ISRC/UI3 Data Science Institute: Introduction to Stata

Frederick J. Boehmke

University of Iowa

January 6, 2020

ISRC/UI3 Data Science Institute: Introduction to Stata

Frederick J. Boehmke

University of Iowa

January 6, 2020

Course Overview

Introduction to Stata
ISRC/UI3 Data Science Institute
January 6, 2020

Course Overview

Introduction to Stata
ISRC/UI3 Data Science Institute
January 6, 2020

Frederick J. Boehmke
Professor of Political Science
Director, Iowa Social Science Research Center
frederick-boehmke@uiowa.edu

Course Overview

Introduction to Stata
ISRC/UI3 Data Science Institute
January 6, 2020

Frederick J. Boehmke
Professor of Political Science
Director, Iowa Social Science Research Center
frederick-boehmke@uiowa.edu

Interests: American Politics; Statistical Programming, Event History Analysis, Multi-Level Modeling, Spatial Analysis, Non-random Sample Selection.

Course Overview

1. Intro to Stata.
2. Good computing practices.
3. Overview of basic commands.
4. Opening and saving data.
5. Manipulating variables and data.
6. Merging, appending, reshaping, and collapsing data sets.
7. Loops and macros.
8. Graphical display of data and results.

A Brief Background

- Version 1.0 was released in 1985. Now on Version 16.
- Has different "flavors": IC, SE, and MP.
- Can be pronounced however you want, but employees say STAY-ta.
- Written Stata, not STATA.

Getting Started

- `www.stata.com` a good starting point: software, books, and web courses.
- Statalist – online discussion forum.
- Stata Journal (was Stata Technical Bulletin).
- UCLA Stata website:
  `https://stats.idre.ucla.edu/stata/`.
- Other options:
  `https://www.stata.com/links/resources-for-learning-stata/`.

Presentation Conventions

1. Stata commands are indicated in `typewriter font`.
2. Stata commands you can run are preceded by a "." and in `typewrite font`.
3. Commands may break across two lines of slides, but they must be one line in the Stata command prompt (they can wrap, but no carriage returns).
4. Some special characters may not copy properly from pdf, e.g., _, ''.
5. I'll share batch files for all the commands we run.

Accessing Presentation Materials

You can access the presentation materials in three different ways.

1. Download from my website via a web browser and unzip:
   `https://myweb.uiowa.edu/fboehmke/dsi2020.zip`
2. Launch and download within Stata after changing to your preferred directory:
```
.cd H:
.copy https://myweb.uiowa.edu/fboehmke/dsi2020.zip dsi2020.zip
.unzipfile dsi2020.zip, replace
.cd dsi2020/
```
3. Just follow along and download the do files one at a time – I'll provide code when you need it and they will pull the data from the web for you.

If you just want the slides, you can access them at:
`https://myweb.uiowa.edu/fboehmke/dsi2020.pdf`

Open up the Software

1. If you have your own version of Stata, open it up.
2. If not, you can access it via Virtual Desktop:
   - Launch VD: `https://virtualdesktop.uiowa.edu/`.
   - Make sure you have the Citrix client, install if needed.
   - Log in and start Stata 16.
   - To change to your H: drive in Stata (if you want):
     `cd \\home.iowa.uiowa.edu\hawkid\`
     or use the menus (`File > Change Working Directory`) to select something on your local machine.
   - Support: `https://its.uiowa.edu/support/article/102185`.

Stata Windows

1. Results window;
2. Command window;
3. Command history window;
4. Variables window;
5. Variable properties window.

Stata Windows Tips/Shortcuts

1. Command window:
   - Type commands here;
   - You can copy and paste multiple commands into it;
   - You can tab-complete partial variable names to fill them in.
2. Command history window:
   - Double click on old commands to re-run;
   - Select and right-click to run multiple commands;
   - Select and right-click to copy or save commands to a file.
3. Variables window:
   - Double click variables to insert them on the command line.

Menus versus Command Line

- Use the command line!
- Menus can be helpful for finding new commands or options.
- Command line can be saved, copied, pasted, repeated.
- Help is available: menus, `search` or `findit`.

Batch Files

1. Even better than the command line.
2. Archive of your commands.
3. Create a library of code for future applications.
4. Assures reproducibility and enhances replication.

The Stata Batch File Editor

1. Open the batch file editor (from the menu bar, ctrl^9, or `doedit` at the command line).
2. Create a new batch file or open an existing one.
3. You run the whole file by clicking "run" or "do" in the menu bar, or by hitting ctrl^r or ctrl^d. You can also run groups of lines by highlighting them first.
4. It has some nice features, such as syntax highlighting.
5. Can use delimiters, e.g., ";", or comment out the end of line ("///") to continue to the next line.
6. There are other good editors out there that can be integrated into Stata with "select and do" commands. I often use Notepad++.

Best Practices

1. Write clear code (use spacing, indenting, blocking, etc.).
2. Use multiple lines when appropriate (with `#delimit;` or ///).
3. Don't overly complicate things.
4. Include comments liberally:
   ```
   /* Like this.  */
   * Or this.
   ```
5. Write file headers with comments.
6. Use version control.
7. Different files for different purposes (data creation, analysis, interpretation).
8. Log the results with `log using <filename>`.

Workflow

1. Set up your directories in a logical manner:
    1. Project directory;
    2. Subdirectories: data, graphs, tables, papers, slides, etc.
2. Name your files in a logical manner:
    1. mkdata-project01.do;
    2. analyze-project01.do;
    3. graph-project01.do;
    4. simulate-project01.do.
3. Use version control for major changes:
    1. mkdata-project01.do;
    2. mkdata-project02.do.

Basic Directory Commands

`pwd`	Print working directory (where am I?).
`ls`	List contents of current directory.
`cd`	Change directory.
`mkdir`	Make directory.

Basic Directory Commands

`pwd`	Print working directory (where am I?).
`ls`	List contents of current directory.
`cd`	Change directory.
`mkdir`	Make directory.

```
.cd H:/workshop2020/code/
.ls
.mkdir comp01
.cd comp01
```

System Variables

- Memory settings:
  - memory;
  - maxvar;
  - matsize.
- System directories:
  - Home directory;
  - Base directories;
  - User directories.

Opening Data

`use`	Open a Stata data set.
`import`	Open various file formats, e.g., Excel, SAS, ODBC, XML.
`insheet`	Open a delimited ascii file (e.g., .csv) – superseded by `import delimited`.
`infile`	Open a fixed format text file.

Opening Data

`use`	Open a Stata data set.
`import`	Open various file formats, e.g., Excel, SAS, ODBC, XML.
`insheet`	Open a delimited ascii file (e.g., .csv) – superseded by `import delimited`.
`infile`	Open a fixed format text file.

On your local machine:

```
.use comp01
.import delimited using comp01.csv, clear delimiter(",") varnames(1)
.import excel using comp01.xlsx, firstrow clear .infile using comp01,
using(comp01.txt) clear
```

Opening Data

We can also open data over the Internet:

```
.use https://myweb.uiowa.edu/fboehmke/stata2020-01/comp01
.import delimited using https://myweb.uiowa.edu/fboehmke/stata2018-01/comp01.csv,
     delimiter(",") case(preserve) varnames(1)
```

Accessing the Batch Files

Or you can grab any file from the Internet using the copy command.

Here is how to grab the first command file for this workshop:

```
.copy https://myweb.uiowa.edu/fboehmke/stata2020-01/comp01basic.do comp01basic.do
.doedit comp01basic.do
```

You may wish to change directories first (File > Change Working Directory) but you don't need to.

Saving Data

`save`	Save a Stata data set.
`saveold`	Save a Stata data set in version 11/13 format.
`export delimited`	Save a delimited file (e.g., .xlsx, .csv).
`outfile`	Save a fixed format text file.

Saving Data

`save`	Save a Stata data set.
`saveold`	Save a Stata data set in version 11/13 format.
`export delimited`	Save a delimited file (e.g., .xlsx, .csv).
`outfile`	Save a fixed format text file.

```
.save comp01basic01, replace
.saveold comp01basic01-version13, replace version(13)
.export delimited using comp01basic01.xlsx, comma replace
.outfile using comp01basic01, nolabel replace
    dictionary
```

Inspecting Your Data

`describe`	Basic features of the data set.
`summarize`	Summary statistics of your variables.
`tabulate`	One- and two-way tabulation of your variables.
`tab1`	One-way tabulation of many variables.
`table`	Tabulation with user-specific entries.
`tabstat`	Choose the statistics you want.
`histogram`	Basic distribution of a variable.

Some Basic Concepts

varnames: naming rules for Stata variables:

- Can be pretty long;
- Can not start with a number;
- Can use var_1 but not var-1.

varlist: way to express a group of Stata variables:

1. var1 var2 var3
2. var1-var3
3. var*

numlist: list of values:

- 1 2 3 4;
- 1/4;
- 1(1)4.

Some Basic Concepts

abbreviations: Stata commands can be abbreviated:

observations: _n accesses the observation number. _N is the largest observation.

if: do a command for certain values.

- equals: `if var==expression`;
- not equals: `var!=expression` or `var~=expression`.

in: do a command for certain observations by number.

inlist: do a command for certain values of a variables.

by varname: do a command for different values of a variable.

capture: run a command and capture any error codes.

Missing Values

- Numeric missing value: “.”;
- String missing value: “ ”;
- Extended numeric missing values: `.a .b .c ... .z`.
- Missing values are infinity!
- Use `if !missing()` rather than `if var != .` to capture missing and extended missing.

Missing Values

- Numeric missing value: “.”;
- String missing value: “ ”;
- Extended numeric missing values: `.a .b .c ... .z`.
- Missing values are infinity!
- Use `if !missing()` rather than `if var != .` to capture missing and extended missing.

```
.summarize educ if income >= 7
.summarize educ if income >= 7 & !missing(educ)
```

Labeling Data and Variables

`label data`	Label a data set.
`label variable`	Label a variable.
`label define`	Create value label.
`label values`	Label a variable with value labels.
`label dir`	List values labels.
`label list`	List contents of value labels.
`label save`	Save value label definitions.
`label notes`	Create and list notes for variables.

Labeling Data and Variables

```
.label data "Stata Workshop (Boehmke, 2020)"
.label variable natlec "Economic Conditions"
.label define female 0 Male 1 Female
.label values female female
.notes natlec "Information about the variable coding."
```

Labeling Data and Variables

```
.label data "Stata Workshop (Boehmke, 2020)"
.label variable natlec "Economic Conditions"
.label define female 0 Male 1 Female
.label values female female
.notes natlec "Information about the variable coding."
```

Then, to see the results, `describe` your data.

Variable Formats

- String variables (recently expanded):
  1. str#, e.g., str8 – up to str2045 (2045 bytes/characters);
  2. strL – up to 2 billion bytes long;
- Numeric variables:
  1. byte;
  2. int;
  3. long;
  4. float;
  5. double.

Issues with precision.
Display formats.

Structuring your Data

`compress`	Store data efficiently.
`sort`	Sort observations by a variable's values.
`gsort`	General version of `sort`.
`order`	Order the variables in the data set.

Basic Commands for Data Analysis

`ttest`	Difference in means test.
`tabulate`	Crosstab with conditional frequencies test.
`regress`	Linear regression.
`mixed`	Multilevel model (a.k.a., HLM, mixed effects).
`logit`	Discrete choice model.
`ologit`	Ordered choice model.
`mlogit`	Unordered choice model.

Now You Can...

- Open data in multiple formats;
- Inspect and create variable and value labels;
- Run common analyses and tests.

So What's Left?

- Few projects are so clean that you can implement them with the above.
- Here are some common challenges researchers encounter I'd like to address in the remaining time:
  1. More complex data cleaning;
  2. Combining multiple data sets;
  3. Reshaping data sets;
  4. Loops and macros;
  5. Making graphs.

The Basics of Data Entry and Cleaning

`generate`	Create a variable.
`replace`	Change values of a variable.
`rename`	Change a variable's name.
`rename group`	Also rename groups of variables.
`renvars`	Rename groups of variables (add-on).
`recode`	Complex value replacement.
`revrs`	Reverse a variable's coding (add on).
`destring`	Convert string variable to numeric.
`real()`	Extract real numbers from string variable.
`encode`	Create labeled numeric variable from string variable.
`decode`	Create string variable from labeled numeric variable.

Groups of Functions in Stata

`help mathfun`	Mathematical functions.
`help probfun`	Probability and density functions.
`help strfun`	String functions.
`help progfun`	Programming functions.

Groups of Functions in Stata

`help mathfun`	Mathematical functions.
`help probfun`	Probability and density functions.
`help strfun`	String functions.
`help progfun`	Programming functions.

```
.generat x1_pos = abs(x1)
.generat x2_den = normalden(x2,3,2)
.generat firstname = word(name,1)
.replace name = trim(name)
.replace name = itrim(name)
.replace name = proper(name)
.replace name = proper(trim(name))
```

Accessing the Second Batch File

Grab the second batch file for this workshop:

```
.copy https://myweb.uiowa.edu/fboehmke/stata2020-01/comp02data.do comp02data.do
.doedit comp02data.do
```

Cleaning Data Example

- Say we wanted to enter some US Census data on state population.
- Open up pop.csv, the file I downloaded from the BEA.
- This is a comma separated file, so we need to use `import delimited` to open it.
- Now let's inspect what we have in Stata.
- And finally, let's do the cleaning.
- Once we've done that, let's check for possible errors.

Reshaping Data Sets

reshape	Switch between long and wide structure: X_{ij} becomes $X1_i, X2_i, \ldots, XJ_i$ or vice-versa.
collapse	Reduce the number of observations and create summary statistics overall or by() a variable.
set obs	Increase the number of observations in the data.
expand	Duplicate the existing observations in a data set.
xpose	rotate a data set so that columns become rows (rarely needed).

Reshaping Data Sets

If we start with a data set in long format....

id	*year*	*x*
1	1	4
1	2	2
1	3	7
2	1	8
2	2	1
2	3	3
3	1	5
3	2	9
3	3	3

Reshaping Data Sets

If we start with a data set in long format....

id	*year*	*x*
1	1	4
1	2	2
1	3	7
2	1	8
2	2	1
2	3	3
3	1	5
3	2	9
3	3	3

We can `reshape` it to wide format....

id	*x*1	*x*2	*x*3
1	4	2	7
2	8	1	3
3	5	9	3

Reshaping Data Sets

If we start with a data set in long format....

id	*year*	*x*
1	1	4
1	2	2
1	3	7
2	1	8
2	2	1
2	3	3
3	1	5
3	2	9
3	3	3

We can `reshape` it to wide format....

id	*x*1	*x*2	*x*3
1	4	2	7
2	8	1	3
3	5	9	3

```
.reshape wide x, i(id) j(year)
```

To go back, just reverse the command:

```
.reshape long x, i(id) j(year)
```

Accessing the Third Batch File

Grab the first batch file for this workshop:

```
.copy https://myweb.uiowa.edu/fboehmke/stata2020-01/comp03reshape_merge.do
     comp03reshape_merge.do
.doedit comp03reshape_merge.do
```

Reshaping Data Sets

```
.reshape wide totpop, i(state) j(year)
.reshape long totpop, i(state) j(year)
.collapse (mean) pop (sd) pop_sd=pop, by(state year)
```

Reshaping Data Sets

```
.reshape wide totpop, i(state) j(year)
.reshape long totpop, i(state) j(year)
.collapse (mean) pop (sd) pop_sd=pop, by(state year)

.set obs 20
.generat country=_n
.expand 5
.bysort country:  generat time=_n
.tab country time
```

Combining Data Sets

append add one data set at the bottom of existing data set.

joinby join one data set to existing data set by varlist (largely obsolete given revisions to merge).

merge combine multiple data sets by matching observations across by varlist. Can do one-to-one, many-to-one, and many-to-many merges (though many-to-many not recommended – see `help merge`).

append

```
.use population1968
.append using population1969
.append using population1970
```

joinby

```
.use income1960-2000
.joinby year using cpi
.generat rpcpi = pcpi*cpi
```

merge

```
.use income1960-2000
.merge 1:1 country year using population1970-2000,
     keep(match master using) generat(_merge_pop70)
.merge 1:1 country year using population1960-1970,
     keep(match) generat(_merge_pop60) update
.merge 1:m year using cpi, keep(match master)
     assert(match master) nogenerate
```

The second merge will only replace missing values of population (or other variables in both data sets).

Using duplicates to Check your Data

`duplicates report`	Summarize duplicate observations.
`duplicates examples`	Show examples of duplicates.
`duplicates list`	Show all duplicates.
`duplicates tag`	Create variable marking duplicates.
`duplicates drop`	Drop duplicates.

`duplicates` works on all variable by default, but you can specify a list of variables if you want:

```
.duplicates report country year
.duplicates drop country year
```

Basic Data Structure Commands

`tsset`	Time-series data.
`xtset`	Cross-sectional time-series data.
`stset`	Survival time data.
`spset`	Spatial data (as of Stata 15).
`svyset`	Survey data.

Basic Data Structure Commands

`tsset`	Time-series data.
`xtset`	Cross-sectional time-series data.
`stset`	Survival time data.
`spset`	Spatial data (as of Stata 15).
`svyset`	Survey data.

```
.xtset stateno year
```

egen() Functions

- egen() is Stata's extended generate functions.
- Very useful for creating variables containing summary information about a variable or variables.
- Can calculate various statistics about a group of variables (e.g., mean, sd, count, etc.).
- With the `by()` option, can do the same for one variable across groups of observations.
- See the add-on `egenmore()` for additional capabilities.

Accessing the Fourth Batch File

Grab the fourth batch file for this workshop:

```
.copy https://myweb.uiowa.edu/fboehmke/stata2020-01/comp04egen.do comp04egen.do
.doedit comp04egen01.do
```

egen() Functions

```
.egen natlec_mean = mean(natlec), by(state)
.egen adopt_min = rowmin(adopt)
.egen nonmissing = rownonmiss(female race educ
    income natlec)
.egen statepid = group(state pid3_p)
```

Generating Unique ID variables

- If you have just one group of observations:
  `.generat id = _n`
- If you have repeated cross sectional time series observations and want a unique identifier for each unit or for each observation:
  `.egen id = group(country)`
  `.egen id = group(country year)`
- If you have different units in each group:
  `.bysort state:  generat id = _n`
  `.egen voter_id = seq(), from(1) by(state)`

Loops

- Loops perform a series of steps for different values of an index variable.
- Can loop over a numlist.
- Can loop over variables.
- Can loop over values of a macro.
- Can loop over values of an arbitrary list.
- Helps to know a little about local macros to do this.

Aside: Globals and Locals

- Globals and locals can be fixed by the user or based on results for conditional command processing.
- Both can be used for repeated values so you only have to change them once.
- Globals are accessible within a session in a batch file and at the command line.
- Locals are accessible in a batch file or at the command line.
- Globals referred to (but not declared) with `$global`.
- Locals referred to (but not declared) with single quotes: `` `local' ``.
- If you want the name of a variable, also add double quotes `` "`local'" ``, otherwise you get the value of `` `local'[1] ``.

Aside: Globals and Locals

```
.local i = 1
.display `i'
.global basevars income educ female
.regress turnout $basevars
.global numobs 1000
.regress turnout $basevars pid3_p in 1/$numobs
.local mean = r(mean)
.display "The mean is `mean'"
```

Accessing Results

- Basic results stored in r().
- Estimation results stored in e().
- System variables stored in c().
- Coefficients stored in [equation]_b[varname];
- Standard errors stored in [equation]_se[varname].
- These can be accessed with `return list`, `ereturn list`, `help creturn`.
- Display will show what you type.

Display and Accessing Results

`display` will show the results of whatever you enter, filling in locals and globals with their values and performing any calculations you write.

```
.display "Some text"
.summarize income
.display r(mean)
.display "This is the mean of the income variable:  " r(mean)
.display 23/5
.display r(n)
.display _b[x1]
```

Accessing the Fifth Batch File

Grab the fifth batch file for this workshop:

```
.copy https://myweb.uiowa.edu/fboehmke/stata2020-01/comp05loops.do comp05loops.do
.doedit comp05loops.do
```

forvalues Loops

```
.forvalues i=1/10 {
.    display `i'
.     }
```

```
.forvalues income=1/14 {
.    summarize turnout if `income' == `i'
.     }
```

forvalues Loops

```
.forvalues i=2(2)170 {
.    rename v`i' v`i'_adopt
.     }
```

```
.forvalues i=3(2)169 {
.    rename v`i' v`i'_inn
.     }
```

foreach Loops

```
.foreach word in "apple" "pear" "banana" {
.    display "`word'"
.     }
```

```
.local words "apple pear banana"
.foreach word of local words {
.    display "`word'"
.     }
```

foreach Loops

```
.foreach rowvar of varlist turnout educ income {
.    tab gender `rowvar'
.     }
```

```
.foreach num of numlist 1 3 7 10 {
.    tab turnout if income == `num'
.     }
```

Installing User-Written Programs

net search	helps you find user written programs.
net from	is how you install from the command line.
net cd	let's you change directories.
net describe	lets you view a packages information.
net install	installs the package.
net set ado	allows you to install into a non-default local directory.
ado	lists your installed packages.
ado uninstall	removes a package.

Managing User Written Packages

adoupdate	lets you update non-Stata packages.
adolist	will list available packages. Even better, it will let you save a file listing all of your installed packages, which you can then transfer to another machine and reinstall.
usersite	provides information about setting up your own Stata package server.
which	allows you to get version information about package.
viewsource	allows you to see the actual code of the .ado file.

```
.adolist store mypackages
.adolist install mypackages
```

Installing Programs

```
.net search interaction
.net search fs.ado
.net from http://fmwww.bc.edu/RePEc/bocode/f
.net install fs
.ado
.adolist store mypackages
```

It is usually easier to just use the clickable links to install programs, but you can put the above in a batch file to document the package's source.

Graphical Displays of Data

- Pictures let us see data in many ways.
- Patterns can emerge that would not in a table.
- Figures are more intuitive for most people.
- Figures may be even more important in the era of big data.

How to Graph your Data

- How many variables are you interested in?
- Are the variables nominal or continuous?
- Is there an interesting way to split the data?
- What is the primary point you are hoping to make?

Figures for One Variable

- If you have a nominal variable, the goal is usually to show the frequency of the different categories.
- You will therefore usually want to use a histogram or bar chart.
- Try to avoid pie charts! The eye is better at linear comparisons.

Figure: A Good Pie Chart

Figures for Two Variables

- If you have two continuous variables, you will likely want to create a scatter plot or a line plot.
- A special case is variables measured in time, for which you would probably want a line or bar graph.
- Another special case is a variable measured in geographic space, for which you would probably want a map (see `spmap` or `spset`).
- If at least one variable is nominal, these approaches won't work as well. You should consider a series of bar charts.
- If both variables are ordinal scatter plots won't work well – use the same approach as you would for a nominal variable.

Figures for Three (or more) Variables

- In many cases you just put multiple lines or scatter plots in the same figure.
- You can also consider area plots, which shade in regions.
- To do controlled comparisons, you can also split a variable by values of another and graph the subsets.
- Use a different dimension to show the third variable, such as size or color.

Graphing in Stata

Two sets of commands: `graph` and `twoway`.

`graph` is used for basic graphs, but mostly for saving, opening, and combining graphs.

`twoway` is the main graphing command to create plots.

twoway Graphs

- Twoway graphs plot one or more variables against another variable:
  1. scatter;
  2. line;
  3. connected;
  4. bar;
  5. area;
- Or to connect the values of two variables at a third:
  1. rline;
  2. rpike;
  3. rbar;
  4. rarea;
- Each has its own combination of line, marker, and bar options.

Accessing the Sixth Batch File

Grab the sixth batch file for this workshop:

```
.copy https://myweb.uiowa.edu/fboehmke/stata2020-01/comp06graph.do comp06graph.do
.doedit comp06graph.do
```

General Features

scheme() Sets the overall color and line/marker scheme.

by() Create multiple graphs for all values of a variable.

Axis Features

yaxis()	Controls which axis (left/right).
ylabel()	Labels the values & controls gridlines.
ytitle()	Axis title.
yscale()	Control appearance & scale (linear, log, reversed, time).
yline()	Draw a horizontal line across the graph.

Title and Text Features

legend() Legend options.
title() Graph title options.
b1title() Title on the bottom (or right, left, top).
note() Add a note at the bottom of the graph.
text() Add text anywhere on the graph.

Size and Storage Features

graphregion() Control the graph region (color, border).
plotregion() Control the plot region (color, border).
ysize() Width of the graph region.
saving() Save the graph to disk.
name() Save the graph in memory.

Combining Graphs

- You can combine both memory and saved graphs:
  `.graph combine` *graph1 graph2*
  `.graph combine` *graph1.gph graph2.gph*
- You can control the number of rows and columns, placement of holes, intergraph margins, etc.
- It's not always necessary to repeat axis titles and labels — use `b1title` and `l1title` instead.
- `grc1leg` is a great add on to combine graphs with just one legend, though it's occasionally a bit quirky.

Overlaying Graphs

Two ways to overlay graphs.

```
.twoway rcap fd_90_lo fd_90_hi obsid
     || scatter fd obsid
.twoway (rcap fd_90_lo fd_90_hi obsid)
     (scatter fd obsid)
```

The graph Command

`.graph` *option*:

combine	Combine multiple graphs into one graph.
save	Save the currently open graph.
use	Open a graph saved to hard drive.
export	Convert a graph to another format (.ps, .pdf, .eps, .wmf).
bar	Graph statistics of variables, possibly over() a variable.
matrix	Scatter plots of multiple variables in one graph.

A Fairly Complicated Graph

```
twoway rbar fd_90_lo fd_90_hi x2, scheme(s2color)
 lcolor(blue) color(ltblue)
 barwidth(0.1)
   || line fd x2,
 lcolor(dknavy) lwidth(thin)
   || scatter fd x2 if mod(_n,2)==1,
 msymbol(i) mlabel(fd) mlabpos(0) mlabcolor(dknavy) mlabsize(medsmall)
 xlabel(#7, labcolor(dknavy))
 ylabel(#5, labcolor(dknavy))
 xtitle(x2, color(dknavy) size(medlarge))
 ytitle(First Difference, color(dknavy) size(medlarge))
 title("First Difference for x1 on Pr(Y=1|X), Varying x2", color(dknavy))
 note("First difference represents a change in x1 from one SD below its"
   "mean to one SD above its mean.", color(dknavy))
 legend(off);
```

Tips for Enhancing Graphs

- Use separate axes for variables on different scales.
- Use text to label specific values or lines.
- Use color to distinguish groups or values.
- Use size to indicate differences (e.g., number of observations).
- Try to group similar values by color or patterns.
- Add reference points and lines to ease interpretation.

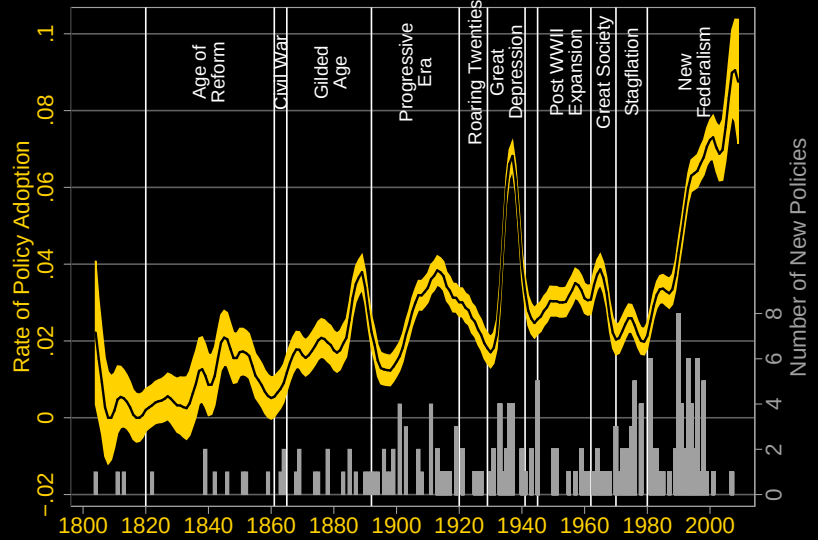
Useful Tips and Commands for Stata Graphs

- Overlaying graphs allows you to be creative.
- Use `mlabel(varname)` to use identifiers instead of markers.
- Use `lowess` to smooth out bouncy simulation lines.
- Use `collapse` and [fw=weightvar] to scale scatter plots by frequency of combinations to show a joint or conditional distribution.
- Use `graph combine` in multiple steps to build a graph.
- Use `xsize(#) ysize(#)` to make graphs the right size for your slides (5x3) or paper (e.g., 6x4.5 or 6.5x6.5).
- Check out `palette_all` (add on) to see a listing of all colors.

Issues to be Aware of

- Not showing the entire scale of the axes.
- Cutting off the trend at the wrong points.
- Not comparing variables on the same scales.
- Using proportional changes inappropriately (e.g., height versus volume).
- Don't use more dimensions than you need.
- Try not waste ink on non-data related information.
- Use inflation and population-adjusted numbers whenever possible.

Figure: State Policy Innovativeness Over Time



Fun Graph Websites

- I Love Charts: http://ilovecharts.tumblr.com/.
- Chart Porn: http://chartporn.org/.
- US wind map: http://hint.fm/wind/.
- Gapminder: http://www.gapminder.org/.