On the shoulders of giants, or not reinventing the wheel

Nicholas J. Cox
Department of Geography
Stata users can stand on the shoulders of giants.

Giants are powerful commands to reduce your coding work.

This presentation is a collection of examples based on some commands that seem little known or otherwise neglected.

Every user is a programmer. The range is from commands usable interactively to those useful in longer programs.
On the shoulders of giants

This saying has a long history, recounted in a monograph by Robert K. Merton (1910–2003).

With gravitas

If I have seen further it is by standing on the shoulders of Giants.

Isaac Newton (left) (1642–1727) writing to Robert Hooke (right) (1635–1703) in 1676
With topological wit

to Christopher Zeeman
at whose feet we sit
on whose shoulders we stand

Tim Poston and Ian Stewart.

Sir Christopher Zeeman
(1925–2016) (right)

Tim Poston (1945–2017)
Ian Nicholas Stewart (1945– )
Tabulation tribulations?
Tabulations and listings

For tabulations and listings, the better-known commands sometimes seem to fall short of what you want.

One strategy is to follow a preparation command such as generate, egen, collapse or contract with tabdisp or _tab or list.
Newer preparation commands

tsegen and rangestat (SSC; Robert Picard and friends) are newer workhorses creating variables to tabulate.

tsegen in effect extends egen to time series and produces (e.g.) summary statistics for moving windows.

rangestat covers a range of problems, including irregular time intervals, look-up challenges, other members of a group.

Search Statalist for many examples.
tabdisp

From the help: `tabdisp` calculates no statistics and is intended for use by programmers.
In the manuals: it is documented at `[P] tabdisp`.

It may seem that StataCorp is *trying to discourage you from using this command*.

Keep off: programmers only!

But it’s easy: you just need to know, or at least calculate in advance, what you want to display.
tabdisp: numbers and strings too

Feature: tabdisp can mix numeric and string variables in its cells.

This is useful in itself and as a way of forcing particular display formats (# of decimal places, date formats).

So before tabdisp you could go

gen show_val = string(myresult, "%2.1f")
gen show_date = string(mydate, "%tdD_M_CY")
tabdisp is used in moments

moments (SSC) shows sample size, mean, SD, skewness and kurtosis. It uses summarize for calculations and tabdisp for tabulation.

In moments the default format for everything but sample size is %9.3f, but that can be overridden.

Aside: Have you ever been irritated that tabstat lets you change but not vary the display format? Even 1 decimal place is too many for sample size, but could be too few for other statistics.
. sysuse auto, clear  
(1978 Automobile Data)

. moments mpg price weight

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>SD</th>
<th>skewness</th>
<th>kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mileage (mpg)</td>
<td>21.3</td>
<td>5.8</td>
<td>0.949</td>
<td>3.975</td>
</tr>
<tr>
<td>Price</td>
<td>6165.3</td>
<td>2949.5</td>
<td>1.653</td>
<td>4.819</td>
</tr>
<tr>
<td>Weight (lbs.)</td>
<td>3019.5</td>
<td>777.2</td>
<td>0.148</td>
<td>2.118</td>
</tr>
</tbody>
</table>

. moments mpg price weight, format(%2.1f %2.1f)

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>SD</th>
<th>skewness</th>
<th>kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mileage (mpg)</td>
<td>21.3</td>
<td>5.8</td>
<td>0.949</td>
<td>3.975</td>
</tr>
<tr>
<td>Price</td>
<td>6165.3</td>
<td>2949.5</td>
<td>1.653</td>
<td>4.819</td>
</tr>
<tr>
<td>Weight (lbs.)</td>
<td>3019.5</td>
<td>777.2</td>
<td>0.148</td>
<td>2.118</td>
</tr>
</tbody>
</table>
lmoments (SSC) is another example.

The code shows examples of a useful technique, storing results in variables that need not be aligned with the main dataset.

Not being able to have two or more datasets in memory is a frequent complaint....
tabdisp uses the first value it sees

Feature: tabdisp uses the value in the first pertinent observation it encounters.

For rows with unique identifiers, that is exactly right.

For groupwise summaries, that is a good default.

You just need to know about it.

It is documented explicitly.

Limit: Up to five variables may be displayed as cells in the table. (Many tables are far too complicated, any way.)
**tabdisp**

Tabulate cumulative frequencies as well as frequencies?

```stata
sysuse auto, clear
by rep78, sort: gen freq = _N
by rep78: gen cumfreq = _N if _n == 1
replace cumfreq = sum(cumfreq)
tabdisp rep78, cell(freq cumfreq)
```

_tab

This really is a programmer’s command, but can be used minimally:

- Top: Declare structure, specify top material
- Body: Loop over table rows, populating the table cells
- Bottom: Draw bottom line

Example in missings (SSC; Stata Journal 15(4) 2015 and 17(3) 2017).

Another example in distinct (Stata Journal 15(3) 2015 and earlier).
// top of table
tempname mytab
```
mytab' = .tab.new, col(`nc') lmargin(0)
if `nc' == 3 .mytab'.width `w1' | `w2' `w3'
else .mytab'.width `w1' | `w2'
.mytab'.sep, top
```
if `nc' == 3 .mytab'.titles " " "#" 
else .mytab'.titles " " 
.mytab'.sep

// body of table
forval i = 1/`nr' {
    forval j = 1/`nc' {
        mata: st_local("t`j'", mout[i, j])
    }
    if `nc' == 3 .mytab'.row "`t1'" "`t2'" "`t3'"
    else .mytab'.row "`t1'" "`t2'"
}

// bottom of table
.mytab'.sep, bottom
Most users know list, but do you know it well?

Any table that can be presented as a listing can be presented with list. It has several useful options.

We can get arbitrarily complicated:

- Row identifiers
- Row and column identifiers
- Many identifiers
list exploited in groups

groups is a tabulation command that is a wrapper for list.

It was originally documented in *Stata Journal* 3(4) 2003 but has been much updated since. A revised account appeared in *Stata Journal* 17(3) 2017, updated in 18(1) 2018.

At its simplest it looks like `tabulate` in disguise, but it can do other stuff too.
groups regards a table as a task for list
row identifier, stuff
row identifier, column identifier, stuff
identifiers, stuff
. sysuse auto, clear

. groups foreign

<table>
<thead>
<tr>
<th>foreign</th>
<th>Freq.</th>
<th>Percent</th>
<th>% &lt;=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domestic</td>
<td>52</td>
<td>70.27</td>
<td>70.27</td>
</tr>
<tr>
<td>Foreign</td>
<td>22</td>
<td>29.73</td>
<td>100.00</td>
</tr>
</tbody>
</table>
. groups foreign rep78

<table>
<thead>
<tr>
<th></th>
<th>foreign</th>
<th>rep78</th>
<th>Freq.</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domestic</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2.90</td>
</tr>
<tr>
<td>Domestic</td>
<td>2</td>
<td>8</td>
<td>11</td>
<td>11.59</td>
</tr>
<tr>
<td>Domestic</td>
<td>3</td>
<td>27</td>
<td>39</td>
<td>39.13</td>
</tr>
<tr>
<td>Domestic</td>
<td>4</td>
<td>9</td>
<td>13</td>
<td>13.04</td>
</tr>
<tr>
<td>Domestic</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2.90</td>
</tr>
<tr>
<td>Foreign</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4.35</td>
</tr>
<tr>
<td>Foreign</td>
<td>4</td>
<td>9</td>
<td>13</td>
<td>13.04</td>
</tr>
<tr>
<td>Foreign</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>13.04</td>
</tr>
</tbody>
</table>
. groups foreign rep78, percent(foreign)

<table>
<thead>
<tr>
<th>foreign</th>
<th>rep78</th>
<th>Freq.</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domestic</td>
<td>1</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>Domestic</td>
<td>2</td>
<td>8</td>
<td>16.67</td>
</tr>
<tr>
<td>Domestic</td>
<td>3</td>
<td>27</td>
<td>56.25</td>
</tr>
<tr>
<td>Domestic</td>
<td>4</td>
<td>9</td>
<td>18.75</td>
</tr>
<tr>
<td>Domestic</td>
<td>5</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>Foreign</td>
<td>3</td>
<td>3</td>
<td>14.29</td>
</tr>
<tr>
<td>Foreign</td>
<td>4</td>
<td>9</td>
<td>42.86</td>
</tr>
<tr>
<td>Foreign</td>
<td>5</td>
<td>9</td>
<td>42.86</td>
</tr>
</tbody>
</table>
. groups mpg, select(f == 1) show(none)

+--------+
| mpg    |
+--------+
| 29     |
| 31     |
| 34     |
| 41     |
+--------+
. groups mpg, select(-5)

<table>
<thead>
<tr>
<th>mpg</th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>2</td>
<td>2.70</td>
<td>93.24</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>1.35</td>
<td>94.59</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>1.35</td>
<td>95.95</td>
</tr>
<tr>
<td>35</td>
<td>2</td>
<td>2.70</td>
<td>98.65</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>1.35</td>
<td>100.00</td>
</tr>
</tbody>
</table>
. groups mpg, select(5) order(h)

<table>
<thead>
<tr>
<th>mpg</th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>9</td>
<td>12.16</td>
<td>12.16</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>10.81</td>
<td>22.97</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>8.11</td>
<td>31.08</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>6.76</td>
<td>37.84</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
<td>6.76</td>
<td>44.59</td>
</tr>
</tbody>
</table>

26
list

Once again, list is the engine here.

My favourite options of list include

abbreviate(#) abbreviate variable names to # columns

noobs do not list observation numbers
(think: no obs, not newb[ie]s)

sepy(varlist) separator line if varlist values change

subvarname characteristic for variable name in header
list: find out about its other options

Many Stata users meet list early in their Stata education.

And they find it easy to understand: it lists data. Sure....

If you are a more experienced user, you should now go back to the help and find out which more advanced options you may have been missing out on.
Graphics grumbles?
statsby

Few needs are commoner than collating groupwise results. Few ways of doing it are more neglected than statsby.

The **statsby** strategy ([Stata Journal](https://www.jstatsoft.org/v10/i01) 10(1) 2010) hinges on using statsby to produce a dataset (results set?) and then firing up graph.

Detailed code is available in the paper, so we’ll switch style to show first some results for box plots in idiosyncratic form.

Key options of statsby here are subsets and total.
Box plots on multiple levels

The bottom line is a box plot for all observations — with a display of just extremes, quartiles and median.

Above that, we have one-way breakdowns.

Above that, we have a two-way breakdown.

Naturally we could see a problem of combinatorial explosion, but that is a problem for the researcher, not the programmer.
statsby

statsby is also a natural for confidence interval plots.

statsby underlies designplot, a generalisation of the not very much used grmeanby.

For designplot see Stata Journal 14(4) 2014 and later updates. (Latest update in press for 19(3) or 19(4).)

The idea is again to show summary statistics on one or more levels, e.g. whole dataset; by categorical predictors; by their cross-combinations; and so on.

In turn it is a wrapper for graph dot or graph hbar.
As before subsets and total options are key.

Next we will see two aligned design plots for a version of a famous dataset on admissions to various majors A ... F at UC Berkeley.

Lower admission rate for females looks like discrimination, but major by major females often do better.

The aggregate pattern is an artefact of different frequencies of application to the majors, a case of Simpson’s paradox.
Simpson’s paradox

This paradox was named for Edward Hugh Simpson (1922—2019) but was discussed much earlier by Karl Pearson and G.U. Yule.

In a nutshell: patterns can appear on amalgamation that are just side-effects of other aspects of the data, here the group frequencies.
Another example

*RMS Titanic* sank after colliding with an iceberg on its maiden voyage in April 1912.

Some of the people on board survived....
Mosaic plots, any one?

Mosaic plots are quite popular for showing proportions and frequencies of cross-classified categorical data.

They can work very well, but arguably such plots can become too busy to decode with three-way or higher data.

!!! can be hard work to decode if zeros or very small proportions or frequencies are present.
by() option for combining graphs

Many users know `graph combine` for combining graphs, and indeed it is very useful. I have used it often, e.g. in `combineplot`, `crossplot` and `corrttable` on SSC.

But there are downsides, notably that the same design repeated is ... the same design repeated, with often similar text shown again and again.

A favourite trick of mine is using a `by()` option to produce separate panels, often after a restructuring of the data. The art that conceals art is then to obscure that trick.
Counting sunspots

Astronomers (and even some economists) have been contemplating sunspots for some centuries.

Even the annual data are a handful. See here for source.

A standard line plot is just a roller coaster. Achterbahn?

(Using sunspots as an example is a miniature tradition here, started by William S. Cleveland in his books.)
Aspect ratio and slicing

R.A. Fisher, W.S. Cleveland and many others recommend getting segment slopes close to 45°.

A standard device is to slice the series and portray slices in short and wide panels, stacked vertically.

An even older device was to have very wide graphs as fold-out illustrations.

sliceplot is a program to do this (Stata Journal 6(3) 2006) which hinges on graph combine, but there is a much simpler solution.
* just two lines of code!

* choose your own # of slices, here 4
* ensure slices are nearly equal

```plaintext
gen slice = ceil(4 * _n/_N)
```

* see how much we zap

```plaintext
line mean year, by(slice, note(""`) col(1)
               xrescale) subtitle("", fcolor(none))
               xtitle("") yla(, ang(h)) ytitle(Mean
               sunspot number)
```
Mean sunspot number

- 1700 to 1780
- 1780 to 1860
- 1860 to 1940
- 1940 to 2020
Fine structure is now visible

Asymmetric saw-tooth cycles are more evident.

The rising limb of each cycle is steeper than the falling limb.

Back to the astronomers, climatologists, economists....

Note that we really don’t need year or any other such word in the x axis title.
Let us look at that code more slowly...

The fraction \( \frac{n}{N} \) is the current observation number \( n \) divided by the total number of observations \( N \).
So it rises from \( \frac{1}{N} \) to \( \frac{N}{N} \) or 1.
So 4 times that rises to 4. Choose your own value of 4.

The `ceil()` function (think: ceiling) rounds up to the next integer, so the result is just integers 1 or 2 or 3 or 4.

But we don’t need or want to see 1 to 4 on the graph. That is why the subtitle is replaced by an empty string — as indeed is the note.
by() to get multiple panels cleanly

The same trick, with more work, underlies multiline and multidot (both from SSC).

In each case several variables are temporarily stacked into one and then by() is the machinery yielding multiple panels in similar, minimal style.

multiline is an alternative to spaghetti plots, especially for responses in time on quite different scales or with quite different units. It is a wrapper for twoway. Compare also sparkline (SSC).
subtitle() is tricky

With `multiline` the `subtitle()` option is tricky. Put it on the left and remove traces of surrounding stuff.

```
subtitle(, pos(9) bcolor(none) nobexpand place(e))
```

This is wired into the code, but it is a little hard to remember it all for your own separate use. Keep a note somewhere accessible.

(I stole this from the manuals.)
multidot

multidot is also a solution when multiple responses have different scales and/or units.

It too is a wrapper for twoway – although in look and feel it is closer to graph dot or graph bar.

The *dot element just refers to the default. You can recast to other twoway types.

It is also arguable that such (Cleveland) dot charts remain underused.
<table>
<thead>
<tr>
<th>Make and Model</th>
<th>Price (USD)</th>
<th>Mileage (mpg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peugeot 604</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volvo 260</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMW 320i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audi 5000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datsun 810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW Dasher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW Scirocco</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audi Fox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datsun 200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toyota Celica</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Honda Accord</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toyota Corona</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW Diesel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datsun 510</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW Rabbit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datsun 210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Honda Civic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiat Strada</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mazda GLC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Renault Le Car</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subaru</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toyota Corolla</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Not using `graph combine` here

Once more: the device here is not to use `graph combine` but to call `graph` with a `by()` option after a temporary change to the data.
Graph and table flavour together

A sharp separation between graphs (Figure 1 ...) and tables (Table 1 ...) was arguably a side-effect of the invention of printing, causing a division of labour. Who did the work, or at least how was it done? (Before that, we had manuscripts, *written by hand*.)

Now graphs and tables need be not so sharply distinct.

Be that as it may, graphs with table flavour often look better with horizontal axis information at the top.
Another example: 16 Länder

The next example shows some German economic data.

Two measures of well-being on quite different scales are arguably better shown juxtaposed than superposed.

Having different graphical styles for different variables is only important for presentation, but being able to highlight a subset of observations (here just the Land of Bayern) can be more valuable yet.
Highlight just one subset

A minimalist principle is to highlight just one subset.

“when everything is emphasized, nothing is emphasized”

Think: Would you colour each sentence word differently?
(Credit for example: Naomi B. Robbins.)
Unconventional coordinates

A tougher problem is drawing graphs in coordinates other than rectangular or Cartesian.

Three proportions adding to a total, say $p + q + r = 1$, fall on a plane and so may be projected to triangular (trilinear, ternary) coordinates.

* triplot * was written up in *Stata Journal* 4(2) 2004 and is intermittently updated on SSC.
So, which giant is underneath?

`triplot` rests on `twoway` `scatter`!

You have to work out where the data should go, which is high or secondary school mathematics.

Axes, axis labels and axis titles that would otherwise appear need to be removed with options such as `xsc(off)` and `ysc(off)`.

New axes and grid lines are inserted with `scatteri`, `recast(line)`. 
Circular arguments

Graphs for circular data such as wind direction or time of year may use polar coordinates.

Here I look at some data showing the directions faced by 158 glacial cirques in the English Lake District.

Cirques are armchair-shaped hollows in mountain sides which were deepened by glaciers. 
https://en.wikipedia.org/wiki/Cirque
mean 049° strength 60% $n = 158$
What does that show?

Each spike is proportional to the number of cirques facing that direction.

Naturally, options are available to tune the bin width – and all else too.

An older version of the code for this is accessible with \texttt{ssc type circrplot.ado}
How was that done?

twoway with x and y axes suppressed.

twoway function (twice) for two semi-circles.

twoway pcarrowi for the resultant

twoway pcspike for the spikes.

and so on.

There is no deep object- or class-oriented stuff here, just a series of calls to twoway.
A project in progress

I want students and colleagues to do kernel density estimation my way (at least more often).

That includes

◊ forcing users to set a range
◊ encouraging awareness of choices (kernel, width)
◊ easier display of several results
◊ default display of areas, not lines
◊ (coming) estimation on transformed scales initially
adensity (provisional name)

Names can be tricky: just another density command, or possibly stress display of areas.

More importantly, the strategy is that kdensity does most of the work.
Sample syntax

sysuse auto, clear

adensity mpg, over(foreign) ra(0 50)
area1(color(orange%50))
area2(color(blue%50))
line1(color(orange*2)) line2(color(blue*2))
legend(pos(1) ring(0) col(1))
kdeopts(biweight bw(8)) yla(, ang(h))
Convention and logic

Why do people generally show histograms as areas and kernel density estimates as curves?

Some good reasons might be
◊ simplicity: no more information in the area
◊ avoiding occlusion of multiple densities

But
◊ now we have transparency
◊ areas easier to see as Gestalt?
The moral

In physics, power is the ability to do work.

In programming, power is the ability to get other programs to do most of the work.

So... write wrapper programs!
That's all Folks!
All graphs use Stata scheme *s1color*, which I strongly recommend as a lazy but good default.

This font is Georgia.

This font is Lucida Console.