The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*), *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$115	1-year subscription	$145
2-year subscription	$210	2-year subscription	$270
3-year subscription	$285	3-year subscription	$375
1-year student subscription	$ 85	1-year student subscription	$115
1-year institutional subscription	$345	1-year institutional subscription	$375
2-year institutional subscription	$625	2-year institutional subscription	$685
3-year institutional subscription	$875	3-year institutional subscription	$965
Electronic only		**Electronic only**	
1-year subscription	$ 85	1-year subscription	$ 85
2-year subscription	$155	2-year subscription	$155
3-year subscription	$215	3-year subscription	$215
1-year student subscription	$ 55	1-year student subscription	$ 55

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, Stata, Stata Press, Mata, Mata, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2014)
14, Number 4, pp. 975–990

Speaking Stata: Design plots for graphical summary of a response given factors

Nicholas J. Cox
Department of Geography
Durham University
Durham, UK
n.j.cox@durham.ac.uk

Abstract. Design plots, as defined in this article, show summaries of a response variable given the classes or distinct levels of numeric or string variables presented as influencing factors. Any `summarize` results can be plotted using `statsby` as an engine to produce summaries for groups of observations defined by classes and their cross-combinations. `graph dot` is used by default, but graphs may readily be recast using `graph hbar` or `graph bar`. Such plots offer scope for detailed yet concise data exploration and reporting.

Keywords: gr0061, designplot, design plots, graphics, grmeanby, statsby, summarize

1 Introduction

In this article, I introduce and explain a new command, `designplot`, that produces a graphical summary of a numeric response variable given one or more factors. The term "factor" in this context means that any (numeric or string) variable concerned will be treated in terms of its distinct classes or levels as they occur in the data. Use of Stata's factor-variable syntax is neither explicit nor implicit.

The focus is, therefore, on a Stata program for a particular kind of graph. The choice of a name for such graphs was, in a sense, backward. A program was written to produce graphs that otherwise would be difficult to produce except through several intricate commands. A Stata program always requires a distinct name. That name may be arbitrary within a few syntactic rules, but it is natural to prefer a name that is memorable, even catchy. I borrowed a name that is used in statistical literature for a graph that looks quite different in practice, but in principle shows statistical summaries of the same kind.

Design plots (as here defined) offer a diversity of uses, ranging from simple exploratory overviews to multiscale breakdowns deserving detailed scrutiny. In this article, I discuss what is possible with the new command and relate the ideas behind `designplot` to previous literature.

2 Examples

To begin, we consider the example in figure 1, produced by

```
. sysuse auto
. designplot mpg foreign rep78
```

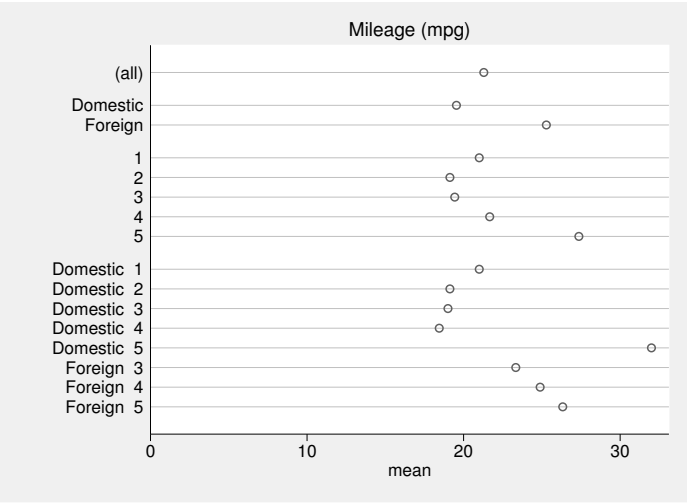

Figure 1. Design plot showing mean miles per gallon (mpg) of all cars and for distinct levels of origin (foreign or domestic) and repair record, singly and jointly

This command produces a plot showing the mean of `mpg` for all observations in Stata's `auto.dta`; for the classes defined by the values of `foreign` and also the classes of `rep78`; and for the classes defined by the cross-combinations of values of `foreign` and `rep78` occurring in the data.

The graph shows several features of the data. There are indications that mean `mpg` differs according to whether cars are foreign (from outside the United States) and according to their repair record. There is a hint that the relation between `mpg` and repair record, an ordered scale, may not be simple. As usual, appearances such as these may be side effects of variations in other predictors not shown.

Because the plot also includes results from cross-combinations of the two categorical variables, we get other clues as to what is occurring. We see that there are no foreign cars with repair record 1 or 2; like almost any other Stata command, `designplot` cannot show results for data that do not exist. Furthermore, whereas foreign cars have higher `mpg` than domestic, given repair record 3 or 4, the reverse is true for cars with repair record 5.

Hence, the plot provides some detail on how a response varies with predictors, presented in terms of their distinct levels. `designplot` takes whatever is offered as pre-

dictors, including string variables as well as numeric. There is no error in presenting a predictor with many distinct levels, but the plot is unlikely to be helpful.

As demonstrated in the first example, `designplot` by default shows means for all the data and for whatever detailed breakdowns (one-way, two-way, and so forth) are possible given the predictors specified. Options give scope for showing other summary statistics as calculated by `summarize` (see [R] **summarize**) and for restricting the results shown in the plot.

By default, the graph is produced by `graph dot` (see [G-2] **graph dot**). Optionally, `graph hbar` (see [G-2] **graph hbar**) or `graph bar` (see [G-2] **graph bar**) may be used instead. Also by default, a `ytitle()` appears at the bottom of the graph when a single summary statistic is shown; if two or more statistics are shown, a legend appears instead. A description of the response variable being shown appears as `t1title()` at the top of the graph, again by default.

Let's look at a different example. The ship *R.M.S. Titanic* sank in the North Atlantic in 1912 with much loss of life. The disaster continues to receive attention in many styles, from books and movies to the statistical approach central here. On the last front, we make no attempt to survey contributions beyond noting the pioneer graphical work of Bron (1912), which seems little known within statistical science.

Dawson (1995) gives an accessible dataset on the fate of those on board (note the small qualifications in his article about the accuracy of the data). Here we read Dawson's version of the data into Stata, specifying whether individuals survived together with various possible predictors. The variable names differ slightly from Dawson's because we follow a convention that $(0, 1)$ indicator or dummy variables should be named for whatever is coded 1. The mean of `survived` is precisely the response of interest as the fraction or proportion surviving. Figure 2 is a first version of our design plot.

```
. infix class 1-9 adult 10-18 male 19-27 survived 28-36 using
> http://www.amstat.org/publications/jse/datasets/titanic.dat.txt, clear
(2201 observations read)
. label define class 0 crew 1 first 2 second 3 third
. label define adult 1 adult 0 child
. label define male 1 male 0 female
. label define survived 1 yes 0 no
. foreach v in class adult male survived {
  2.     label values `v´ `v´
  3. }
. designplot survived class adult male, maxway(2) ysize(7)
> ylabel(0 .25 "25" .5 "50" .75 "75" 1 "100", angle(h)) ytitle(% survived)
> yscale(alt) t1title("")
```

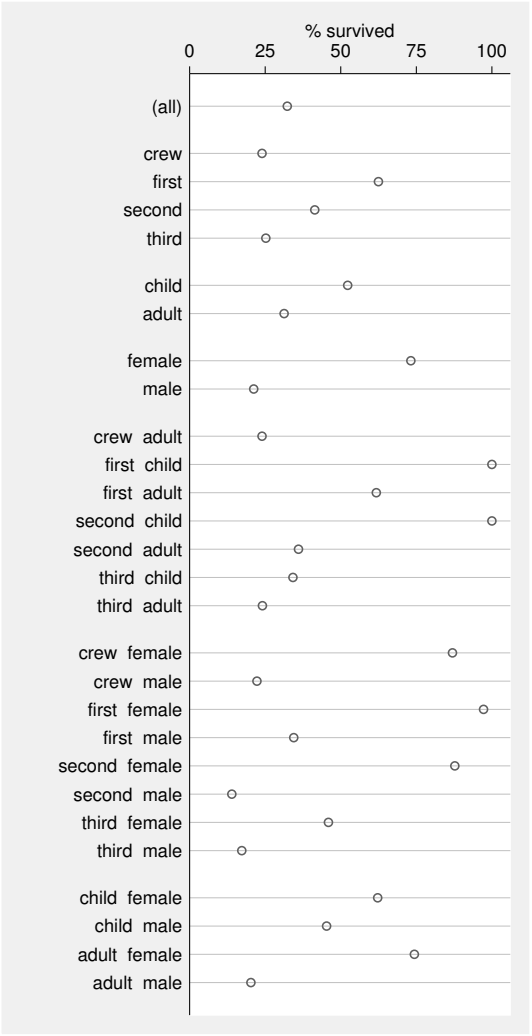

Figure 2. Design plot showing percent survived from the *Titanic* in relation to class, age, and gender

The option choices in this last `designplot` command not only improve the graph as compared with the defaults, but also show how you can additionally exploit the options of the underlying `graph` command (specifically here, `graph dot`):

1. The graph stops short of showing the three-way breakdown (with categories such as "male adults in first class") by using the option `maxway(2)`. The graph still deserves greater height as compared with the default, obtained with `ysize(7)`.

2. Fraction or proportion is a natural scale. If you prefer to show percents on the y axis, you need to change only the axis labels and the axis title.

3. Partly to show that it can be done, the y axis is placed at the top of the graph in a manner common with tables but less conventional with graphs. Its title is made more informative, as `% survived` rather than `mean`. The `t1title()` should, therefore, be suppressed. See Cox (2012a) if desired for more discussion on axis conventions and choices.

The graphics syntax here shows a small clash of conventions. The option `ysize()` for controlling the vertical size of the graph echoes the usual convention that the y axis is vertical. The options `ylabel()` and `ytitle()` echo a convention peculiar to `graph dot` and its siblings `graph bar` and `graph hbar`: the y axis is the axis showing numeric summaries, regardless of its orientation. This convention is adopted to ease experimentation. In particular, the single change from `bar` to `hbar`, or vice versa, is sufficient to move between one command and another without making any changes to options. Less well known is that `graph dot` has an undocumented `vertical` option.

In problems like this, many researchers would prefer a bar chart. `designplot` has a special option to make this easier. The `recast()` option is inspired by the option of the same name for `twoway`. Either option recasts a `graph` command to an equivalent. In `designplot`, you can recast from `graph dot` to `graph hbar` or `graph bar`. `hbar` is far more useful because the categorical axis labeling of `graph dot` rarely works well if transposed to vertical. Note that `recast()` here will not recast your graph to any `twoway` type; as said, the name is inspired by a `twoway` option, but it is not the same option.

If we `recast(hbar)`, we can add whatever small flourishes are permitted by `graph hbar`. Suppose we would like to show the percents as numeric labels at the top of each bar. For this, it is easiest to multiply the binary response by 100 first to change its mean to percent terms. We need a little more space to show such labels for those bars at or near 100%. Figure 3 is the result.

```
. generate survived2 = 100 * survived
. designplot survived2 class adult male, maxway(2) ysize(7)
> ylabel(0(25)100, angle(h)) ytitle(% survived) yscale(alt) recast(hbar)
> blabel(total, format(%2.0f) size(medsmall)) yscale(r(0 110)) t1title("")
```

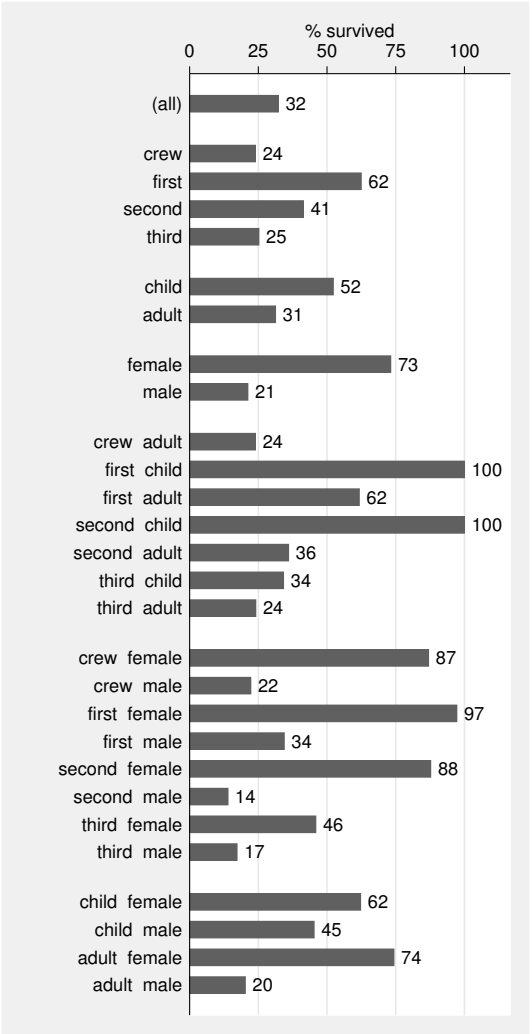

Figure 3. Design plot showing percent survived from the *Titanic* in relation to class, age, and gender; this is figure 2 recast as a horizontal bar chart

In this graph, the size of the numeric labels was determined by experiment. It is arguable that the axis labels and ticks are now redundant given the bar labels. In a moment, we shall see how to remove them.

Figure 3 exemplifies a simple strategy: to blur or even ignore a conventional distinction between graphs and tables (Cox 2008).

Focusing on percent survival is a good idea, but we still should keep track of how many people were in each category. The count or number of observations is one of several summaries available from `summarize`, so a bar chart of frequencies is easy to use

within the same framework. Note that a response variable *yvar* must still be specified, even though it is not evident on the graph. In this graph, we omit axis labels and ticks. We also omit the count for all observations by using the `minway()` option. The small amount of extra space needed for the y axis was again determined by experiment. Figure 4 is the result.

```
. designplot survived  class adult male, statistics(count) minway(1) maxway(2)
> ysize(7) yscale(alt) recast(hbar) blabel(total, format(%2.0f) size(medsmall))
> yscale(r(0 2300)) ylabel(, nolabels noticks) t1title("")
```

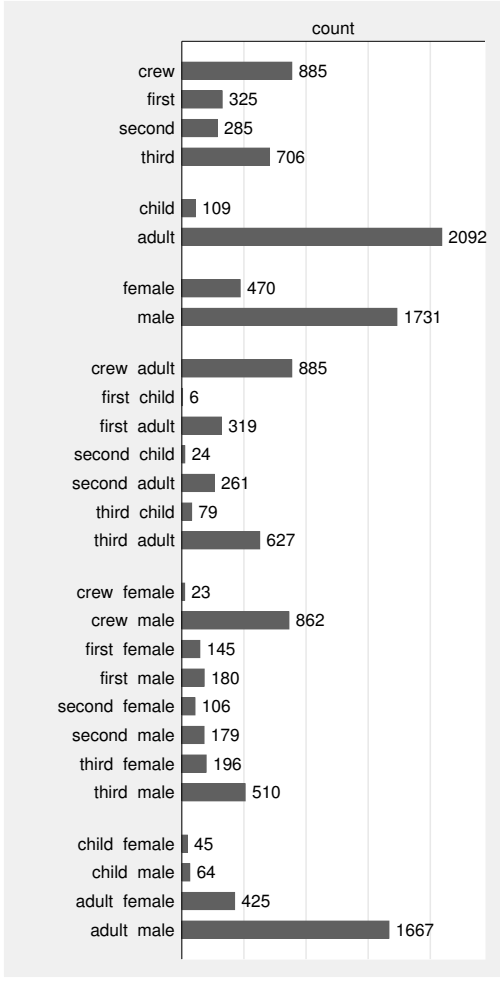

Figure 4. Design plot in the form of a bar chart, showing frequencies of people on the *Titanic* by class, age, gender, and two-way combinations of those categories

3 Origins

`designplot` is based on an eclectic combination of ideas. Readers are warmly invited to inform the author of other similar or related work.

1. The existing Stata command `grmeanby` (see [R] **grmeanby**) shows means (or, optionally, medians) of a response variable given one or more other variables. The scope of `grmeanby` is identical to that of `designplot` insofar as the other variables could be string variables as well as numeric variables. As recorded by Gould (1993) and in the manual entry, `grmeanby` was inspired by examples in Chambers and Hastie (1992). `grmeanby` is based on direct use of `summarize`.

2. Freeny and Landwehr (1992) gave the name "design plot" to plots similar to those in Chambers and Hastie (1992), and that name is associated with software implementations outside Stata, notably in S, S-Plus, and R. The name is also consistent with the S syntax detailed by Chambers and Hastie (1992, 546–547). In these implementations, plots show results from fitting linear models, specifically analyses of variance. The name evokes the idea of an underlying experimental design, but the command here clearly may be applied to any data, including observational data in any sense of that term. The graph shown by Zuur, Ieno, and Smith (2007, 37) is an example from the applied literature. See also Crawley (2013) for more detail on a wider-ranging implementation in R.

3. Various plots given in Hoaglin, Mosteller, and Tukey (1991) show displays "side-by-side" of main effects, interactions, and residuals as fitted in analysis of variance. Roberts (1993, 310) cited an earlier instance of the same idea in Tukey (1977, 451). Cook (1985) gave several examples from three-way analyses. Yandell (1997) called these "effect plots" or "effects plots". Gelman and Hill (2007) gave some loosely similar plots, notably showing standard deviations of coefficients.

4. Broadly similar plots for "graphical analysis of variance" appear in Box, Hunter, and Hunter (2005). See also the earlier work in Box (1993). van Belle (2008, 201) called them "BHH plots". Note that this is not "graphical exploratory analysis of variance" in the sense of Johnson and Tukey (1987).

   Graphs of types 3 and 4 commonly show effects and residuals scaled to be comparable in terms of variability.

5. Graphically, these displays share a possible problem: points may need to be plotted close to each other, creating difficulties especially if any text labels occlude each other or need to be abbreviated. Three out of four examples in Chambers and Hastie (1992) show this, as does the example in [R] **grmeanby**. Several examples in Hoaglin, Mosteller, and Tukey (1991) avoid the problem only by jittering points apart. Harrell (2001) used a different display based on dot charts or dot plots (in the sense of Cleveland [1984, 1994]) that avoids this problem. Conversely, a dot chart representation will work well with, say, 10 entries, but not with 100 or more.

6. On a simpler level, tables or graphs reporting survey results often show two or more separate breakdowns of some sample. Examples are shown by Tufte (2001, 179) and (more trivially) Cox (2008), among many others.

7. The `statsby` command with its `subsets` option provides an easy framework for calculation and assembly of summary statistics for zero-, one-, two-way and higher breakdowns of a dataset. Cox (2010) illustrated its exploitation for graphics. More will be said later on how the term "way" is used with `designplot`.

The term "design plot" is adopted here as a simple, memorable name and given its earlier and widespread use to show similar information. These are positive features. On the other hand, the connotation of experimental design will often be inappropriate. The use of dot-chart (or, optionally, bar-chart) form also distinguishes the results of this command from others published as design plots. People who like the plots and dislike the name are free to use other terminology, or none at all. Not every kind of graph needs a distinct name, but clearly every graph program does.

This lack of standardization is not new:

> "Most or all features of statistical computation—computer hardware, software systems, coding, languages, symbols, terminology, procedures—have much to gain from elimination of pointless variations, redundancies and confusion. Yet pointlessness is not always easy to judge. The only quite satisfying rule of standardization is that you adopt my standards." (Anscombe 1981, 3)

To summarize in Stata terms: `designplot` is a generalization and recasting of `grmeanby`, using `summarize` to produce summaries, `statsby` to provide machinery for multiway breakdowns, and `graph dot` (or `graph bar` or `graph hbar`) to plot the graph rather than `twoway`.

4 Discussion

`designplot` creates a new dataset of `summarize` results that uses default variable names (`_stat1` and so forth) for each statistic and uses `_way`, `_group`, and `_entry` to describe the results. If the number of observations is not one of the statistics requested, a variable with default name `_nobs` is added anyway, on the grounds that it will often be interesting or useful. The original dataset will be restored after the graph is drawn, but the results set may be `save`d for other use with the `saveresults()` option.

We can now epitomize how `designplot` differs from what is readily available through (for example) `graph dot`. There are two main differences. First, `graph dot` and its siblings are more restricted in offering only one-way or two-way or three-way breakdowns given, respectively, one or two or three "factors" as arguments to `over()` or `by()` options. Second, they do not give scope for saving results for separate graphing or tabulation.

Similarly, `designplot` is more general than `grmeanby`, which allows means or medians and one-way breakdowns only.

Consider again the example of figure 1. This example produces a plot that displays the following:

1. the mean of `mpg` for all observations, which may be called a "zero-way" breakdown;
2. the means for all the classes defined by the values of `foreign` and also of `rep78`, which may be called "one-way" breakdowns, as is often done in statistical literature; and
3. the means for all the classes defined by the cross-combinations of values of `foreign` and `rep78` occurring in the data, which similarly may be called a "two-way" breakdown, again as is often done.

In general, specifying one or more factors gives scope for various breakdowns, but the number of (cross-)combinations may grow rapidly and the resulting graph might be too complicated to be readable or useful. Thus `designplot` also offers options to restrict the scope of what is plotted.

Missing values require a special note. `designplot` may be applied when users want to show summaries for missing values of the factors. The recommended approach, however, is to clone the variable concerned and use new codes to show missings explicitly. This is mainly because values of `.` or empty strings would not show up well on graphs. (Missings would be problematic otherwise, given their use by `statsby` to denote all the data.) The help for `designplot` includes a detailed example in which `rep78` for `auto.dta` is cloned and missings are recoded to 6, with value labels to make matters clear.

Some users may wish to add reference lines for (for example) the overall mean (or, optionally, median) in the style of `grmeanby`. This is easy with a prior calculation. The examples in the help include a typical sequence.

The extension likely to be of greatest interest is to move beyond predefined categorical variables that arrive as part of a dataset to intervals defined by the researcher, subdividing the range of counted or measured variables. There is no syntax in `designplot` for this because various methods might be useful. Typically, an extra line of code is required to create a new variable before `designplot` is called.

A method very popular in some quarters is to identify quantile-based bins that contain approximately equal frequencies. `xtile` (see [D] **pctile**) is the usual command of choice here. (Note that researchers are often disappointed by the failure of `xtile` to produce exactly equal frequencies. This is the case whenever the sample size is not a multiple of the number of groups desired, as when 42 can at best be divided into two groups of 10 and two of 11. But the major reason for unequal frequencies is the existence of tied values. Sometimes results better than those of `xtile` can be obtained by using a different inequality at bin boundaries or, equivalently, by binning a negated version of the variable. If this issue is interesting or important to your work, see the comments of Cox [2012b].)

An alternative that deserves greater use by comparison is just to define bins of equal width. On cosmetic grounds, we might have a preference for nice round numbers, where "nice" is a little hard to define but easy to recognize. The functions `floor()` and `ceil()` can crack such a problem (Cox 2003).

The capacity of `designplot` to show frequencies of various unions and intersections of classes or sets makes it an alternative to Venn diagrams. Venn diagrams are popular partly because people recall from early courses (say, in probability) how they make simple problems even simpler. Unfortunately, Venn diagrams in general are very hard to draw usefully. Edwards (2004) gives a definitive account. While he rightly explains how clever tricks make drawing arbitrarily complicated Venn diagrams possible at all, it is difficult to avoid concluding that the results are often too bizarre to be useful statistically.

A yet further possibility is that `designplot` could be applied to cope with multiple response variables. As with researcher-defined binning of counted or measured variables, coping with a different data structure can be delegated to a `reshape long` of the dataset so that several variables are stacked into one. Returning to `auto.dta`, we want to get a plot of skewness and kurtosis for all numeric variables. Figure 5 is the result.

```
. sysuse auto, clear
(1978 Automobile Data)
. rename (price-foreign) (num=)
. reshape long num, i(make) j(variable) string
(note: j = displacement foreign gear_ratio headroom length mpg price rep78
> trunk turn weight)
Data                                wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                        74   ->     814
Number of variables                   12   ->       3
j variable (11 values)                     ->   variable
xij variables:
numdisplacement numforeign ... numweight   ->   num
-----------------------------------------------------------------------------
. designplot num variable, statistics(skewness kurtosis) minway(1)
> t1title(auto dataset) yline(0, lcolor(gs12) lwidth(vthin))
> yline(3, lcolor(gs12) lwidth(vthin) lpattern(dash))
> entryopts(sort(1) descending)
```

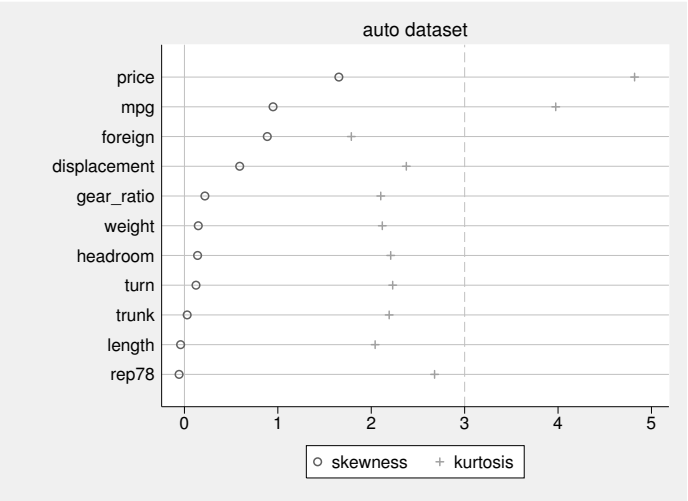

Figure 5. Design plot showing skewness and kurtosis of numeric variables in `auto.dta`

The ease with which the dataset can be restructured in just two lines is offered as grounds for not complicating the syntax, let alone the code, of `designplot`. We add two vertical reference lines. Gaussian (normal) distributions, often used as reference distributions even when we do not expect to observe them in practice, have skewness 0 and kurtosis 3. The skewness and kurtosis of a mix of variables with quite different units of measurement and magnitude would have no meaning; hence, the option calls `minway(1)`. So the interest is just in one group of results. `entryopts()` is a handle to pass options, here to sort the individual entries on the first "variable" plotted or the results for skewness.

5 The designplot command

5.1 Syntax

`designplot` *yvar xvarlist* [*if*] [*in*] [*weight*] [`,` `statistics(`*statistics*`)`
`maxway(`#`)` `minway(`#`)` `saveresults(`*filename*[`,` *save_options*]`)` `prefix(`*prefix*`)`
`recast(bar|hbar)` {`variablelabels`|`variablenames`}` `alllabel(`*text*`)`
`entryopts(`*over_subopts*`)` `groupopts(`*over_subopts*`)` *graph_options*]

`aweights` and `fweights` are allowed; see [U] **11.1.6 weight**.

5.2 Options

`statistics(`*statistics*`)` specifies statistics calculated by `summarize` to be calculated. The default is the mean (only). One or more statistics may be specified. Note that no allowance is made in graphics for different statistics being on different scales, so the user may need to exercise discretion over what is specified. The names allowed include the names of the r-class results as visible after `summarize, detail` or as documented in [R] **summarize**. Thus `p50` specifies the median available as `r(p50)`.

Allowed synonyms also include the following (any synonyms specified will be echoed to the `ytitle()` or legend):

1. `n` or `count` or any abbreviation of `frequency` for `N`.
2. `minimum` for `min` and `maximum` for `max`.
3. `total` for `sum`.
4. `median` for `p50`.
5. `SD` for `sd`.
6. Any abbreviation of `variance` or `Variance` for `Var`.
7. `skew` for `skewness` and `kurt` for `kurtosis`.

Note that if just `statistics(N)` is specified, the *yvar* specified is immaterial so long as it is nonmissing whenever values of *xvarlist* are nonmissing.

`maxway(`#`)` specifies the maximum "way" to be plotted. See the earlier explanation on breakdowns that are called zero-way, one-way, two-way, and so forth. Thus `maxway(1)` by itself specifies that only zero-way and one-way breakdowns be shown.

`minway(`#`)` specifies the minimum "way" to be plotted. See the earlier explanation on breakdowns that are called zero-way, one-way, two-way, and so forth. Thus `minway(1)` by itself specifies that the zero-way breakdown not be shown.

`saveresults(`*filename*[`,` *save_options*]`)` saves the results as a Stata dataset. Options of `save` may be specified, most usefully `replace`. The dataset will include `notes` on the `designplot` command issued and (if defined) the filename and its date for the (`saved`) dataset.

`prefix(`*prefix*`)` is an occasionally used option. `designplot` creates a dataset of results with variable names such as `_stat1` and so forth. If these names clash with existing variable names, this option may be used to add a prefix to all such names to remove the clash.

`recast(hbar|bar)` specifies that the graph be drawn using `graph hbar` or `graph bar`. The default is `graph dot`. People fond of bar charts are advised to try `graph hbar` for greater readability of axis information. Note for experienced users: although the

option name is suggested by another `recast()` option, this is not a back door to recasting to a `twoway` plot.

`variablelabels` specifies that one-way breakdowns be labeled by the corresponding variable labels or by the corresponding variable names if no variable label is defined. The default is, or should be, an invisible label (precisely, an instance of `char(160)`).

`variablenames` specifies that one-way breakdowns be labeled by the corresponding variable names. The default is, or should be, an invisible label (precisely, an instance of `char(160)`). The reason for using this option rather than `variablelabels` is likely to be that variable labels would take up too much space.

Only one of `variablelabels` and `variablenames` may be specified.

`alllabel(`*text*`)` specifies text to label results for all observations used. The default is `alllabel(all)`.

`entryopts(`*over_subopts*`)` specifies *over_subopts* of `graph dot`, `graph hbar`, or `graph bar`, used to tune the corresponding call to an `over()` option that affects the display of individual entries in the graph. Users unsure of what this means may find it helpful to inspect the source code or, alternatively, to just modify a graph using the Graph Editor. Useful examples are `entryopts(sort(1))` and `entryopts(sort(2) descending)`, where `(1)`, `(2)`, etc., indicate the first, second, etc., statistic specified.

`groupopts(`*over_subopts*`)` specifies *over_subopts* of `graph dot`, `graph hbar`, or `graph bar`, used to tune the corresponding call to an `over()` option that affects the display of groups of entries in the graph. Users unsure of what this means may find it helpful to inspect the source code or, alternatively, to just modify a graph using the Graph Editor.

graph_options are other options allowed with `graph dot`, `graph hbar`, or `graph bar`. Note that, among other defaults, `t1title()` is used to display information on *yvar*.

6 Conclusions

The design of design plots was the outcome of an irregular but repetitive personal path. Over the last 20 years or so—for example, in repeated readings of Harrell (2001)—I have often encountered graphs I liked that were loosely or even closely similar to those here. Over that period, `grmeanby` was available as a Stata command offering one solution, but the need was for something more general.

`designplot` is offered with a suggested variety of uses. It builds on versatile commands: `summarize`, `statsby`, and `graph dot` and its siblings. The way they come together is distinctive. `designplot` could be useful in exploration, even if its graphs are never made public, and in reporting, either for one response variable or for several.

7 Acknowledgments

Elizabeth Allred, William Dupont, and Frank Harrell supplied various kinds of encouragement.

8 References

Anscombe, F. 1981. *Computing in Statistical Science through APL*. New York: Springer.

Box, G. E. P. 1993. How to get lucky. *Quality Engineering* 5: 517–524.

Box, G. E. P., J. S. Hunter, and W. G. Hunter. 2005. *Statistics for Experimenters: Design, Innovation, and Discovery*. 2nd ed. Hoboken, NJ: Wiley.

Bron, G. 1912. The loss of the "Titanic". The results analysed and shown in a special "Sphere" diagram drawn from the official figures given in the House of Commons. *Sphere* 4: 103. http://novascotia.ca/archives/virtual/titanic/magazines.asp?ID=56.

Chambers, J. M., and T. J. Hastie, eds. 1992. *Statistical Models in S*. Pacific Grove, CA: Wadsworth and Brooks/Cole.

Cleveland, W. S. 1984. Graphical methods for data presentation: Full scale breaks, dot charts, and multibased logging. *American Statistician* 38: 270–280.

———. 1994. *The Elements of Graphing Data*. Rev. ed. Summit, NJ: Hobart.

Cook, N. R. 1985. Three-way analyses. In *Exploring Data Tables, Trends, and Shapes*, ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 125–188. New York: Wiley.

Cox, N. J. 2003. Stata tip 2: Building with floors and ceilings. *Stata Journal* 3: 446–447.

———. 2008. Speaking Stata: Between tables and graphs. *Stata Journal* 8: 269–289.

———. 2010. Speaking Stata: The statsby strategy. *Stata Journal* 10: 143–151.

———. 2012a. Speaking Stata: Axis practice, or what goes where on a graph. *Stata Journal* 12: 549–561.

———. 2012b. Speaking Stata: Matrices as look-up tables. *Stata Journal* 12: 748–758.

Crawley, M. J. 2013. *The R Book*. 2nd ed. Chichester, UK: Wiley.

Dawson, R. J. M. 1995. The "unusual episode" data revisited. *Journal of Statistics Education* 3(3). http://www.amstat.org/publications/jse/v3n3/datasets.dawson.html.

Edwards, A. W. F. 2004. *Cogwheels of the Mind: The Story of Venn Diagrams*. Baltimore, MD: Johns Hopkins University Press.

Freeny, A. E., and J. M. Landwehr. 1992. Displays for data from large designed experiments. In *Computing Science and Statistics: Proceedings of the 22nd Symposium on the Interface. Statistics of Many Parameters: Curves, Images, Spatial Models*, ed. C. Page and R. LePage, 117–126. New York: Springer.

Gelman, A., and J. Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge: Cambridge University Press.

Gould, W. W. 1993. gr12: Graphs of means and medians by categorical variables. *Stata Technical Bulletin* 12: 13. Reprinted in *Stata Technical Bulletin Reprints*, vol. 2, pp. 44–45. College Station, TX: Stata Press.

Harrell, F. E., Jr. 2001. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. New York: Springer.

Hoaglin, D. C., F. Mosteller, and J. W. Tukey, eds. 1991. *Fundamentals of Exploratory Analysis of Variance*. New York: Wiley.

Johnson, E. G., and J. W. Tukey. 1987. Graphical exploratory analysis of variance illustrated on a splitting of the Johnson and Tsao data. In *Design, Data, and Analysis by Some Friends of Cuthbert Daniel*, ed. C. L. Mallows, 171–244. New York: Wiley.

Roberts, S. 1993. Review of *Fundamentals of Exploratory Analysis of Variance* edited by David C. Hoaglin, Frederick Mosteller, and John W. Tukey. *American Journal of Psychology* 106: 308–320.

Tufte, E. R. 2001. *The Visual Display of Quantitative Information*. 2nd ed. Cheshire, CT: Graphics Press.

Tukey, J. W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison–Wesley.

van Belle, G. 2008. *Statistical Rules of Thumb*. 2nd ed. Hoboken, NJ: Wiley.

Yandell, B. S. 1997. *Practical Data Analysis for Designed Experiments*. London: Chapman & Hall/CRC.

Zuur, A., E. N. Ieno, and G. M. Smith. 2007. *Analyzing Ecological Data*. New York: Springer.

About the author

Nicholas Cox is a statistically minded geographer at Durham University. He contributes talks, postings, FAQs, and programs to the Stata user community. He has also coauthored 15 commands in official Stata. He was an author of several inserts in the Stata Technical Bulletin and is an editor of the Stata Journal. His previous Speaking Stata articles on graphics have been collected as *Speaking Stata Graphics* (College Station, TX: Stata Press, 2014).