `/intermediate` (Could include intermediate data files)
  - `/output` (Includes the completed clean data file)
  - `/programs` (Includes the relevant programs to build the analysis data file)
- `HRS/Analyze`
  - `/input` (Linked to the output folder of build)
  - `/output` (Includes the tables and figures from your analysis output)
  - `/programs` (Includes the relevant programs to analyze your data)

Commenting and Spacing

- Skip lines (makes it easier to comment!)
- Indent loops (this is not Python!)
- Outline/divide up your do-file like a paper
- Use `#delimit;` for long commands example

Writing efficient code

- Write loops as much as you can: `foreach`, `forvalues`, `while`, etc.
    - Side note: if you don't know it, `levelsof` will change your life
      e.g.
      ```
      levelsof age, local(ages)
      foreach a of local ages {
       reg y x z if age==‘a’
      }
      ```
- Store constants as macros at the top of your dofile
- Shorten running time using switches (`if` statements)



```
loc runthis=1

if `runthis'==1 {
    reg x y z
}
```

- Create `tempfiles` instead of .dta files
    - ... unless you badly need to troubleshoot

Shell files and 'include'

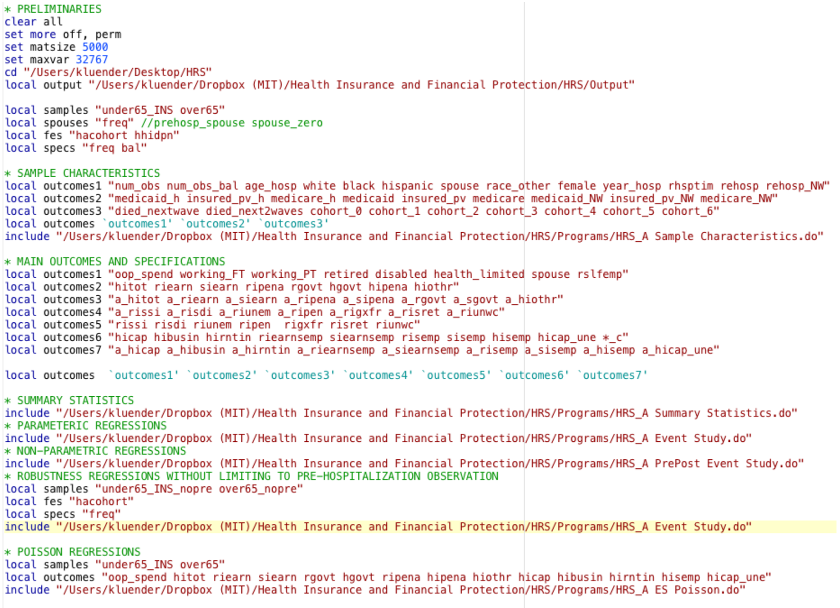
Super common situation: Your advisor recommends you change the sample definition, add a variable, drop a year of data, etc. and you need to re-run all of your analysis.

Do you:

- 1. Re-run your data preparation, open every analysis file and run them individually?
- 2. Make the change, open your shell file, and press "Do"?
  - The command `include` will call the individual analysis programs

Additional benefit: If your programs take a long time to run, this can automate them to run in succession.

Shell files and ‘include’



```
* PRELIMINARIES
clear all
set more off, perm
set matsize 5000
set maxvar 32767
cd "/Users/kluender/Desktop/HRS"
local output "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Output"

local samples "under65_INS over65"
local spouses "freq" //prehosp_spouse spouse_zero
local fes "hacohort hhidpn"
local specs "freq bal"

* SAMPLE CHARACTERISTICS
local outcomes1 "num_obs num_obs_bal age_hosp white black hispanic spouse race_other female year_hosp rhsptim rehosp rehosp_NW"
local outcomes2 "medicaid_h insured_pv_h medicare_h medicaid insured_pv medicare medicaid_NW insured_pv_NW medicare_NW"
local outcomes3 "died_nextwave died_next2waves cohort_0 cohort_1 cohort_2 cohort_3 cohort_4 cohort_5 cohort_6"
local outcomes `outcomes1' `outcomes2' `outcomes3'
include "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Programs/HRS_A Sample Characteristics.do"

* MAIN OUTCOMES AND SPECIFICATIONS
local outcomes1 "oop_spend working_FT working_PT retired disabled health_limited spouse rslfemp"
local outcomes2 "hitot riearn siearn ripena rgovt hgovt hipena hiothr"
local outcomes3 "a_hitot a_riearn a_siearn a_ripena a_sipena a_rgovt a_sgovt a_hiothr"
local outcomes4 "a_rissi a_risdi a_riunem a_ripen a_rigxfr a_risret a_riunwc"
local outcomes5 "rissi risdi riunem ripen  rigxfr risret riunwc"
local outcomes6 "hicap hibusin hirntin riearnsemp siearnsemp risemp sisemp hisemp hicap_une *_c"
local outcomes7 "a_hicap a_hibusin a_hirntin a_riearnsemp a_siearnsemp a_risemp a_sisemp a_hisemp a_hicap_une"

local outcomes  `outcomes1' `outcomes2' `outcomes3' `outcomes4' `outcomes5' `outcomes6' `outcomes7'

* SUMMARY STATISTICS
include "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Programs/HRS_A Summary Statistics.do"
* PARAMETERIC REGRESSIONS
include "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Programs/HRS_A Event Study.do"
* NON-PARAMETRIC REGRESSIONS
include "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Programs/HRS_A PrePost Event Study.do"
* ROBUSTNESS REGRESSIONS WITHOUT LIMITING TO PRE-HOSPITALIZATION OBSERVATION
local samples "under65_INS_nopre over65_nopre"
local fes "hacohort"
local specs "freq"
include "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Programs/HRS_A Event Study.do"

* POISSON REGRESSIONS
local samples "under65_INS over65"
local outcomes "oop_spend hitot riearn siearn rgovt hgovt ripena hipena hiothr hicap hibusin hirntin hisemp hicap_une"
include "/Users/kluender/Dropbox (MIT)/Health Insurance and Financial Protection/HRS/Programs/HRS_A ES Poisson.do"
```

Avoiding Mistakes

Avoiding Mistakes: General Advice

- Always know the unique identifier in your data
  - `isid` or `isid, missok`
  - use `duplicates tag, gen()` to fix unique id issues
- Assert what should be the truth
  - e.g. `assert _N==800` or `ass var1>0 & var1<.`

Avoiding Mistakes: Handling Missing Values

A large number of mistakes are generated from missing values, some important things to remember:

- Stata treats missing values (".") as ∞
- Be really careful using creating and referencing dummies:
  - `gen medicareeligible=(age>65)`
    - This will mistakenly categorize anyone with missing age as medicare eligible
    - Use `gen medicareeligible=(age>65) if age!=.`
    - This is different from `gen medicareeligible=(age>65 & age<.)` !
  - `reg y x z if medicareeligible`
    - Even if you're smart and you re-code your medicareeligible variable to missing for those with missing ages, this will still include the missing in the regression!
    - Use `if medicareeligible==1` instead

Avoiding Mistakes: Don't be lazy

- Never use `,force` option (e.g. `duplicates drop, force`)
  - If you don't know why you need to force: that's a problem
  - If you think you know why you need to force: prove it to yourself and fix it
- Beware of wildcards and empty macros (e.g. `` `controls' ``)
  - `foreach var of varlist b1* ....`
  - `loc controls x y z ....`
- Don't hard code anything: If you need to input an elasticity of "2", then set a local at the top (`local elasticity = 2`) and refer to `` `elasticity' `` instead
- Avoid using `capture`, unless you're following it up with
  - `` if _rc==1 {di ``Variable missing"} ``

Avoiding Mistakes: Do be lazy

- Do be lazy when you can have the data fill your parameters in for you. E.g. Censoring at the 99th percentile:
  - `sum income, det`
  - `local topcode = r(99)`
  - `replace income = ‘topcode’ if income>‘topcode’ & /// !missing(income)`
- Never forget to include a caveat for `!missing` when using the >

Avoiding Mistakes: Merging errors

- Use `merge 1:1` or `merge 1:m` or `merge m:1` syntax (`isid` comes in handy).
- Stay away from `m:m` merges! Why?
- (Side note: m:m issues sneakily arise with other merging commands, e.g. string match using `reclink`)
- Assert results of the merge whenever you can
    - `,assert(match master)` equivalent to `,assert(1 3)` as well as `ass _m==1 | _m==3`
- Beware of overlapping variable names and labels
    - By default, variable values in the master dataset are retained over conflicting values in the using dataset
    - Use `update` (update only missing values) and `replace` (replace conflicting values) options as needed. `match_update` (`_m==4`) and `match_conflict` (`_m==5`) provide merge results for overlapping variables

Working with Groups and Panel Data

Working with Groups and Panel Data

Often you'll have multiple observations with the same identifier (e.g. multiple individuals in a household, multiple years for an individual in panel data). There are a few commands that are really useful for working within an identifier:

- First, make sure the data is at the level we think it is: `isid hhidpn wave`
- Next, `sort hhidpn wave` if `hhidpn` is our individual identifier and we have a number of survey waves for the individual.
- `by hhidpn: egen firstwave = min(wave)`
- `by hhidpn: gen initialearnings = earnings[1]`
- `by hhidpn: gen hospnextwave = hospitalized[_n+1]`
- Check that all observations are the same within an individual
    - `sort hhidpn cohort`
    - `by hhidpn: assert cohort[1]==cohort[_N]`
- Generate the total number of hospitalizations:
    - `egen totalhosps = total(numhosps), by(hhidpn)`

Working with Groups and Panel Data, Examples

Here I am coding up the timing for an event study based around a hospitalization:

```
* Figure out which survey wave an individual is first observed in
by hhidpn: egen first_wave = min(wave)
tab first_wave, mi

* Code up first wave which references a prior hospitalization
gen wave_hosp = wave if rhosp==1
by hhidpn: egen first_hosp = min(wave_hosp)
tab first_wave first_hosp, row mi

* Want to set event time to 0 at the time of the individual's FIRST hospitalization
gen evt_time = 0 if wave==first_hosp
    replace evt_time = wave - first_hosp

* Code up age of hospitalization so we can use that to select sample
gen temp = ragey_b - 1 if rhosp==1
by hhidpn: egen age_hosp = min(temp)
drop temp

* Define re-hospitalization rates:
gen rehosp = rhsptim>1 & !missing(rhsptim) if evt_time==0
by hhidpn: gen rehosp_NW = rhosp[_n+1]
```

Working with Groups and Panel Data, Examples

Here I am examining household capital income at each survey wave of the Health and Retirement Survey:

```
. table wave, c(mean hicap sd hicap max hicap p99 hicap)

----------------------------------------------------------------------
     wave | mean(hicap)     sd(hicap)    max(hicap)    p99(hicap)
----------+-----------------------------------------------------------
        1 | 5560.100308      21278.18        515500         87170
        2 | 11597.36638      59722.31   3215064.296        180000
        3 | 16639.11386      60213.27   2944234.157   242728.5944
        4 | 13740.93966       91362.7       7797767   181315.3303
        5 | 14376.11886      56455.38       3365000   203635.4858
        6 | 12854.58494      74401.05       7334785        195000
        7 | 13805.16951      65099.76       3536642        205400
        8 | 18116.09098        294345      25360250   208012.9248
        9 | 15851.88176      65736.25       3000480   249313.6396
       10 | 12228.16818      57416.18       3001280        193000
       11 | 13015.03242       67510.7       3663000        200200
----------------------------------------------------------------------
```

Presentation

Creating TeX tables with `esttab`

- There are many ways to export nice-looking tables to TeX or Excel
- Today: `esttab`
  - Simple & (reasonably) intuitive
  - Can fully automate formatting
- How `esttab` works
  - Create a shell TeX file that will host all your tables
  - Create TeX files for each regression table using `eststo`/`esttab`
  - Customize tables and regression output inside Stata

How esttab tables look

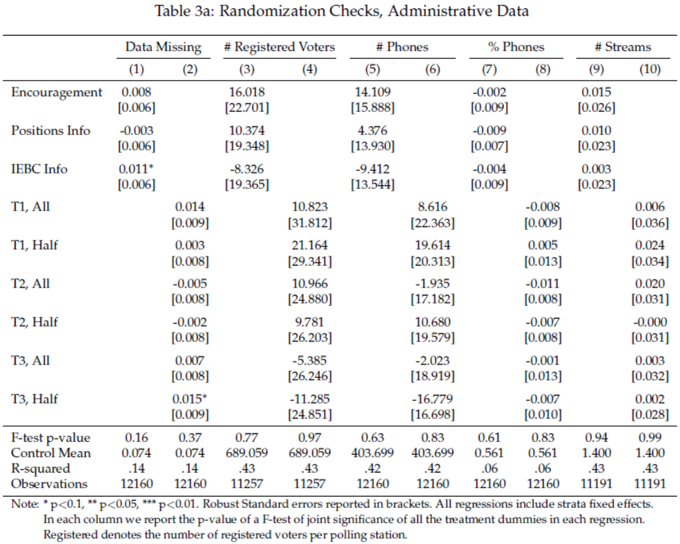


Table 3a: Randomization Checks, Administrative Data

	Data Missing		# Registered Voters		# Phones		% Phones		# Streams	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Encouragement	0.008		16.018		14.109		-0.002		0.015	
	[0.006]		[22.701]		[15.888]		[0.009]		[0.026]	
Positions Info	-0.003		10.374		4.376		-0.009		0.010	
	[0.006]		[19.348]		[13.930]		[0.007]		[0.023]	
IEBC Info	0.011*		-8.326		-9.412		-0.004		0.003	
	[0.006]		[19.365]		[13.544]		[0.009]		[0.023]	
T1, All		0.014		10.823		8.616		-0.008		0.006
		[0.009]		[31.812]		[22.363]		[0.009]		[0.036]
T1, Half		0.003		21.164		19.614		0.005		0.024
		[0.008]		[29.341]		[20.313]		[0.013]		[0.034]
T2, All		-0.005		10.966		-1.935		-0.011		0.020
		[0.008]		[24.880]		[17.182]		[0.008]		[0.031]
T2, Half		-0.002		9.781		10.680		-0.007		-0.000
		[0.008]		[26.203]		[19.579]		[0.008]		[0.031]
T3, All		0.007		-5.385		-2.023		-0.001		0.003
		[0.008]		[26.246]		[18.919]		[0.013]		[0.032]
T3, Half		0.015*		-11.285		-16.779		-0.007		0.002
		[0.009]		[24.851]		[16.698]		[0.010]		[0.028]
F-test p-value	0.16	0.37	0.77	0.97	0.63	0.83	0.61	0.83	0.94	0.99
Control Mean	0.074	0.074	689.059	689.059	403.699	403.699	0.561	0.561	1.400	1.400
R-squared	.14	.14	.43	.43	.42	.42	.06	.06	.43	.43
Observations	12160	12160	11257	11257	12160	12160	12160	12160	11191	11191

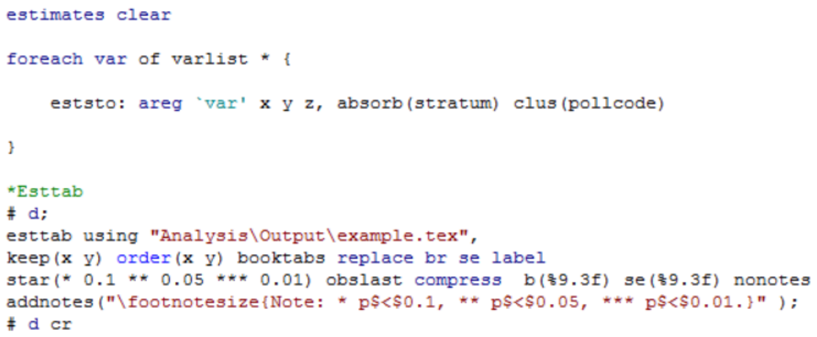
Note: * p<0.1, ** p<0.05, *** p<0.01. Robust Standard errors reported in brackets. All regressions include strata fixed effects. In each column we report the p-value of a F-test of joint significance of all the treatment dummies in each regression. Registered denotes the number of registered voters per polling station.

How esttab tables look

	(1) Completed	(2) Completed	(3) Duration	(4) Duration
Staff Ethnic Match	0.066***	0.096***	0.813***	0.763***
	[0.023]	[0.029]	[0.212]	[0.262]
Random Visit Order		-0.005***		-0.036***
		[0.002]		[0.010]
IEBC Treatment		-0.048**		0.040
		[0.018]		[0.170]
Clustering by Team-Week	X	X	X	X
Controls		X		X
Dep Var Mean	0.413	0.413	1.788	1.788
Observations	30936	30886	29377	29327

How esttab works

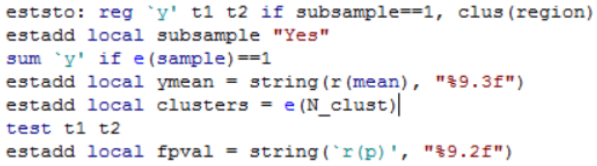
General structure:



```
estimates clear

foreach var of varlist * {

    eststo: areg `var' x y z, absorb(stratum) clus(pollcode)

}

*Esttab
# d;
esttab using "Analysis\Output\example.tex",
keep(x y) order(x y) booktabs replace br se label
star(* 0.1 ** 0.05 *** 0.01) obslast compress  b(%9.3f) se(%9.3f) nonotes
addnotes("\footnotesize{Note: * p$<$0.1, ** p$<$0.05, *** p$<$0.01.}" );
# d cr
```

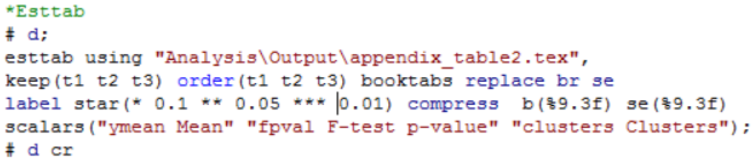
How esttab works

Customizing regression output:



```
eststo: reg `y' t1 t2 if subsample==1, clus(region)
estadd local subsample "Yes"
sum `y' if e(sample)==1
estadd local ymean = string(r(mean), "%9.3f")
estadd local clusters = e(N_clust)
test t1 t2
estadd local fpval = string(`r(p)', "%9.2f")
```

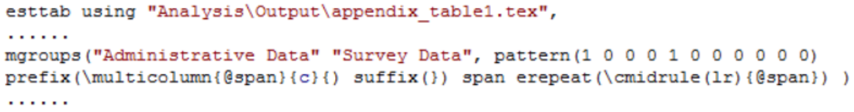
Then:



```
*Esttab
# d;
esttab using "Analysis\Output\appendix_table2.tex",
keep(t1 t2 t3) order(t1 t2 t3) booktabs replace br se
label star(* 0.1 ** 0.05 *** 0.01) compress  b(%9.3f) se(%9.3f)
scalars("ymean Mean" "fpval F-test p-value" "clusters Clusters");
# d cr
```

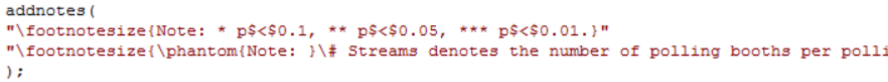
Making tables pretty

- What to include: `keep`, `order`
- Column headings: `mtitles`, `mgroups`



```
esttab using "Analysis\Output\appendix_table1.tex",
......
mgroups("Administrative Data" "Survey Data", pattern(1 0 0 0 1 0 0 0 0 0 0)
prefix(\multicolumn{@span}{c}{) suffix(}) span erepeat(\cmidrule(lr){@span}) )
......
```

- Locals: add as `string(‘r(p)’, "%9.2f")`
- Footnotes: `addnotes`



```
addnotes(
"\footnotesize{Note: * p$<$0.1, ** p$<$0.05, *** p$<$0.01.}"
"\footnotesize{\phantom{Note: }\# Streams denotes the number of polling booths per polli
);
```

- `booktabs`

Other Useful Commands

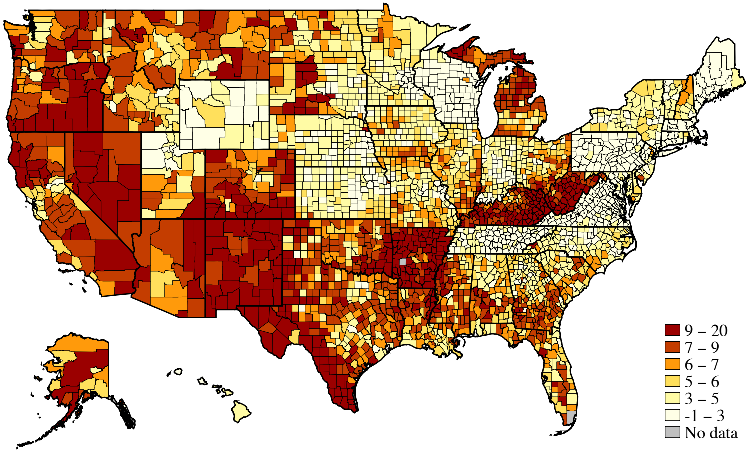
Other Useful Commands

- `saveold` and `use13`
- `egen` (`group`, `cut`, `row*`) and `egenmore`
- `expand`
- GIS stuff: `geodist`, `shp2dta`, `maptile`
- String functions: `regexm`, `subinstr`, etc.
  - For example, `gen eligible=(regexm(status,"Retired")==1)`
  - Use `trim`, `itrim`, `lower`/`proper`/`upper` to standardize strings
  - `strip, parse(....)` can also come in handy
- String matching: `reclink`
  - `reclink parish village using "$Uganda files\list.dta", idmaster(id1) idusing(id2) gen(score)`
  - `keep if score<. & score>0.9`
  - `drop _m`

Maptile

You want to create a map, but you don't want to use GIS (who does?). Try `maptile`!

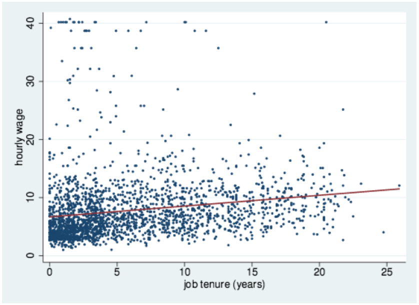
```
maptile changeinsured, geo(county)
```

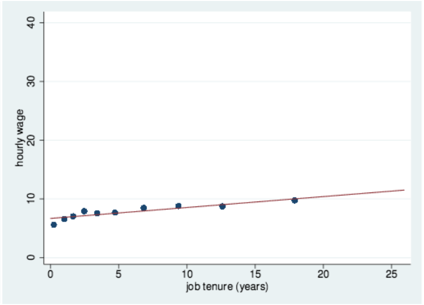


Binscatter

Want to communicate your results like Raj Chetty? Try `binscatter`!

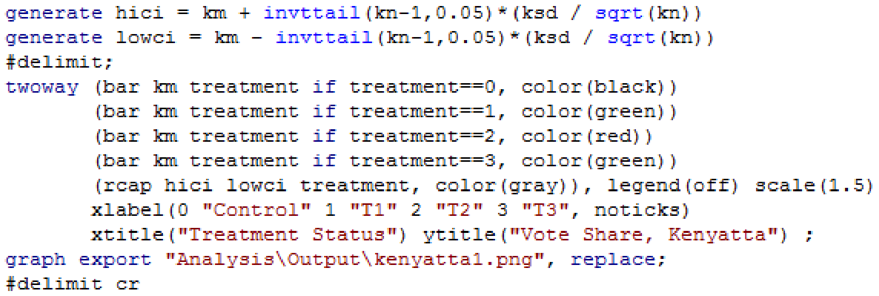
`scatter wage tenure` vs. `binscatter wage tenure`





Additional Slides

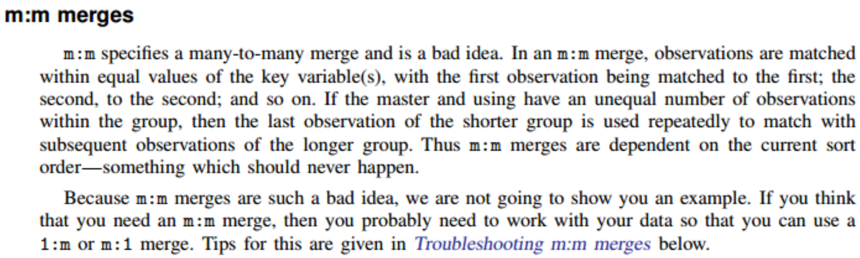
Use delimiters for long commands



```
generate hici = km + invttail(kn-1,0.05)*(ksd / sqrt(kn))
generate lowci = km - invttail(kn-1,0.05)*(ksd / sqrt(kn))
#delimit;
twoway (bar km treatment if treatment==0, color(black))
       (bar km treatment if treatment==1, color(green))
       (bar km treatment if treatment==2, color(red))
       (bar km treatment if treatment==3, color(green))
       (rcap hici lowci treatment, color(gray)), legend(off) scale(1.5)
      xlabel(0 "Control" 1 "T1" 2 "T2" 3 "T3", noticks)
      xtitle("Treatment Status") ytitle("Vote Share, Kenyatta") ;
graph export "Analysis\Output\kenyatta1.png", replace;
#delimit cr
```

back

m:m merges in the Stata manual



m:m merges

`m:m` specifies a many-to-many merge and is a bad idea. In an `m:m` merge, observations are matched within equal values of the key variable(s), with the first observation being matched to the first; the second, to the second; and so on. If the master and using have an unequal number of observations within the group, then the last observation of the shorter group is used repeatedly to match with subsequent observations of the longer group. Thus `m:m` merges are dependent on the current sort order—something which should never happen.

Because `m:m` merges are such a bad idea, we are not going to show you an example. If you think that you need an `m:m` merge, then you probably need to work with your data so that you can use a `1:m` or `m:1` merge. Tips for this are given in *Troubleshooting m:m merges* below.

back