The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–9

A note on creating inset plots using graph twoway

Matthew Tibbles
London
mawtibbles@gmail.com

Eric Melse
Amsterdam University of Applied Sciences
eric.melse@planet.nl

Abstract. Inset plots can be used to "zoom-in" on densely populated areas of a graph or to add extra relevant data in the form of, for example, distribution plots. The standard Stata command for combining plots, `graph combine`, does not, however, permit this type of seamless integration. Each plot within a `graph combine` object is allocated a grid cell, which cannot be placed within another grid cell - at least not without ceartain (invariably unwanted) graphical complications. We present a fairly simple workaround to this issue using reproducible examples. The main idea is to plot insets along a second axis, and to artificially modify the range of this axis in order to constrain the inset plot within a specified area of the main graph. Additional tips are included for producing more intricate, multi-layered inset graphs.

Keywords: st0001, twoway, graph combine, inset plots, scatterplots, graphics

1 Introduction

An inset is defined as 'a small map or picture that is shown on or next to a larger map or picture in order to show more detail' (Merriam-Webster). William Shakespeare supposedly used five types of inset as an 'episode that is at odds with [the] actual spectacle', but seemingly required to appreciate his play (Berry 1965). An Internet search result list on inset will contain mostly technical references about 'how to' create and plot an inset, suggesting that it is merely a data visualization gimmick. We argue, however, that insets can - and should - be used to support purposive communication. In scientific publications an inset might be a diagram or an image, as well as a graph or plot that nudges the eye onto meaningful details or a supportive side story on top of the main message (Zhuang et al. 2021, Wang et al. 2021).[1]

Inset plots, in particular, are most typically used to "zoom-in" on data that is densely clustered; though they may also be used to view variable distributions, anamolies (without the effects of visual compression) or additional relevant data. In Stata, it is (at the very least) difficult to superimpose one graph on top of another due to the underlying grid architecture of `graph combine` - the standard command for combining multiple graphs. Each graph within a `graph combine` object is allocated a grid cell, which cannot be placed within another grid cell unless the user is willing to make some rather large and, in our opinion, unacceptable compromises in terms of scale and aesthetic. In spite of this, Stata graphics are highly flexible, permitting a single `twoway`

1. Users of Stata have long expressed interest in functionality to inset plots and/or images, e.g. slide 42 in Naqvi (2021).

graph to use up to 9 x, y and z axes (Wiggins 2010). Through the use of `graph twoway`, it is thus possible to add inset plots to any `graph twoway` object by creating and plotting along one or more additional sets of axes.

In what follows, we describe this method for creating inset plots in Stata through reference to general, reproducible examples. We first demonstrate the advantages of our method - which is to modify the range of the added axes so that the inset is constrained within the main plot - as compared to the more "routine" approach given (albeit quite unwillingly) by `graph combine`. We then explore the broader functionality of this method by plotting multiple insets, as well as adding `twoway` objects to link the inset plots to the main graph.

2 The basic idea

As a general working example, we use climate data on 956 U.S. cities. We will examine the relationship between average July temperature, `tempjuly`, and Cooling Degree Days (CDD), `cooldd`, which is the difference between mean July temperature and 65°F. In order to accentuate the advantages of inset plotting, we `separate(tempjuly)` by `region`. Following (Cox 2016), we use local macros to store commands which require multiple reuse.

```
webuse citytemp4, clear
separate(tempjuly), by(region)
label variable tempjuly1 "NE"
label variable tempjuly2 "N Cntrl"
label variable tempjuly3 "South"
label variable tempjuly4 "West"
local mops msymbol(O D S T) msize(1.2 1.2 1.2 1.2) mfcolor(gs0 gs5 gs10 gs15)    ///
mlcolor(gs0 gs0 gs0 gs0)  mlwidth(.05 .05 .05 .05)
local lops legend(order(0 "Region" 1 2 3 4) rows(1))
local labs xtitle("Cooling Degree Days (CDD)") ytitle("Average July temperature")
scatter tempjuly1 tempjuly2 tempjuly3 tempjuly4 cooldd, `mops' `lops' `labs'
```

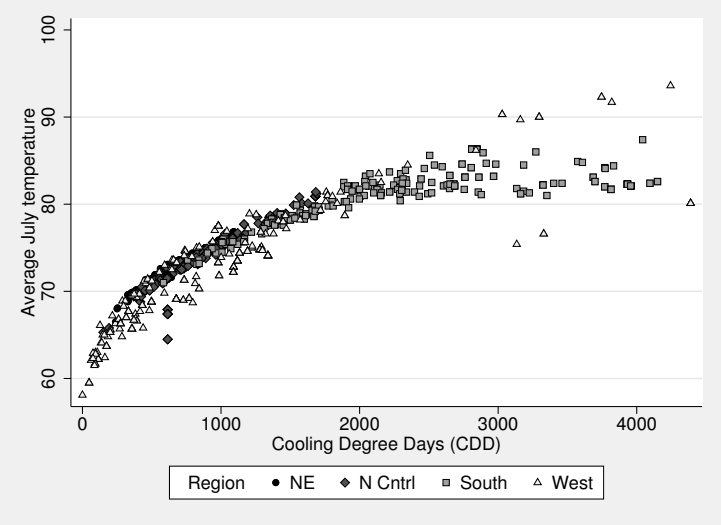
Figure 1 shows the relationship between average July tempature and CDD by region for 956 U.S. cities. North East and North Central cities are densely clustered between 70°F and 75°F on the y axis, while visibility is further impaired by a scattering of South and West cities. To increase visibility, we may want to add an inset plot; one which "zooms-in" on this densely populated area of the graph.

The natural starting point is to create the inset plot as a separate graph object. Using the `if` qualifier we restrict the plotting range on the y axis to between 70°F and 75°F. We additionally limit the x range to 950 CDD at the upper limit in order to exclude a handful of West cities which would otherwise result in visual compression of the points of interest. We use the forced size options, `fysize(20)` and `fxsize(40)`, to control the scale of the inset relative to the main graph. We also set the color of the outer graph region to white and margins to zero so that the inset blends into the main plot region:

```
local iops xlab(,labsize(2)) ylab(,labsize(2)) ytitle("") xtitle("")
scatter tempjuly1 tempjuly2 tempjuly3 tempjuly4 cooldd                    ///
```

```
if inrange(tempjuly, 70, 75) & cooldd <= 950, 'mops' 'labs' 'iops' legend(off)  ///
fysize(20) fxsize(40)  graphregion(color(white) margin(0)) name(inset)
```

Figure 1: Scatter plot of average July temperature and CDD by region.



We then combine the graphs using `graph combine`:

```
graph combine Graph inset, holes(2) imargin(0 0 0 0) name(grc, replace)
```

Before finally moving the inset into a smaller grid cell using `grid edit`, and moving the cell containing the inset into the main graph:

```
_gm_edit .grc.plotregion1.move graph2 rightof 1 1
_gm_edit .grc.plotregion1.graph2.DragBy 40 -45
graph display grc
```

Figure 2 shows the result of our first attempt. What immediately stands out is that the combined plot has been pushed toward the upper left, with blank grid space encompassing more than a third of the overall graph region. Aesthetically, this lopsided arrangement looks rather strange. But for us the more salient concern is that the data is now difficult to see and, in turn, make sense of.

The obvious fix is to reduce the amount of unused graph space by expanding the main cell and and adjusting the position of the inset cell:

```
_gm_edit .grc.plotregion1.Expand graph1 right 1
_gm_edit .grc.plotregion1.Expand graph1 bottom 1
_gm_edit .grc.plotregion1.graph2.DragBy -5 85
graph display grc
```

Yet this introduces a new problem. As seen in Figure 3, the grid cell of the inset plot now extends beyond the plot region of the graph, resulting in the intrusion of white grid space upon the plot border and graph region. Furthermore, the command we use

Figure 2: Scatter plot of average July temperature and CDD by region; an inset plot is included to aid visibility of a densely clustered group of North and North Central cities where average July temperature falls between 70°F and 75°F.

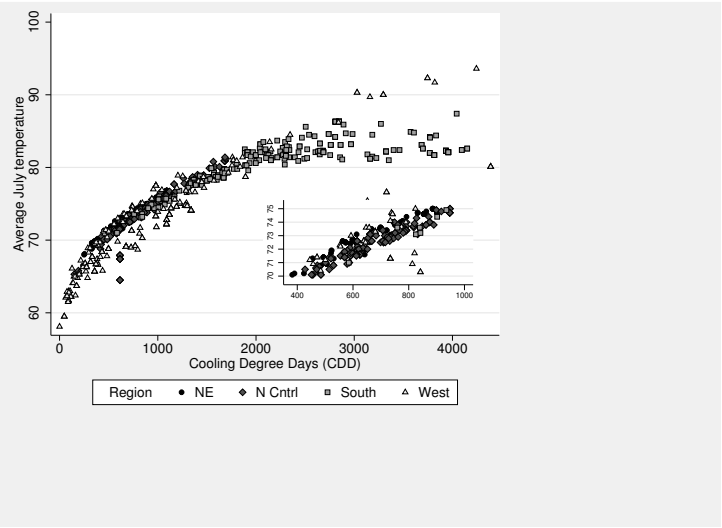
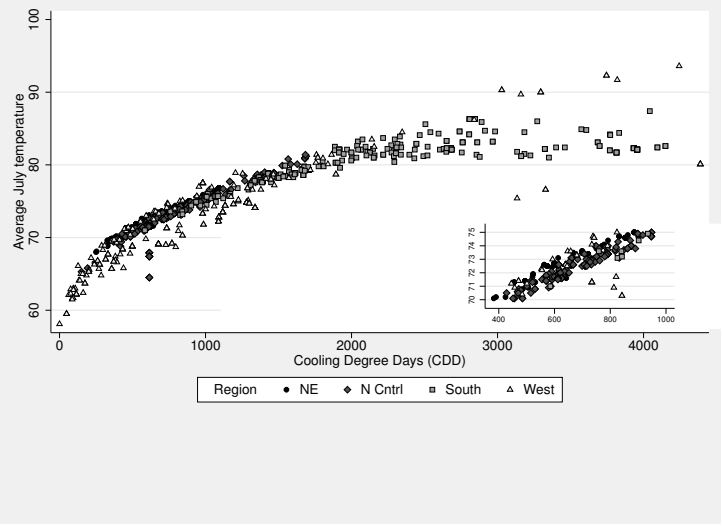


Figure 3: A considerable reduction in the amount of unused graph space is achieved by using the grid editor to adjust the grid layout used for Figure 2. The alignment of the plot area is far from optimal, however, given that about 21% of the graph area is still unused and that the plot area of the inset now extends rightward, obfuscating the right-hand side of the graph.



to expand the main graph cell downward has zero effect (presumably because the cell

that we want to expand into is reserved for the inset). The upshot is that there is still a sizeable area of blank grid space spanning the bottom of the graph region. Of course, it is possible to remove this padding using external software; it has even been suggested (not unreasonably considering the above) that the most efficient way of adding an inset plot to a Stata graph is to export the entire process.[2] In our opinion, though, this approach is severely limiting. For one, it removes the possibility of using `graph combine` to add, for example, multiple inset graphs together. Perhaps more importantly for many Stata users, it would also mean that our graph would no longer be reproducible. In short, what would seem to be a fairly routine approach to creating inset plots in Stata is, in fact, highly inflexible and functionally unsuitable for producing high-quality graphical ouput. For these reasons, we advise that users adopt an alternate approach; one which requires, perhaps, a little more leg-work, but which yields far superior, publication-quality results that are also reproducible.

Our alternate method for creating inset plots exploits the capability that is given by `graph twoway` to create a single graph comprised of multiple plots. The first step - after adding a second `twoway` scatter plot to the initial `twoway` call - is to limit the plotting range via the `if` qualifier. Next, we create and plot along a new set of axes by specifying `yaxis(2)` and `xaxis(2)`. The "trick" is then to artifically modify the range of each new axis so that the resulting plot is constrained within an area of the main plot. We opt to add the inset at the bottom, right-of-centre. To do so, we limit the y axis range to between 69°F and 90°F via `ylabel(69 90, axis(2))`. The x axis range is limited to between -300 and 1100 CDD via `xlabel(-300 1100, axis(2))`. We then make the inset axes invisible via `yscale(axis(2)off)` and `xscale(axis(2)off)` and specify the (undocumented) `norescaling` option of `graph twoway` in order to preserve the axis range/labels of the main plot. The final step is to add a border around the inset plot using the main graph axes. We do this via `scatteri` - which accepts multiple paired-coordinates as inputs - with the option `recast(line)`.

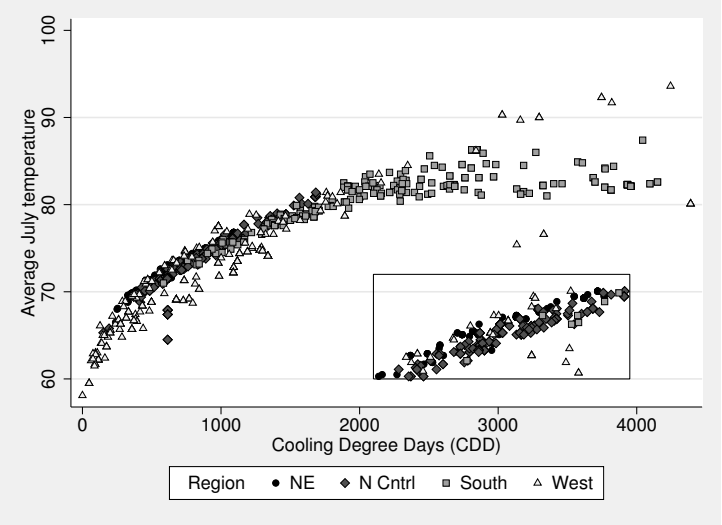
```
local bops lpattern(solid) lwidth(thin) lcolor(black)
graph twoway scatter tempjuly1 tempjuly2 tempjuly3 tempjuly4  cooldd,          ///
‘mops’ ‘lops’ ‘labs’                                                          ///
|| scatter tempjuly1 tempjuly2 tempjuly3 tempjuly4 cooldd                     ///
if inrange(tempjuly, 70, 75) & cooldd <= 950, ‘mops’                          ///
yaxis(2) xaxis(2)  ylabel(69 90, axis(2)) xlabel(-300 1100, axis(2))          ///
yscale(axis(2)off) xscale(axis(2)off) norescaling                             ///
|| scatteri 60 2100 72 2100 72 3950 60 3950 60 2100, recast(line) ‘bops’
```

Finding an appropriate range for the constrained inset axes is a simple though, admittedly, iterative process. For the example below, we opted to add the inset at the bottom of the main graph. As such, the lower limit on the y axis needed to be approximately equal to the lowest `tempjuly` value found within the inset sample. Conversely, the upper limit needed to be considerably higher than the highest `tempjuly` inset value so as to constrain the inset plot within the bottom third of the main graph. After a couple of near-misses - in which the inset took up too much vertical space - we landed on 69°F and 90°F for the lower and upper limits respectively (Figure 4).

2. See the following discussion on the Stata forum: https://www.statalist.org/forums/forum/general-stata-discussion/general/1594095-insetting-graphs-using-stata.

Compared to the various complications resulting from the more routine approach, our method seemlessly intregrates the inset plot within the main graph. The one limitation that users may have adjust for is that it is not possible to produce inset plots outside of `graph twoway`. In our opinion, this is a workable compromise considering the capability and range of plotting options given by `graph twoway`.

Figure 4: Scatter plot of average July temperature and CDD by region; an inset plot is included to aid visibility of a densely clustered group of North and North Central cities where average July temperature falls between 70°F and 75°F. The result of using our alternate method is that the plot area is optimally aligned within the graph area.



3 Extra details and multiple insets

We now turn to slightly more sophisticated usage of these plots. In the first instance, it is difficult to determine precisely which area of the main graph is plotted within the inset. Using `scatteri` and option `recast(line)`, we draw a border around the inset area using the main graph axes. We add connecting lines between this border and that of the inset, as well as ticks and labels on the inset axes using `pci` and `scatteri` with option `text`. Now, suppose that there is some particular interest in CDD at specific average July temperatures - say, between 70°F and 72°F. One way to draw attention to this y range is to highlight it on both the main graph and the inset. Using `scatteri` and option `recast(area)`, we shade the horizontal zone between 70°F and 72°F using the x axis of the main graph - via `xaxis(1)` - and the x axis of the inset - via `xaxis(2)`. This is done prior to `scatter` so that the shaded zones do not blur the scatter points (Cox 2016). Finally, to (further) aid interpretation, it is generally a good idea to plot the distribution of each variable. The standard approach is to use `graph combine` (see Ängquist 2014), however inset plotting offers a slightly more flexible, space-saving alternative. Using the

(undocumented) command `twoway__histogram_gen` (see Harrison 2005), we generate histogram bin variables by `region` and subtract an incremental constant from the x bins of each region (`x1 - x4`) to prevent overlap. We plot the histogram across the full width of the main graph using a slightly adjusted x axis to control the alignment of bins vis-á-vis scatter points. We then constrain the inset within the upper y quadrant of the main graph by inflating the upper y limit using `ylabel(0.0075 0.005, axis(3))`. We also reduce the top plot margin via `plotregion(margin(t=-2.2))` to eliminate the marginal offset between inset and graph. A near-inverse of this logic is used to plot a kernel density of `tempjuly` on `xaxis(4)` and `yaxis(4)`.

Histogram variables and new macros:

```
forvalues r = 1/4 {
   twoway__histogram_gen cooldd if region ==‘r’,                              ///
   density gen(h‘r’ x‘r’) start(0) width(300)
}
replace x1 = x1 - 300
replace x2 = x2 - 350
replace x3 = x3 - 400
replace x4 = x4 - 450
local l2ops legend(order(0 "Region" 2 3 4 5) rows(1))
local dops lcolor(black) lwidth(vthin)
```

Main graph with shaded zone:

```
graph twoway scatteri 70 -50 72 -50 72 4700 70 4700 70 -50, recast(area)       ///
color(gs14%70) lpattern(solid)  lwidth(vthin) plotregion(margin(l=-2.75))      ///
|| scatter tempjuly1 tempjuly2 tempjuly3 tempjuly4 cooldd,‘mops’ ‘l2ops’ ‘labs’ ///
xlab(0 1000 2000 3000 4000) xscale(range(-150 4750) extend)                    ///
```

Inset plot with shaded zone:

```
|| scatteri 70 373 72 373 72 971.2 70 971.2 70 373, recast(area) color(gs14%70) ///
lpattern(solid) lwidth(vthin) yaxis(2) xaxis(2) ylabel(69 90, axis(2))         ///
xlabel(-300 1150, axis(2)) yscale(axis(2)off) xscale(axis(2)off) norescaling   ///
|| scatteri 74.9 373 75.93 373 75.93 971.23 74.9 971.3 74.9 373, recast(area)  ///
yaxis(2) xaxis(2) color(white)  lpattern(solid) lwidth(vthin)                  ///
|| scatter tempjuly1 tempjuly2 tempjuly3 tempjuly4 cooldd                      ///
if inrange(tempjuly, 70, 75) & cooldd <= 950, ‘mops’ ‘l2ops’ yaxis(2) xaxis(2) ///
|| scatteri 60 2120 72 2120 72 4150 60 4150 60 2120, recast(line) ‘bops’       ///
```

Main graph inset box and connecting lines:

```
|| scatteri 70 500 76 500 76 950 70 950 70 500, recast(line) ‘bops’            ///
|| scatteri 70 500 60 2120, recast(line) ‘bops’                               ///
|| scatteri 76 950 72 4150, recast(line) ‘bops’                               ///
```

Inset axis ticks and labels:

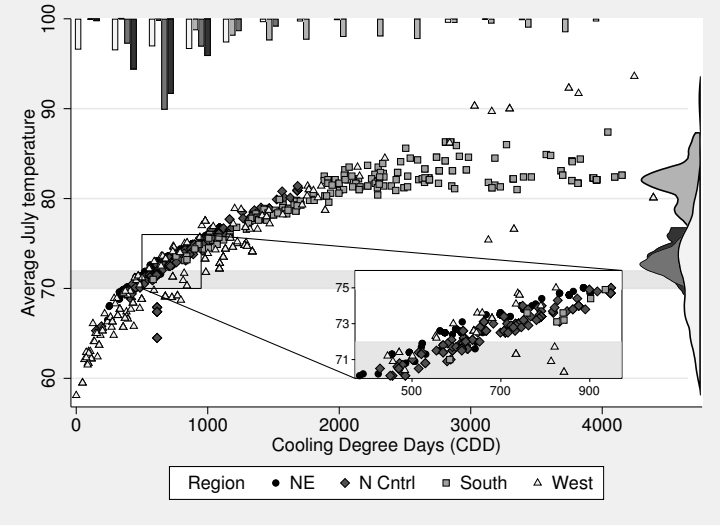
```
|| pci 71 364 71 372, xaxis(2) yaxis(2) ‘bops’                                ///
text(71 344 "71", size(4.5pt) xaxis(2) yaxis(2) just(left))                   ///
|| pci 73 364 73 372, xaxis(2) yaxis(2) ‘bops’                                ///
text(73 344 "73", size(4.5pt) xaxis(2) yaxis(2) just(left))                   ///
|| pci 75 364 75 372, xaxis(2) yaxis(2) ‘bops’                                ///
text(75 344 "75", size(4.5pt) xaxis(2) yaxis(2) just(left))                   ///
|| pci 69.75 500 69.95 500, xaxis(2) yaxis(2) ‘bops’                         ///
text(69.25 500 "500", size(4.5pt) xaxis(2) yaxis(2) just(left))               ///
|| pci 69.75 700 69.95 700, xaxis(2) yaxis(2) ‘bops’                         ///
text(69.25 700 "700", size(4.5pt) xaxis(2) yaxis(2) just(left))               ///
|| pci 69.75 900 69.95 900, xaxis(2) yaxis(2) ‘bops’                         ///
```

```
text(69.25 900 "900", size(4.5pt) xaxis(2) yaxis(2) just(left))                    ///
```

Inset histogram and kernel density:

```
|| bar h1 x1, barw(50) fcolor(gs0) 'dops' yaxis(3) xaxis(3)                        ///
ylabel(0.0075 0.005, axis(3)) xlabel(-300 4750, axis(3))                           ///
yscale(reverse axis(3) off) xscale(alt axis(3) range(-400 4750) off)               ///
|| bar h2 x2, barw(50) fcolor(gs5) 'dops' yaxis(3) xaxis(3)                        ///
|| bar h3 x3, barw(50) fcolor(gs10) 'dops' yaxis(3) xaxis(3)                       ///
|| bar h4 x4, barw(50) fcolor(gs15) 'dops' yaxis(3) xaxis(3)                       ///
plotregion(margin(t=-2.2))                                                         ///
|| kdensity tempjuly if region == 1, recast(area) horizontal fcolor(gs0%90)        ///
'dops' yaxis(4) xaxis(4) ylabel(60 100, axis(4)) xlabel(0 2, axis(4))              ///
xscale(axis(4) reverse off) yscale(axis(4)off)                                     ///
|| kdensity tempjuly if region == 2, recast(area) horizontal fcolor(gs5%90)        ///
'dops' yaxis(4) xaxis(4)                                                           ///
|| kdensity tempjuly if region == 3, recast(area) horizontal fcolor(gs10%90)       ///
lcolor(black) yaxis(4) xaxis(4)                                                    ///
|| kdensity tempjuly if region == 4, recast(area) horizontal fcolor(gs15%90)       ///
lcolor(black) yaxis(4) xaxis(4) plotregion(margin(r=-1.95 l=-2.2))
```

Figure 5: Scatter plot of average July temperature and CDD by region. An inset scatter plot is included to aid visibility of densely clustered North and North Central cities. Histogram and kernel density plots are also included for CDD and average July temperature respectively. Each inset is plotted on an additional (invisible) x and y axis.



Admittedly, this final example conveys probably too much information (Figure 5). Yet the broader - and, for our purposes, more important - take-away is that it is possible to produce this type of intricate, multi-layered inset graph using native Stata commands. Before closing this discussion, we would be remiss if we did not draw attention to the excellent `addplot` package by Ben Jann, which permits users to add plots to existing `twoway` objects (Jann 2015). The advantage of this is that large and somewhat unwieldy code chunks - such that is needed to produce the example above - can be broken up into more manageable, cleaner-looking segments.

4 Final thoughts

We have described a fairly straightforward method for creating highly flexible inset plots in Stata. The main idea - to artifically modify axis ranges so as to constrain the inset within a specified area of a larger graph - is, perhaps, not an ideal solution. But it appears to be the best available. The examples we have presented above barely touch the surface of the full functionality of these plots. Indeed, to do so would be to provide an exhaustive account of every `twoway` *plottype* and their possible combinations. That is to say, these inset plots are simply `twoway` objects which have been manipulated to seamlessly appear within another `twoway` object. If Stata can (`twoway`) plot it, then Stata can inset it.

5 References

Ängquist, L. 2014. gr0057. Stata tip 117: graph combine—Combining graphs. *Stata Journal* 14: 221–225.

Berry, F. 1965. The Shakespeare inset. Word and Picture. 1st ed. London: Routledge & Kegan Paul.

Cox, N. J. 2016. gr0067. Speaking Stata: Shading zones on time series and other plots. *Stata Journal* 14: 805–812.

Harrison, D. A. 2005. gr0014. Stata tip 20: Generating histogram bin variables. *Stata Journal* 5: 280–281.

Jann, B. 2015. gr0065. A note on adding objects to an existing twoway graph. *Stata Journal* 15: 751–755.

Merriam-Webster. Inset. https://www.merriam-webster.com/dictionary/inset.

Naqvi, A. 2021. Advanced data visualizations with Stata. Stata UK Conference 10 September, 2021. https://www.stata.com/meeting/uk21/slides/UK21_Naqvi.pdf.

Wang P. Y., S. Sapra, V. K. George and G. A. Silva. 2021. Generalizable Machine Learning in Neuroscience Using Graph Neural Networks. *Front. Artif. Intell.* 4:618372.

Wiggins, V. 2010. gr0047. Stata tip 93: Handling multiple y axes on twoway graphs. *Stata Journal* 10: 689–690.

Zhuang H. T., Y. Huang and D. E. Acuna. 2021. Graphical integrity issues in open access publications: Detection and patterns of proportional ink violations. *PLoS Comput Biol* 17(12): e1009650.

About the authors

Matthew Tibbles is an independent researcher with a broad interest in experimental methods and their application to novel political and historical contexts. He obtained his BA in Ancient History from the University of Kent in 2018.

Eric Melse is associate professor at the Amsterdam University of Applied Sciences. He obtained his PhD in accounting from Maastricht University in 2008. His research currently focuses on vocational careers in the labor market, professional development of students in higher education and their academic success. Tools and techniques for exploratory data analysis and visualization have his interest.