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 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).

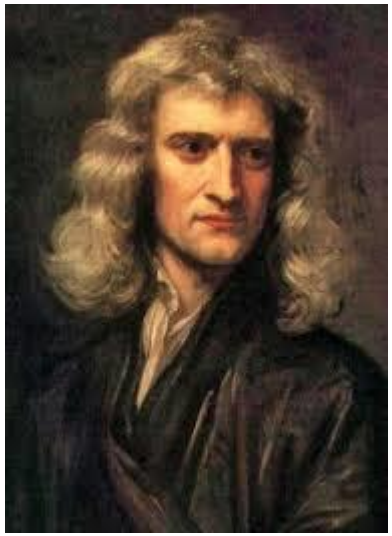
1965/1985/1993.
University of Chicago Press.

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. 1978. *Catastrophe Theory and its Applications*. London: Pitman, p.v

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

------------------------------------------------------------------
       n = 74 |       mean          SD    skewness    kurtosis
--------------+---------------------------------------------------
Mileage (mpg) |     21.297       5.786       0.949       3.975
        Price |   6165.257    2949.496       1.653       4.819
Weight (lbs.) |   3019.459     777.194       0.148       2.118
------------------------------------------------------------------

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

------------------------------------------------------------------
       n = 74 |       mean          SD    skewness    kurtosis
--------------+---------------------------------------------------
Mileage (mpg) |       21.3         5.8       0.949       3.975
        Price |     6165.3      2949.5       1.653       4.819
Weight (lbs.) |     3019.5       777.2       0.148       2.118
------------------------------------------------------------------
```

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` 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?

```
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)
```

http://www.stata.com/support/faqs/data-management/tabulating-cumulative-frequencies/

`_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
```

`list`

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 Cell(s)

Row and column identifiers Cell(s)

Many identifiers 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.

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

  +-----------------------------------+
  |  foreign   Freq.   Percent     % <= |
  |-----------------------------------|
  | Domestic      52     70.27    70.27 |
  |  Foreign      22     29.73   100.00 |
  +-----------------------------------+
```

```
. groups foreign rep78
```

foreign	rep78	Freq.	Percent
Domestic	1	2	2.90
Domestic	2	8	11.59
Domestic	3	27	39.13
Domestic	4	9	13.04
Domestic	5	2	2.90
Foreign	3	3	4.35
Foreign	4	9	13.04
Foreign	5	9	13.04

```
. 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)

  +-----+
  | mpg |
  |-----|
  |  29 |
  |  31 |
  |  34 |
  |  41 |
  +-----+
```

```
. groups mpg, select(-5)

  +-------------------------------------+
  |  mpg    Freq.    Percent      Cum.  |
  |-------------------------------------|
  |   30        2       2.70     93.24  |
  |   31        1       1.35     94.59  |
  |   34        1       1.35     95.95  |
  |   35        2       2.70     98.65  |
  |   41        1       1.35    100.00  |
  +-------------------------------------+
```

```
. groups mpg, select(5) order(h)

  +-----------------------------------+
  | mpg   Freq.   Percent     Cum.    |
  |-----------------------------------|
  |  18       9     12.16    12.16    |
  |  19       8     10.81    22.97    |
  |  14       6      8.11    31.08    |
  |  21       5      6.76    37.84    |
  |  22       5      6.76    44.59    |
  +-----------------------------------+
```

`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)
`sepby(`*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?

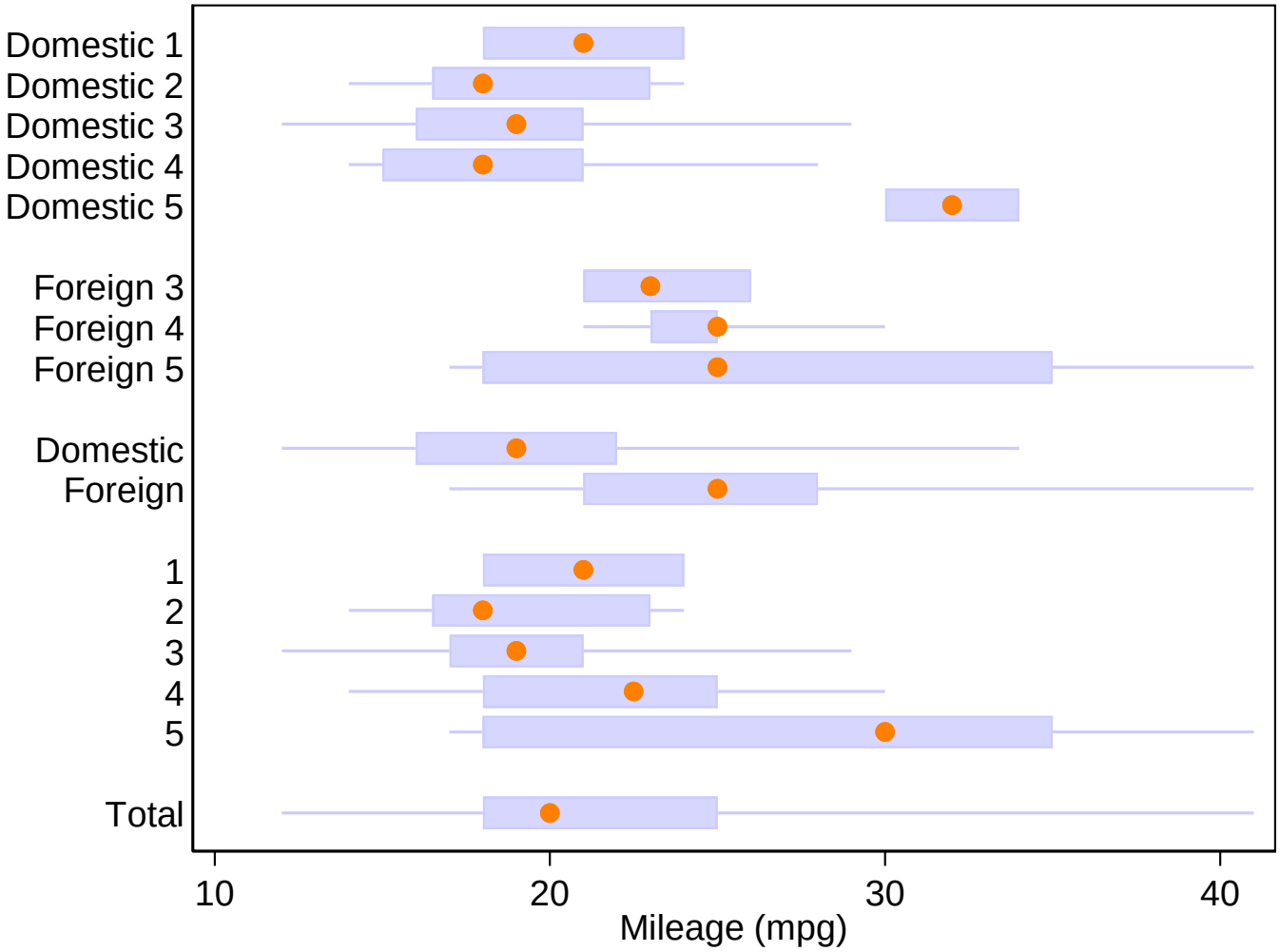
statsby

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

The `statsby` strategy (*Stata Journal* 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`.


Domestic 1
Domestic 2
Domestic 3
Domestic 4
Domestic 5
Foreign 3
Foreign 4
Foreign 5
Domestic
Foreign
1
2
3
4
5
Total
10
20
30
40
Mileage (mpg)

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`.

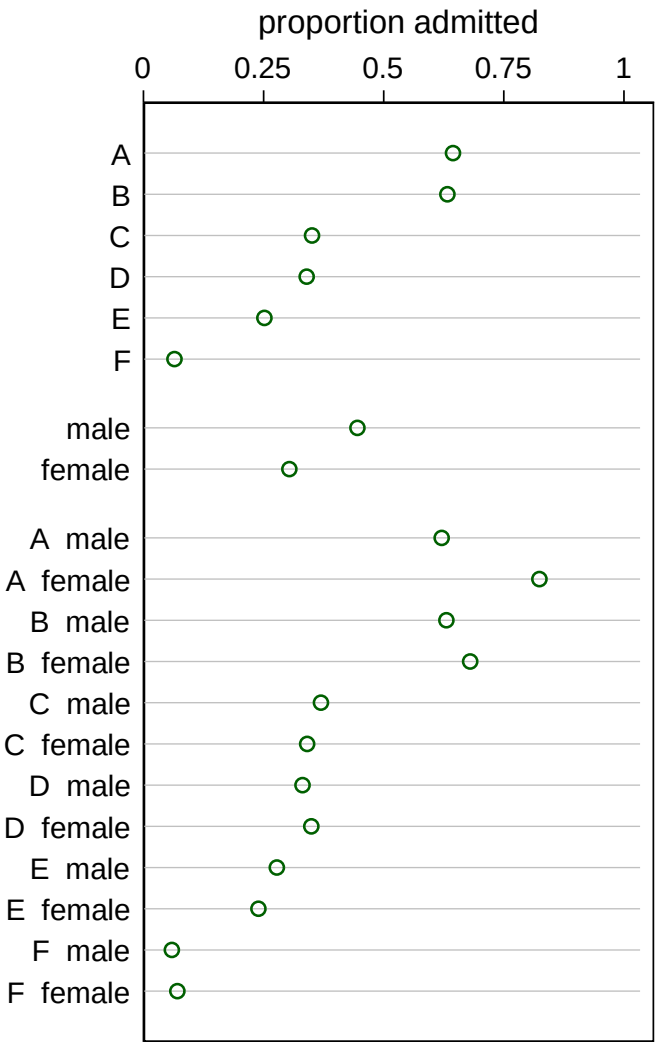
`statsby` used in `designplot`

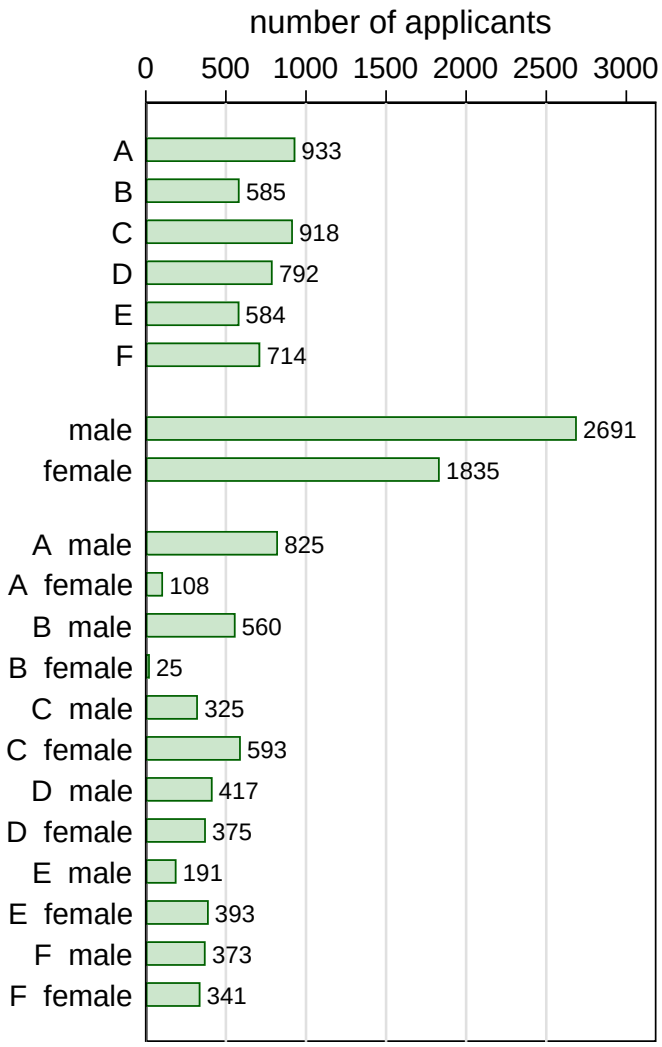
As before `subsets` and `total` options are key.

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.


proportion admitted
0
0.25
0.5
0.75
1
A
B
C
D
E
F
male
female
A male
A female
B male
B female
C male
C female
D male
D female
E male
E female
F male
F female

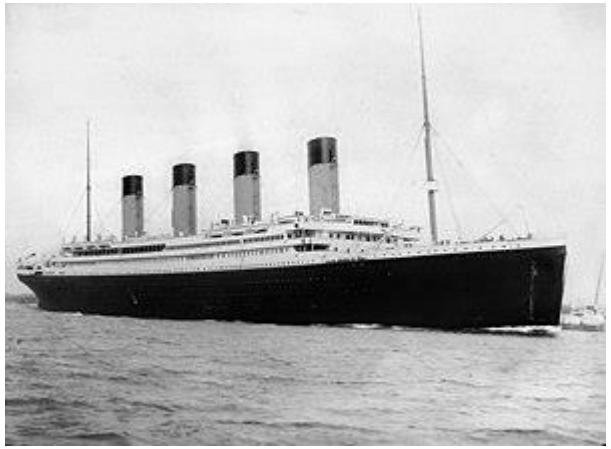

number of applicants
0
500
1000
1500
2000
2500
3000
A
933
B
585
C
918
D
792
E
584
F
714
male
2691
female
1835
A male
825
A female
108
B male
560
B female
25
C male
325
C female
593
D male
417
D female
375
E male
191
E female
393
F male
373
F female
341

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.

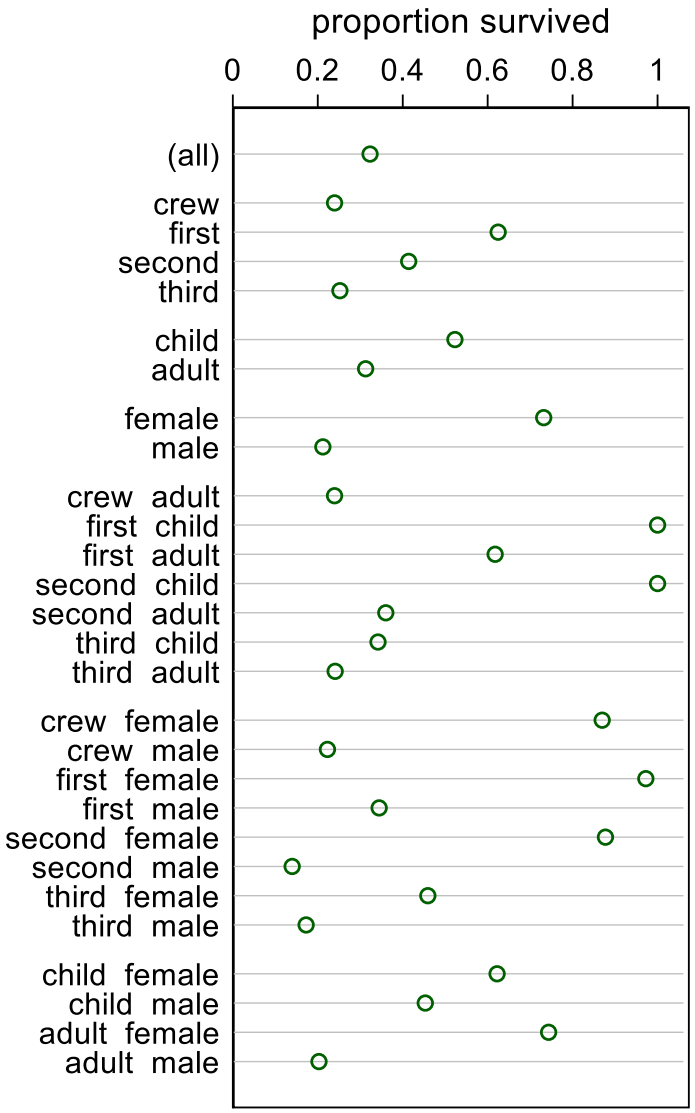
In a nutshell: patterns can appear on amalgamation that are just side-effects of other aspects of the data, here the group frequencies.

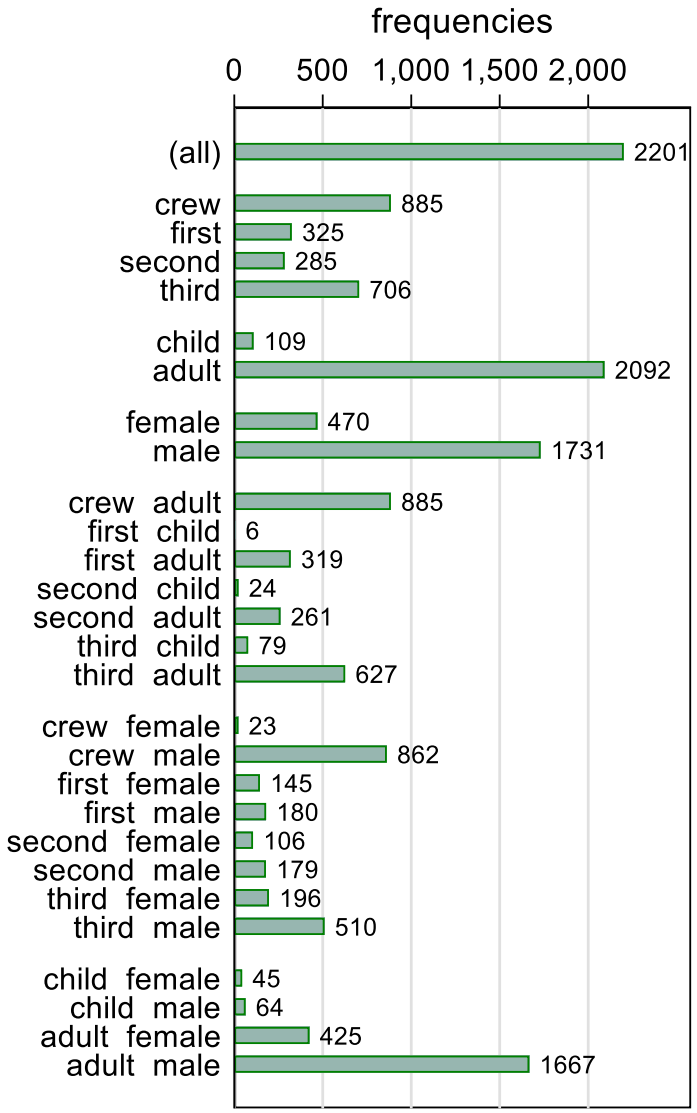
Another example



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

Some of the people on board survived….


proportion survived
0
0.2
0.4
0.6
0.8
1
(all)
crew
first
second
third
child
adult
female
male
crew adult
first child
first adult
second child
second adult
third child
third adult
crew female
crew male
first female
first male
second female
second male
third female
third male
child female
child male
adult female
adult male


frequencies
0
500
1,000
1,500
2,000
(all) 2201
crew 885
first 325
second 285
third 706
child 109
adult 2092
female 470
male 1731
crew adult 885
first child 6
first adult 319
second child 24
second adult 261
third child 79
third adult 627
crew female 23
crew male 862
first female 145
first male 180
second female 106
second male 179
third female 196
third male 510
child female 45
child male 64
adult female 425
adult male 1667

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 `corrtable` 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.

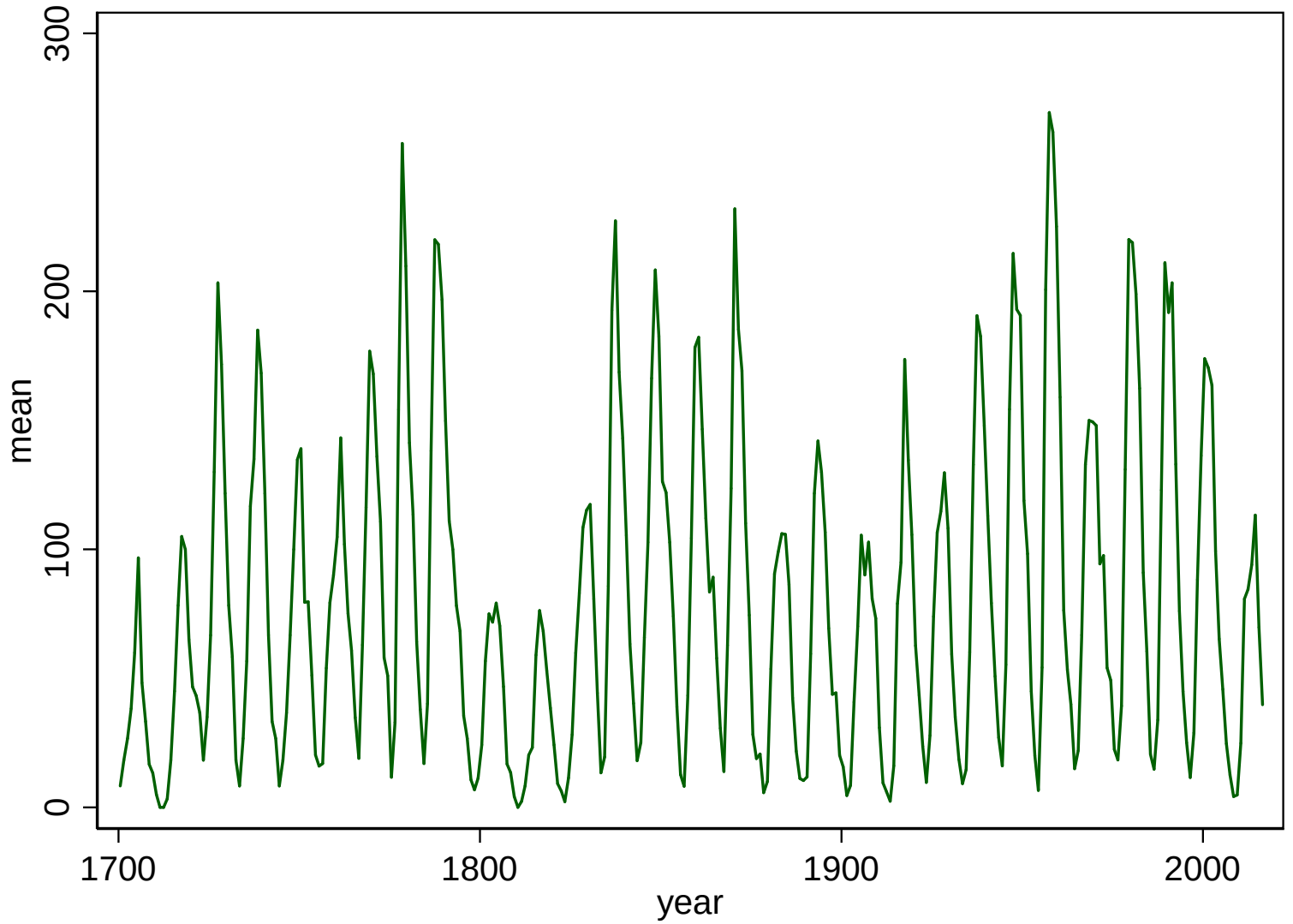
Counting sunspots

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.)


mean
0
100
200
300
1700
1800
1900
2000
year

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.

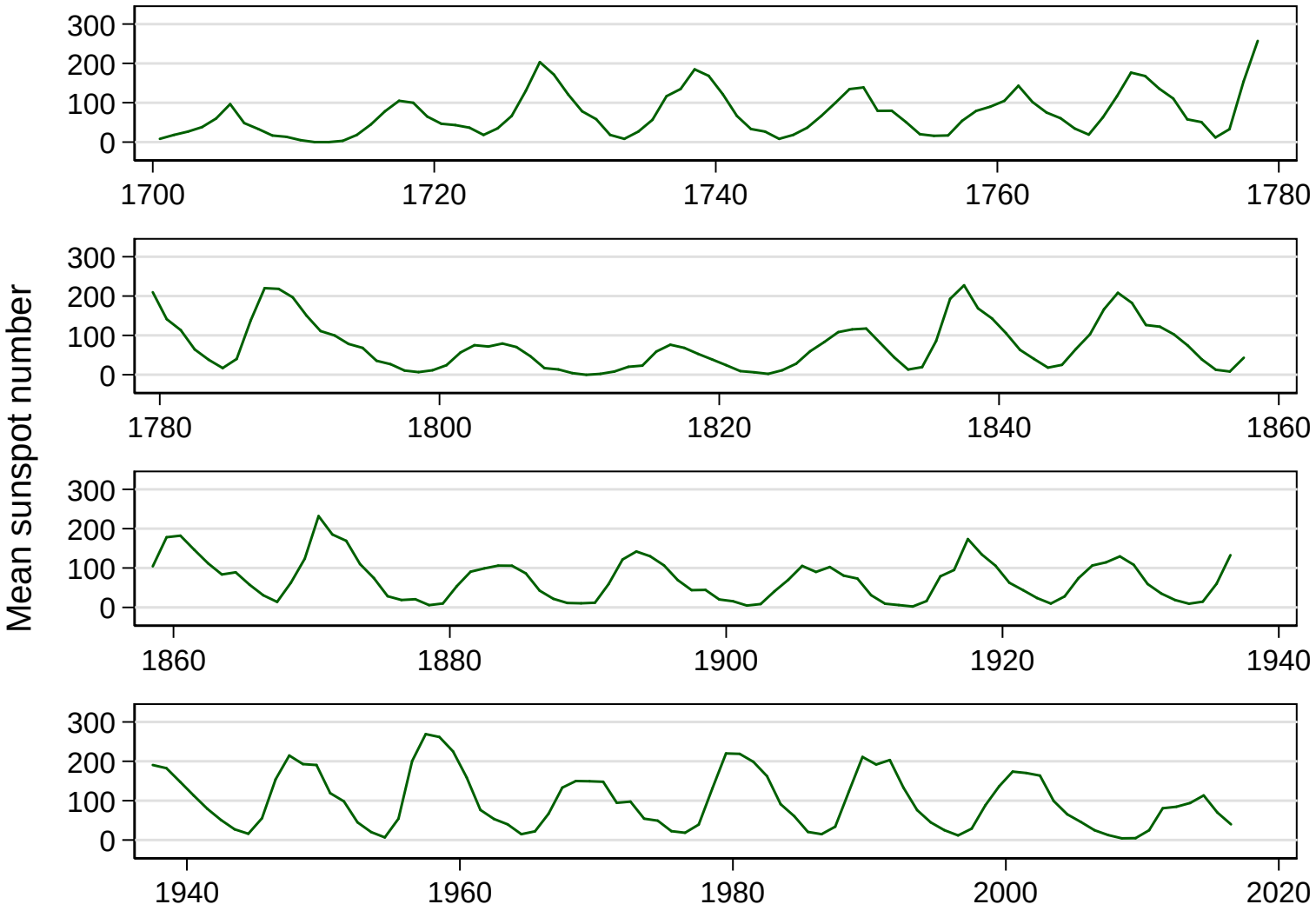
An even older device was to have very wide graphs as fold-out illustrations.

`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
300
200
100
0
1700
1720
1740
1760
1780
1780
1800
1820
1840
1860
1860
1880
1900
1920
1940
1940
1960
1980
2000
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 `_n/_N` is the current observation number `_n` divided by the total number of observations `_N`.

So it rises from `1/_N` to `_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.

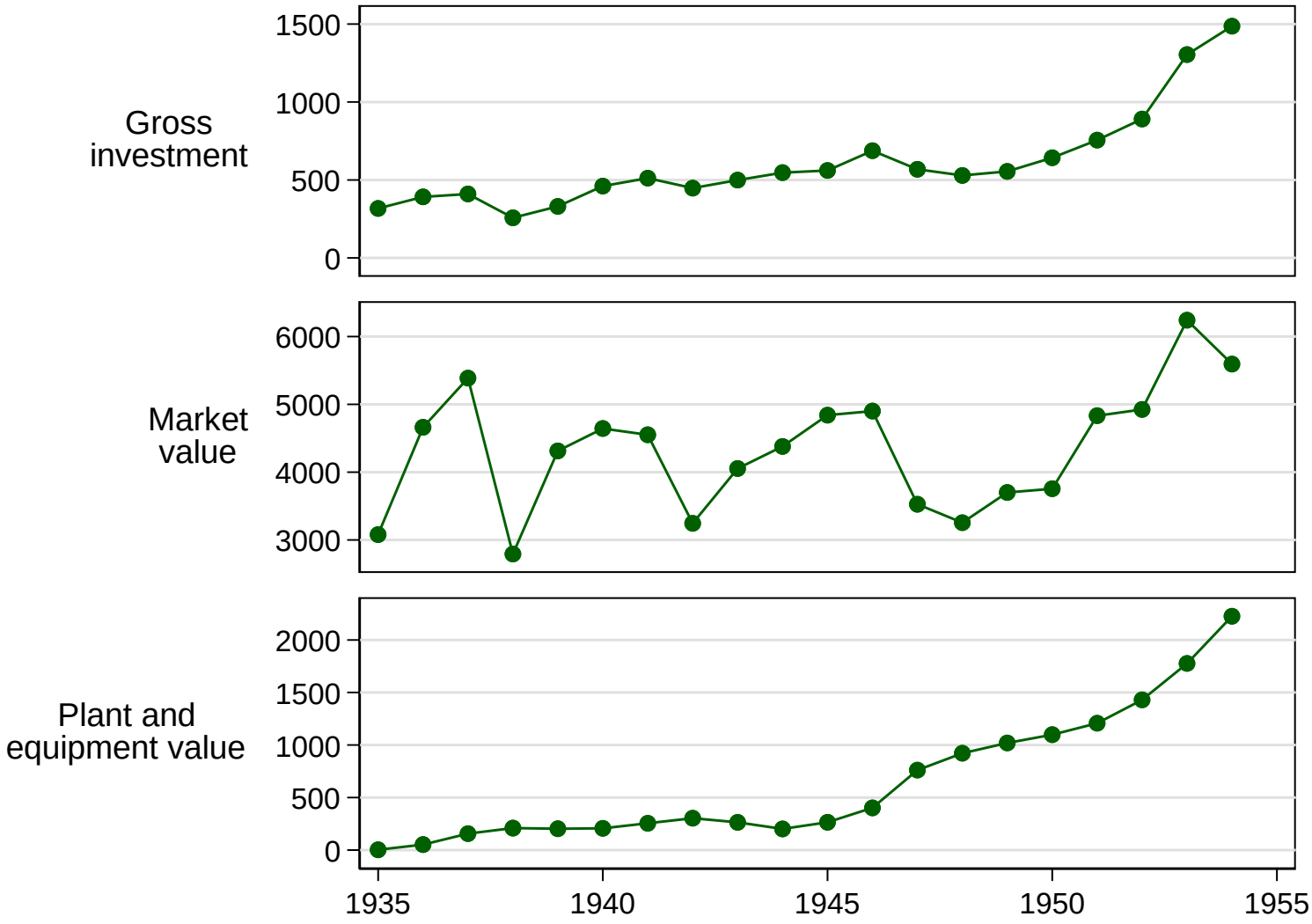
by() to get multiple panels cleanly

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).


Gross investment
1500
1000
500
0
Market value
6000
5000
4000
3000
Plant and equipment value
2000
1500
1000
500
0
1935
1940
1945
1950
1955

`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.)

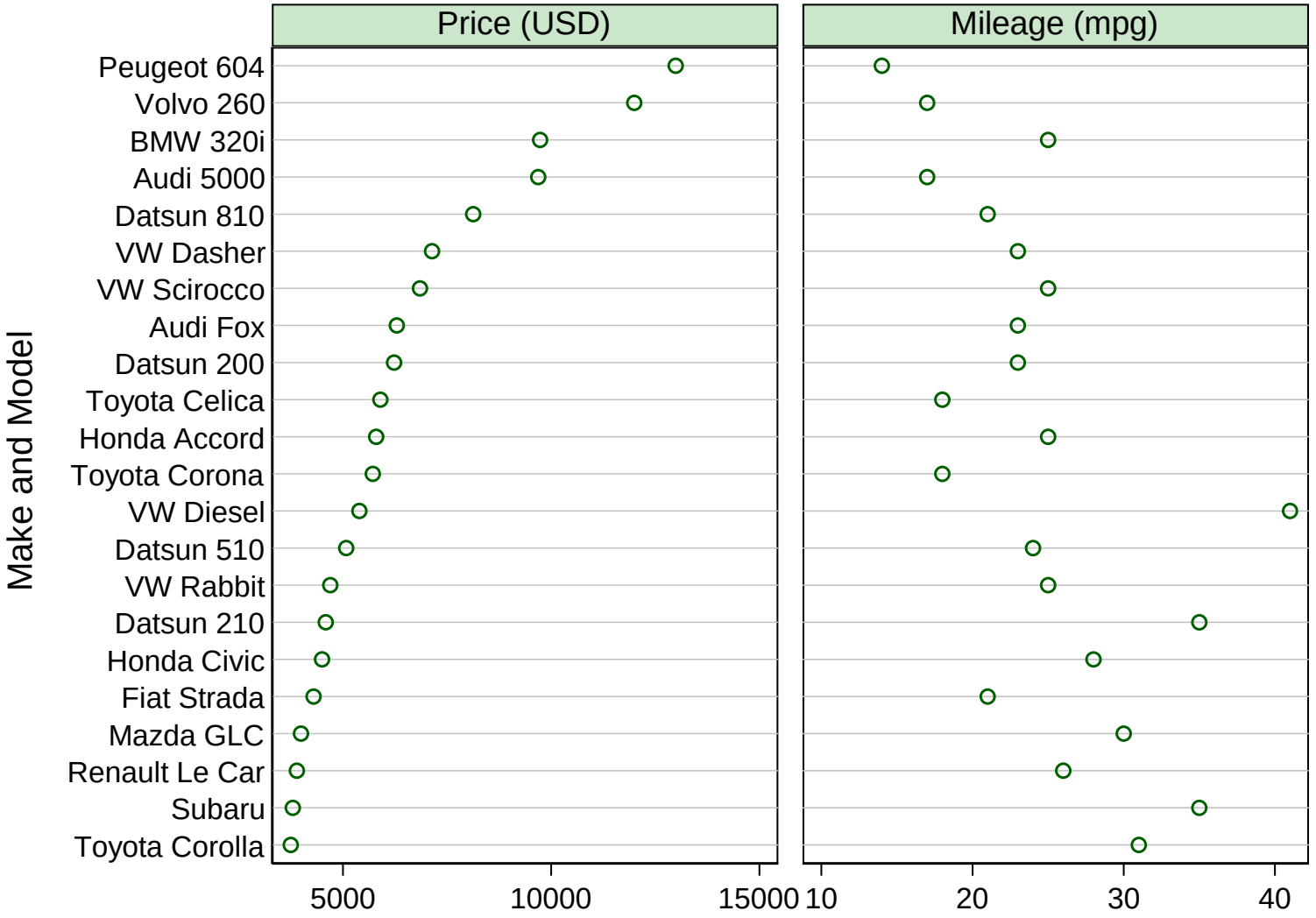
multidot

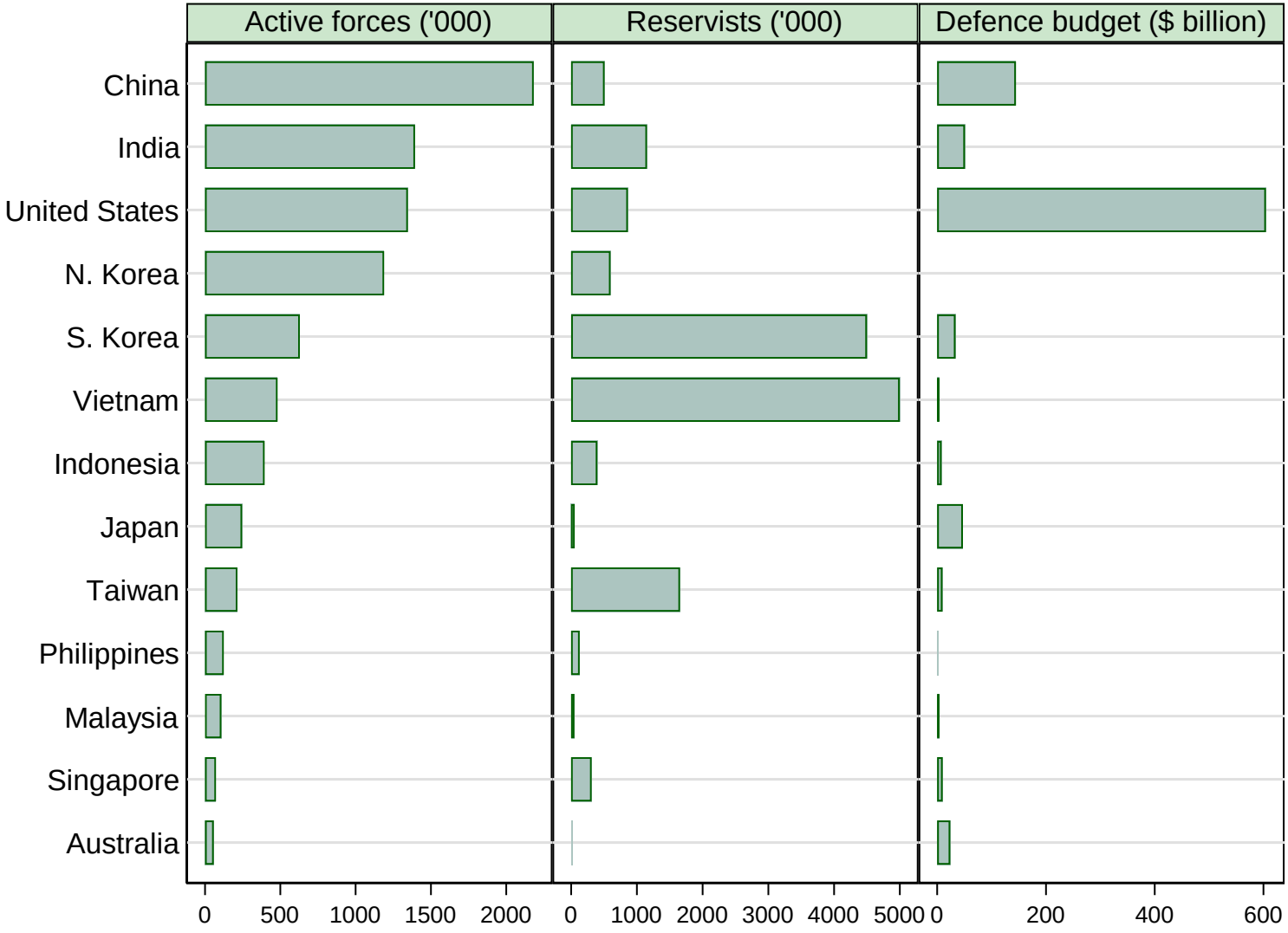
multidot is also a solution when multiple responses have different scales and/or units.

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.


Price (USD)
Mileage (mpg)
Make and Model
Peugeot 604
Volvo 260
BMW 320i
Audi 5000
Datsun 810
VW Dasher
VW Scirocco
Audi Fox
Datsun 200
Toyota Celica
Honda Accord
Toyota Corona
VW Diesel
Datsun 510
VW Rabbit
Datsun 210
Honda Civic
Fiat Strada
Mazda GLC
Renault Le Car
Subaru
Toyota Corolla
5000
10000
15000
10
20
30
40


Active forces ('000)
Reservists ('000)
Defence budget ($ billion)
China
India
United States
N. Korea
S. Korea
Vietnam
Indonesia
Japan
Taiwan
Philippines
Malaysia
Singapore
Australia
0
500
1000
1500
2000
0
1000
2000
3000
4000
5000
0
200
400
600

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.

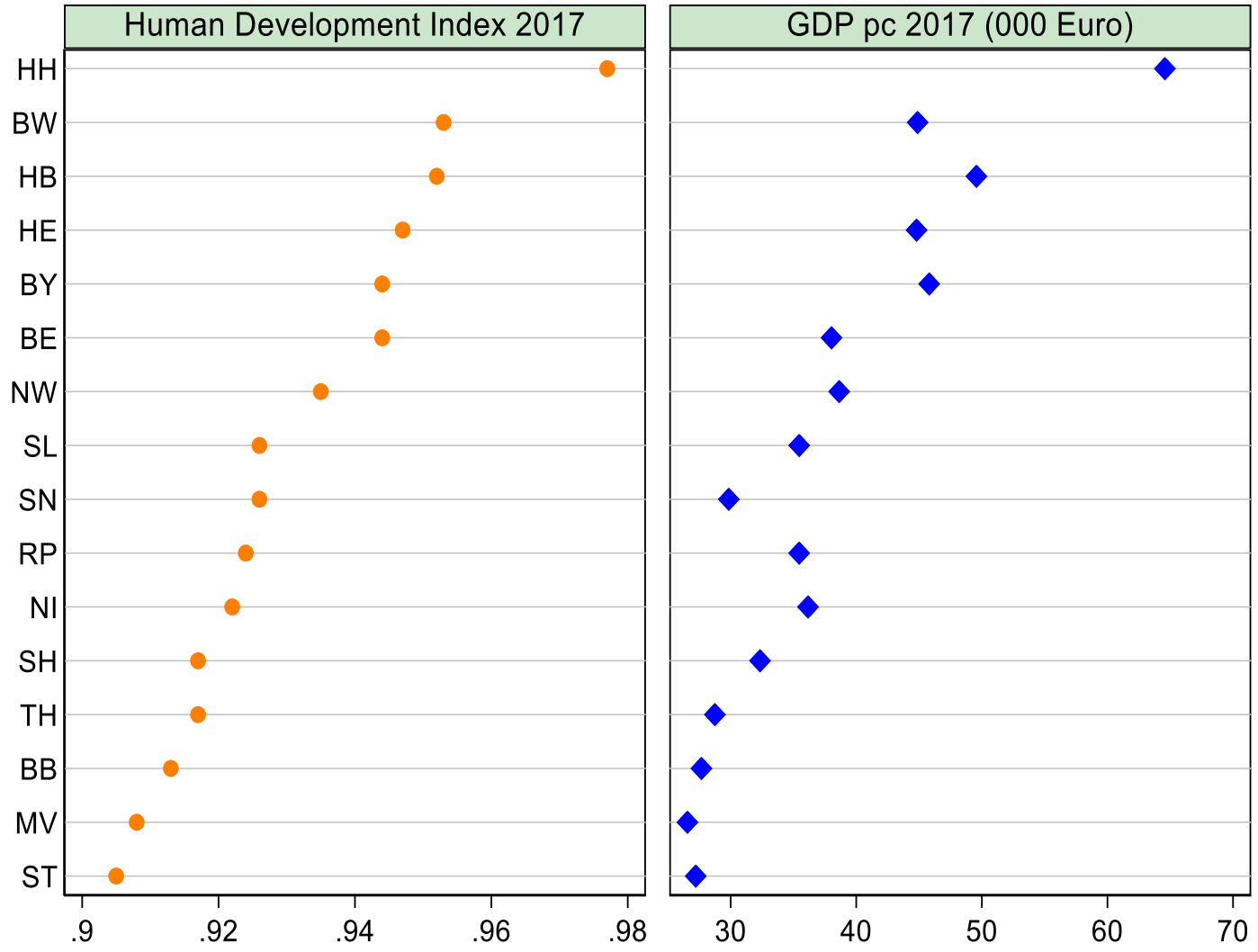
Be that as it may, graphs with table flavour often look better with horizontal axis information at the top.

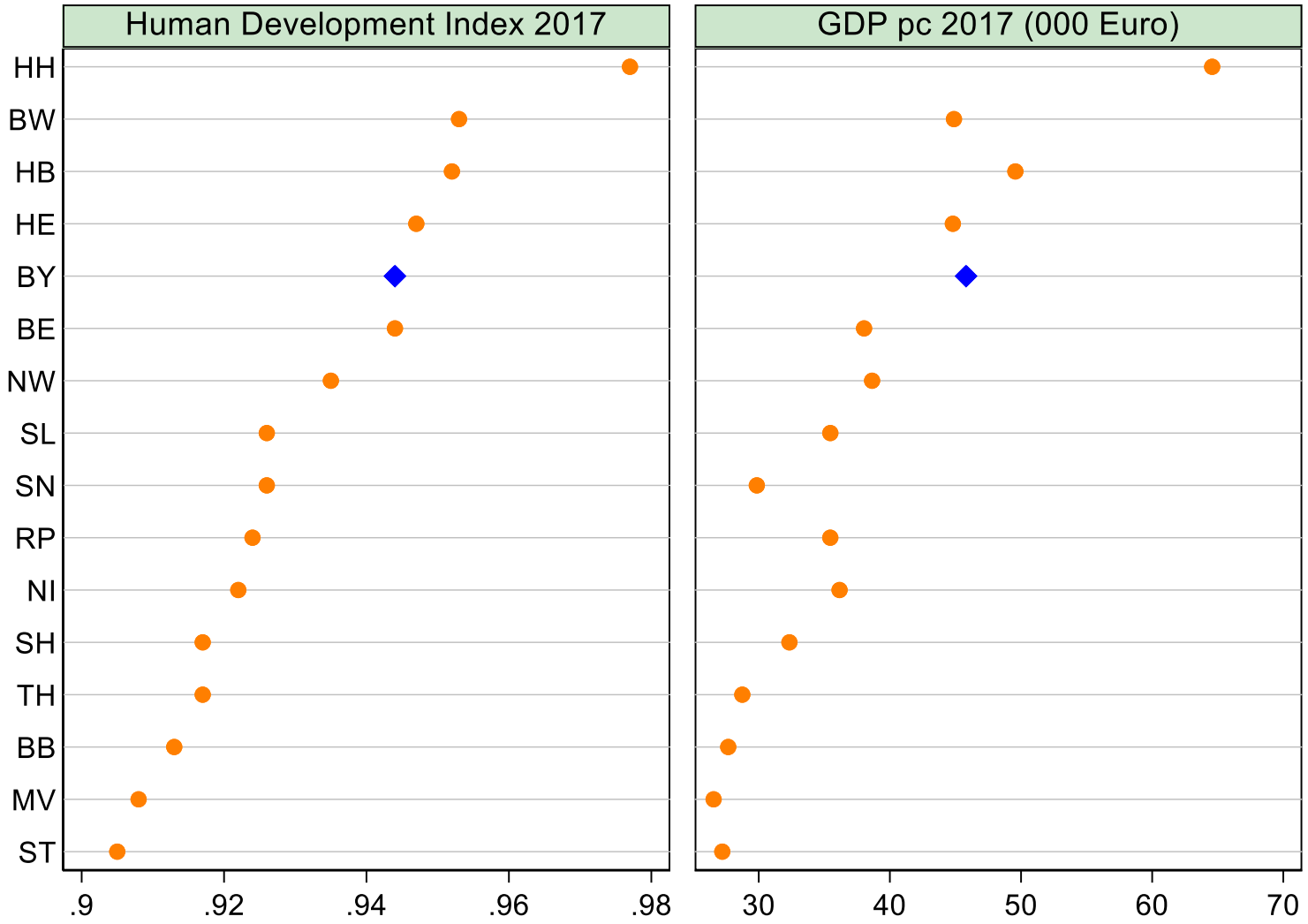
Another example: 16 Länder

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.


Human Development Index 2017
GDP pc 2017 (000 Euro)
HH
BW
HB
HE
BY
BE
NW
SL
SN
RP
NI
SH
TH
BB
MV
ST
.9
.92
.94
.96
.98
30
40
50
60
70


Human Development Index 2017
GDP pc 2017 (000 Euro)
HH
BW
HB
HE
BY
BE
NW
SL
SN
RP
NI
SH
TH
BB
MV
ST
.9
.92
.94
.96
.98
30
40
50
60
70

Highlight just one subset

A minimalist principle is to highlight just one subset.

"when *everything* is emphasized, *nothing* is emphasized"
Edward R. Tufte. 1997. *Visual explanations: images and quantities, evidence and narrative,* p.74.

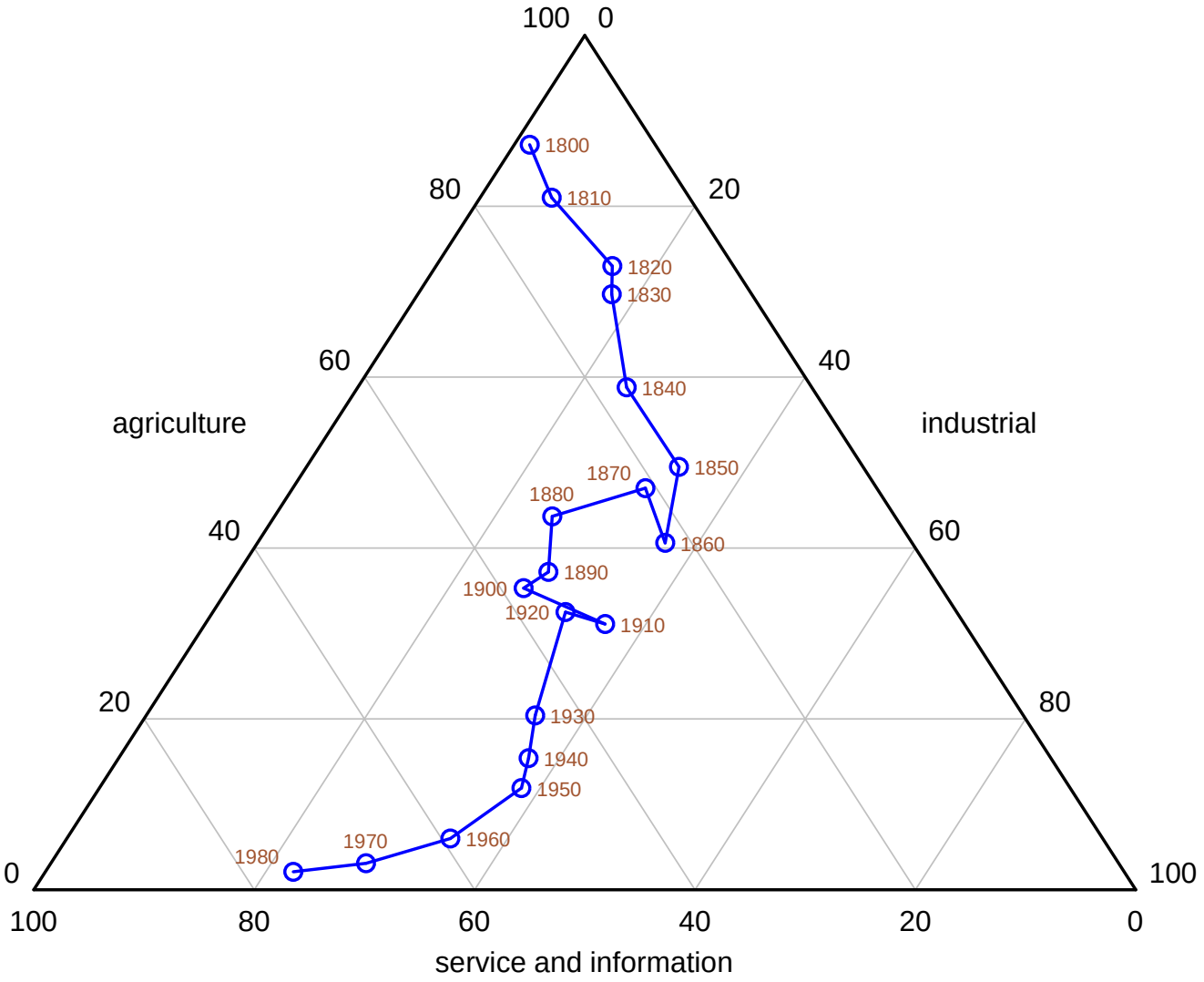
Think: Would you colour each sentence word differently?
(Credit for example: Naomi B. Robbins.)

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 Journ*al 4(2) 2004 and is intermittently updated on SSC.


100
0
80
20
60
40
40
60
20
80
0
100
agriculture
industrial
service and information
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980

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)`.

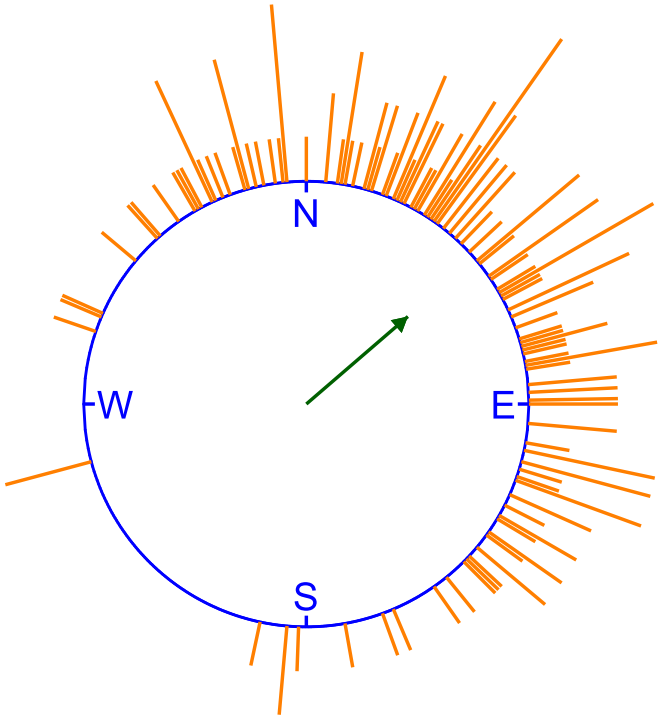
New axes and grid lines are inserted with `scatteri, recast(line)`.

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

```
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)

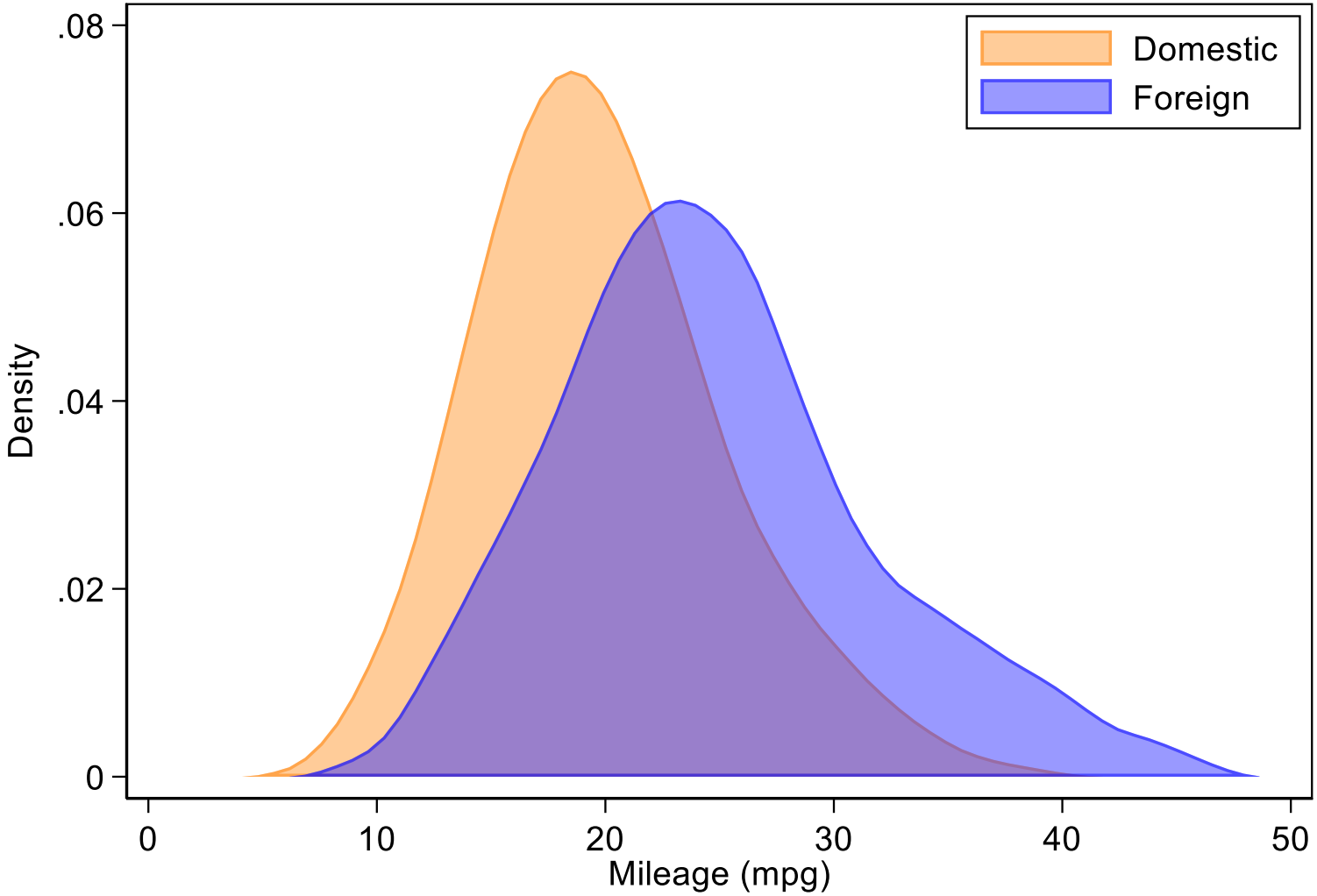
Names can be tricky: just another density command, or possibly stress display of areas.

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))
```



biweight: width 8

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.`