On the shoulders of giants, or not reinventing the wheel

Nicholas J. Cox
Department of Geography

Durham University
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 useable interactively to those underpinning long programs.
On the shoulders of giants

Robert K. Merton
(1910–2003)

University of Chicago Press.
With gravitas

If I have seen further it is by standing on the sholders 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: documented at [P] `tabdisp`.

But it’s easy: you just need to know or at least calculate in advance what you want to display.

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).
tabdisp

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

The default format for everything but sample size is %9.3f, but that can be overridden.
. 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>21.297</td>
<td>5.786</td>
<td>0.949</td>
<td>3.975</td>
</tr>
<tr>
<td>Price</td>
<td>6165.257</td>
<td>2949.496</td>
<td>1.653</td>
<td>4.819</td>
</tr>
<tr>
<td>Weight (lbs.)</td>
<td>3019.459</td>
<td>777.194</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>
tabdisp

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

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.)
Tabulate cumulative frequencies as well as frequencies?

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 in press).

Another example in distinct (*Stata Journal* 15(3) 2015).
/ tempname mytab
   `mytab' = __tab.new, col(`nc') _margin(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 \texttt{list}, but do you know it well?

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

We can get arbitrarily complicated:

- Row identifiers \hspace{2cm} Cell(s)
- Row and column identifiers \hspace{2cm} Cell(s)
- Many identifiers \hspace{2cm} Cell(s)
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 on SSC. A revised account is forthcoming in *Stata Journal* 17(3) 2017.

At its simplest it looks like `tabulate` in disguise, but it can do other stuff too.
. 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>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>2.90</td>
</tr>
<tr>
<td>Domestic</td>
<td>2</td>
<td>8</td>
<td>11.59</td>
</tr>
<tr>
<td>Domestic</td>
<td>3</td>
<td>27</td>
<td>39.13</td>
</tr>
<tr>
<td>Domestic</td>
<td>4</td>
<td>9</td>
<td>13.04</td>
</tr>
<tr>
<td>Domestic</td>
<td>5</td>
<td>2</td>
<td>2.90</td>
</tr>
<tr>
<td>Foreign</td>
<td>3</td>
<td>3</td>
<td>4.35</td>
</tr>
<tr>
<td>Foreign</td>
<td>4</td>
<td>9</td>
<td>13.04</td>
</tr>
<tr>
<td>Foreign</td>
<td>5</td>
<td>9</td>
<td>13.04</td>
</tr>
</tbody>
</table>
```
. groups foreign rep78, percent(foreign)

+------------------------------------+
| foreign   rep78   Freq.   Percent |
+------------------------------------+
| Domestic  1       2      4.17     |
| Domestic  2       8     16.67    |
| Domestic  3      27     56.25    |
| Domestic  4       9     18.75    |
| Domestic  5       2      4.17    |
| Foreign   3       3     14.29    |
| Foreign   4       9     42.86    |
| Foreign   5       9     42.86    |
+------------------------------------+
```
. groups mpg, select(f == 1) show(none)

+------+
<table>
<thead>
<tr>
<th>mpg</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
</tr>
<tr>
<td>31</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>41</td>
</tr>
</tbody>
</table>
+------+
. 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>

24
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

`sepby(varlist)` separator line if *varlist* values change

`subvarname` characteristic for variable name in header
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 10(1) 2010) hinges on using statsby to produce a dataset (resultssset?) 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.
statsby

statsby is also a natural for confidence interval plots.

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

For designplot see Stata Journal 14(4) 2014 and in press 17(3) 2017.

The idea is 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.
statsby used in designplot

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.
statsby

It would be hard work programming that without the machinery provided by statsby.
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 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.

(Using sunspots as an example is a miniature tradition here, started by William S. Cleveland in his books.)
The line graph shows the mean values over different years, with peaks and troughs occurring in the 18th, 19th, and 20th centuries. The y-axis represents the mean values ranging from 0 to 300, and the x-axis represents the year from 1700 to 2000.
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

gen slice = ceil(4 * _n/_N)

* see how much we zap
line mean year, by(slice, note("")) col(1) xrescale) subtitle("", fcolor(none)) xtitle("") yla(, ang(h)) ytitle(Mean sunspot number)
Mean sunspot number over time.
Fine structure is now visible

Asymmetric saw-tooth cycles are more evident.

Back to the astronomers, climatologists, economists....
by() to get multiple panels cleanly

The same trick, with more work, underlies multiline and multidot (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 \texttt{multiline} the \texttt{subtitle()} option is tricky. Put it on the left and remove traces of surrounding stuff.

\texttt{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.
Unconventional coordinates

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

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

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)`.
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.