The Stata Journal (2018)
18, Number 1, pp. 262–286

Speaking Stata: Logarithmic binning and labeling

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

Abstract. Histograms on logarithmic scale cannot be produced by an option like `xscale(log)`. You need first to transform the variable concerned with a logarithm function. That raises small choices: how to select bin start, bin width, and informative axis labels and titles? Problems and solutions are discussed here in detail.

In contrast, for logarithmic scales on other graphs, options `xscale(log)` and `yscale(log)` may do most of what you want. But there is usually still scope for "nicer" axis labels than are given by default and indeed scope for differing tastes on what "nice" means. This column introduces the `niceloglabels` command for helping (even automating) label choice.

Historical notes and references are sprinkled throughout.

Keywords: gr0072, niceloglabels, logarithms, axis scales, axis labels, binning, histograms, quantile plots, transformations, graphics

1 Introduction

Here is a common problem. You look at a histogram of a highly positively skewed variable and realize that you would be better off with a graph showing magnitudes on a logarithmic scale. Typically, that means equal widths of histogram bins (classes or intervals) on that logarithmic scale.

Here is the same problem in another guise. Often, you know in advance to move straight to logarithmic scale. Economists and others frequently work with income or wealth on such a scale. Many young children know the words *millionaires* and *billionaires* and so have taken the first steps to appreciating that several orders of magnitude (powers of ten) separate the very rich from the very poor. Examples in scholarly books aimed also at general readers include graphs in Atkinson (2015), Milanovic (2016), and Pinker (2018). Another fundamental variable best thought of on logarithmic scale is plant height (Moles et al. 2009). The range in height from short herbs through shrubs and on to the tallest trees is at least 4 orders of magnitude. So your design of a histogram starts knowing that.

The nub of a Stata solution for histograms on logarithmic scale is to use a logarithmic function to generate a transformed variable and then to show a histogram of that new variable. In practice, you should want to go further and improve the small details of axis labels and titles. This column focuses first on the choices needed and how to make them easily. Section 2 explains such binning. It is assumed that you are comfortable with the idea of logarithms.

If you are not so comfortable, you perhaps should read section 5 first, which assumes rather less. Alternatively, you understand the idea well, but your students or colleagues might find more explanation helpful.

Section 3 extends the discussion to labeling using logarithmic scales on other kinds of graphs, such as scatter or line graphs. The focus is on using "nice" labels. Official Stata graph commands default to using labels equally spaced on the original scale, which is often not what you want. A new helper program, `niceloglabels`, is published with this column. Section 4 gives a formal statement of its syntax.

The column does not quite extend to underlining how similar ideas could be used for nonlinear scales other than logarithmic. Such extensions have already been discussed in previous columns (Cox 2008, 2012).

Similarly, the question of when and why histograms should or should not be used is not tackled directly. A subversive minor theme, however, is that quantile plots are generally a good thing.

Various historical details are sprinkled capriciously throughout. Here is the first. The *Oxford English Dictionary*, consulted online, has its first example of *bin* in this sense as mentioned by the statistically minded geologist William Christian Krumbein (1902–1979), writing on particle size data from sedimentology: "In setting up a histogram, we are in effect setting up a series of separate 'bins', each of which contains a certain percent of the grains" (Krumbein 1934, 68).

2 Histogram binning

2.1 Axis scale option is not the solution

At this point, I should underline that the option `xscale(log)`—or if your bins are `horizontal`, the option `yscale(log)`—is not the answer to histogram binning on logarithmic scale. It is often the answer for other needs of logarithmic scales, as I will discuss in section 3. So why not here?

Either axis scale option takes the histogram that you would otherwise have, from `histogram` or `twoway histogram`, and warps the magnitude axis logarithmically. The bin boundaries shown are what would have been shown otherwise. The graph command does not redo the calculation and give you equal-width bins on a logarithmic scale. Worse than that, areas of bars no longer have the interpretation of showing the probability distribution geometrically. The key idea behind a histogram that bar area encodes bin frequency (or some equivalent) is no longer honored—or rather not honored consistently, insofar as the scale varies within the histogram. Using this option to achieve log scale on the magnitude axis is therefore not a solution.

What to do instead is a longer story to which we now turn.

2.2 Sandbox: data on island areas

We use two datasets publicly available as sandboxes for play.

Figure 1 shows the areas of islands above 2,500 km^2 using data from Wikipedia. A data file is posted in the website directory associated with this issue.

```
. set scheme sj
. use island_areas
(https://en.wikipedia.org/wiki/List_of_islands_by_area 21 Sept 2017)
```

The `histogram` syntax is a bare default, apart from pulling the x-axis title downward a little, given the power 2 inside the variable label.

```
. histogram area, xscale(titlegap(*5))
(bin=13, start=2535, width=163712.69)
```

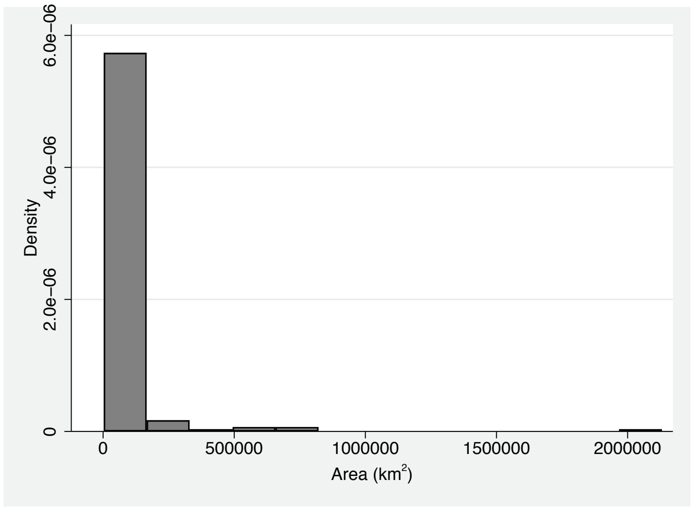

Figure 1. Histogram of areas of those islands above 2,500 km^2. A highly skewed distribution is evident.

The histogram illustrates a glaring problem. It conveys clearly that the distribution is highly skewed. Apart from that, it delivers very little detail. Almost all the islands fall within the leftmost bin. We could tinker with the possible choices, say, by showing more bins, but it is predictable that most of the space on the graph would still be wasted.

With an outcome or response variable that is always positive, measured on a continuous scale, and highly positively skewed, statistical experience should indicate that we try a logarithmic scale.

Before we do that, we should back up and say more about the data. The lower cutoff of 2,500 km^2 arises because the Wikipedia article disclaims completeness of listing below that island size. The data are also questionable at the upper end. You will recall the elementary definition that an island is an area of land surrounded by water. You will also recall that what are conventionally called continents are typically listed separately in reference works. So we are missing Africa, Asia, and Europe (combined); North and South America (combined); Antarctica; and Australia. Those are four (not seven!) very large islands if we respect the criterion of being surrounded by water.

We could digress further and discuss the geographical, geological, and geopolitical senses of the idea of continents. Ambrose Bierce's jest (16 April 1881; see Bierce [2002, 19]) unerringly singled out the main absurdity:

> Australia, *n.* A country lying in the South Sea, whose industrial and commercial development has been unspeakably retarded by an unfortunate dispute among geographers as to whether it is a continent or an island.

For more discussion, see the scholarly and incisive analysis by Lewis and Wigen (1997). At its simplest, the idea of a continent was negative, say, that of land not known to be an island at one time. Now that the world is mapped fairly completely, all conventional continents are known to be islands or parts of islands.

We can summarize for our purposes by noting that the distribution in figure 1 is doubly incomplete, because data are omitted beyond both lower and upper limits. Thus, the problem of showing the distribution is even worse than the graph implies. Firing up `summarize, detail` shows that the ratio of maximum and minimum of the data in hand is about 840. It would be much larger with more data at lower and upper ends.

2.3 Logarithmic functions

Stata (and Mata too) offers two logarithmic functions:

`log()`, and synonymously `ln()`, offers natural (hyperbolic or Napierian) logarithms to base $e \approx 2.71828$. If you do not know e, there is more in section 5 (including some helpful references).

I note in passing an often quoted remark by a famous mathematician (Paul Richard Halmos [1916–2006]) that the notation ln is "a textbook vulgarization" (Halmos 1985, 271). That is a natural (indeed) attitude for mathematicians to take, but applied people should find it useful as a flag to themselves and others that they are not using logarithms to base 10. Oddly, although many notations and abbreviations were tried over several centuries, log (lowercase l, no stop or period) for natural logarithm did not become standard notation until the 20th century (Cajori 1929).

The ln notation is often attributed to Stringham (1893), following Cajori, but is older yet. Velleman (2013) found an earlier example in Steinhauser (1875).[1]

`log10()` offers logarithms to base 10.

If you ever forget whether `log()` means to base e or to base 10, the quickest check is to use `display` with known answers. `display` can be abbreviated `di`.

```
. display log10(10)
1

. display log(10)
2.3025851

. display ln(10)
2.3025851
```

If you have need of logarithms to other bases, my guess is that you know that already and know how to calculate them. What is the base-2 logarithm of 8 (mental check: $2^3 = 8$)?

```
. display log10(8)/log10(2)
3
```

From the first example, and the definition of logarithms discussed in section 5, it can be said that natural logarithms are just logarithms to base 10 multiplied by $\ln 10$. The other way round, logarithms to base 10 are natural logarithms multiplied by $\log_{10}(e) = \log_{10}\{\exp(1)\}$ or divided by $\ln 10$. Graphs with logarithms to different bases look the same; it is just the numbers in axis labels that will differ. So, for many purposes, it does not much matter which base you use, so long as you are consistent and remember which.

For most of my statistical and scientific work, I reach for `ln()`. My reasoning is that this relates more directly to any context with mathematical reasoning using natural logarithms or exponentials. That said, for the problem in view of choosing bins on logarithmic scale, I recommend logarithms to base 10. People in most scientific or practical fields are more likely to relate to powers of 10 than to powers of e (or even 2).

In their excellent books, Cleveland (1985, 1993, 1994) and Robbins (2005, 2013) encourage the use of logarithmic axis labels using powers of 2. But how many people would prefer 1,024 to 1,000 or 1,048,576 to 1 million in the graphs they read? For how many, except possibly computer scientists or experts in combinatorics, does 2^{10} or 2^{20} have more impact and meaning than 10^3 or 10^6? Whatever the answers, a new command to be discussed later does offer some support to those wishing to use powers of 2.

1. David Velleman is a brother of the statistician Paul F. Velleman.

2.4 Digression: A binning scheme using base 3

Base 3 may seem surprising as a basis for binning, but consider this intriguing proposal from the statistical ecologist C. B. Williams (1953; 1964, 9; 1970, 24).[2] His work was also discussed in a previous column (Cox 2005). Williams's main focus is on counted data such as the number of insects caught in a light trap or the number of words in sentences of text, so the data are just counts ≥ 1.

Williams suggests using bins that are 1; 2 3 4; 5 6 7 8 9 10 11 12 13; and so on. Hence, bin midpoints run $1, 3, 9, \ldots$, so the kth bin has midpoint 3^{k-1}. The bin widths (here the number of distinct values) are the same as the midpoints.

Constructing these bins in Mata is engaging. One method is to start with the first bin, for which lower, middle, and upper values are identically 1; then, we add further lower limits by adding 1 to the previous upper limit and further upper limits by adding 3^{k-1} to the previous upper limit.

```
. mata
------------------------------------------------ mata (type end to exit) ------
: bin = (1,1,1)

: for(k = 2; k <= 7; k++) bin = bin \ (bin[k-1, 3] + 1, 3^(k-1),
> bin[k-1, 3] + 3^(k-1))

: bin
          1      2      3
    +----------------------+
  1 |     1      1      1  |
  2 |     2      3      4  |
  3 |     5      9     13  |
  4 |    14     27     40  |
  5 |    41     81    121  |
  6 |   122    243    364  |
  7 |   365    729   1093  |
    +----------------------+

: end
--------------------------------------------------------------------------------
```

Alternatively, the upper and lower limits can be given directly as $(3^{k-1} + 1)/2$ and $(3^k - 1)/2$. Compare entries https://oeis.org/A007051 and https://oeis.org/A003462 in the *On-line Encyclopedia of Integer Sequences*, or sequences M1458 and M3463 in Sloane and Plouffe (1995).

These limits are not quite geometric progressions, so rounding $\log_3$ of the data does not yield those bins. Rather, binning for a variable y is given directly by

$$k = \lceil \log_3(2y + 1) \rceil$$

or in Stata terms,

```
. generate k = ceil(log(2 * y + 1)/log(3))
```

2. Williams (1889–1981) and I went to the same high school and have one university in common. We did not overlap.

2.5 Island areas

Let us go back to work on our island areas. The first step is to get a log base-10 version of `area` into a new variable.

```
. clonevar log10_area = area

. replace log10_area = log10(area)
variable log10_area was long now double
(180 real changes made)
```

What I just did is a twist on the more obvious

```
. generate log10_area = log10(area)
```

Indeed, I am already looking ahead because I know how I will use this new variable. Some small details thus need explanation.

Why is `clonevar` used first? The reasoning is with that command any variable label will be copied over from the original variable. That saves the labor of looking up the variable label and retyping it, or working out how to copy it in some other way. Variable labels can be long and they can be fiddly (mentioning units of measurement, or whatever), so any gain on this front can be welcome.

The variable label no longer matches the contents of the variable. The label says area in square km, but the contents are the base-10 logarithm of that area. So I am playing a little dangerously. That is good reason for making the variable name as informative as possible, in this case `log10_area`. I will need to ensure that what readers (me, anybody else) see on the histogram is consistent: text and numbers should all match.

Let us look at a rudimentary histogram for this log transformed variable (figure 2):

```
. histogram log10_area, xscale(titlegap(*5))
(bin=13, start=3.403978, width=.22496652)
```

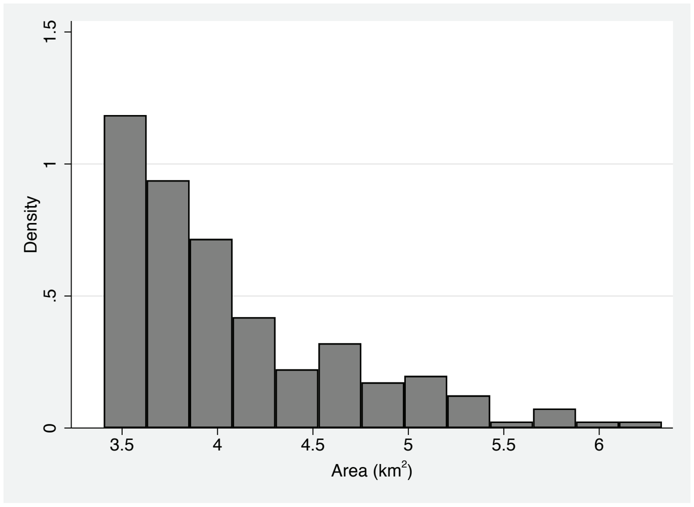

Figure 2. Histogram of areas of those islands above 2,500 km^2 using a logarithmic scale. A reduction in skewness is evident.

That shows progress. The distribution is still skewed, but we are getting a clearer picture in the histogram. A refinement here (and quite often when magnitudes are truncated below) is that we might prefer that binning start at (the logarithm base 10 of) 2,500 km^2. We already know that `display` will give us this number:

```
. display log10(2500)
3.39794
```

So we could retype (or, more smartly, copy and paste) that number into the `histogram` command. But there is an even better way. We can ask Stata to do the calculation on the fly and then use the result, all in one. An example is easier to understand than an explanation:

```
. histogram log10_area, xscale(titlegap(*5)) start(`=log10(2500)´)
(bin=13, start=3.39794, width=.22543098)
```

To save space, we will not show the graph here; it differs only slightly from figure 2.

Do you get the trick here? One piece of syntax that may be new to you here is the punctuation surrounding an expression to be evaluated. The syntax form `` `=*exp*´ `` is an instruction to evaluate the expression *exp* and insert the result in its place. So Stata goes off on the side and works out the expression, here just `log10(2500)`. Then, the `histogram` command sees and uses the result. A tiny bonus, not usually important in practice but welcome in principle, is that you get more precision than `display` shows you by default. It turns out that in this example, 3.397940008672 is the result used. The extra digits make no discernible graphical difference, but using the maximum precision possible can be important for reproducibility in other nongraphical problems.

We have arranged that the bins start where they should, given the lower cutoff. The next task is to work a bit at the axis labels. We know that 10^4 is 10,000, 10^5 is 100,000, and 10^6 is 1 million. Those seem obvious label choices. I would want 2,500 to be a label, too, and we can use the same trick as before to get that shown as a label. We show the resulting graph as figure 3.

```
. histogram log10_area, xscale(titlegap(*5)) start(`=log10(2500)´)
> xlabel(`=log10(2500)´ "2500" 4 "10000" 5 "100000" 6 "1000000")
(bin=13, start=3.39794, width=.22543098)
```

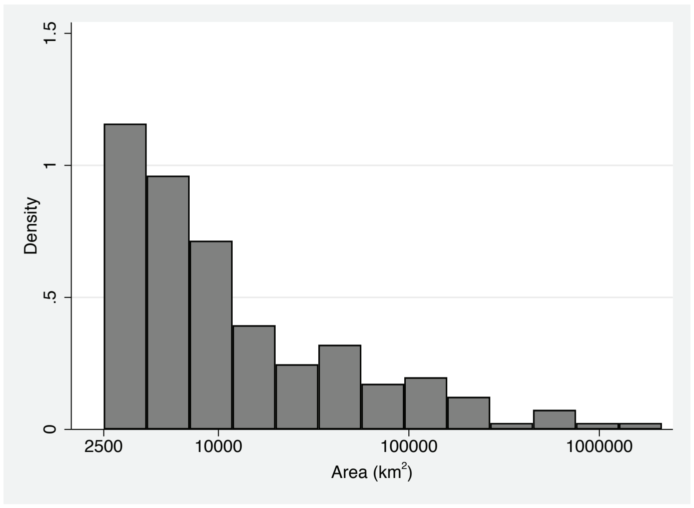

Figure 3. Improved histogram on logarithmic scale with better labeling on the horizontal axis

The effect of the syntax should seem simple, even if any detail in the `xlabel()` call is new to you. We say where the axis labels should go (at 3.39794, or so, and at 4 5 6), and, crucially, we say what text should appear at those positions. Now the axis labels make sense to the reader as areas in square km and match the axis title, inherited automatically from the variable label we arranged earlier.

We are most of the way there, but there is just one more fix that is usually needed. The densities shown are still calculated with respect to $\log_{10}$ area. Getting axis labels to show areas in the original units has not changed that. Almost always, density calculated with respect to a variable not even shown directly is not what you want to see. I would usually reach for one of three options: `frequency`, `fraction`, or `percent`. For exploratory work on one-off datasets, `frequency` is often simplest and best. For comparing with other work, `fraction` or `percent` might seem better. Figure 4 shows where we are now.

```
. histogram log10_area, xscale(titlegap(*5)) start(`=log10(2500)´)
> xlabel(`=log10(2500)´ "2500" 4 "10000" 5 "100000" 6 "1000000") frequency
(bin=13, start=3.39794, width=.22543098)
```

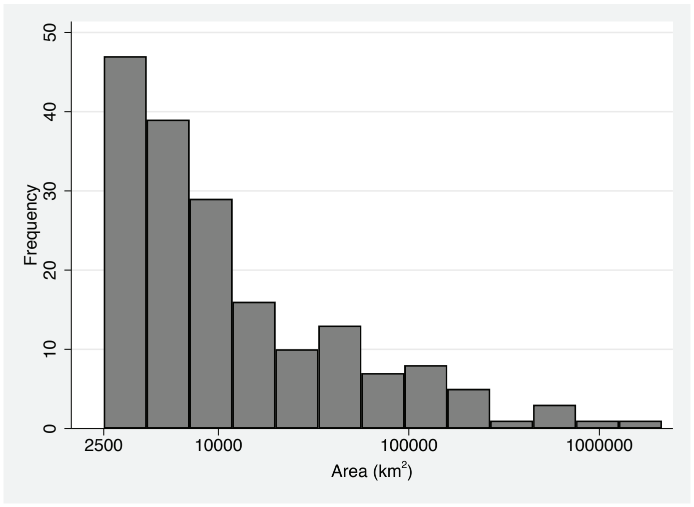

Figure 4. Histogram with better magnitude axis labeling

Small tweaks are always possible to any graph. Here are some possibilities. Naturally, my own tastes weigh heavily in the suggestions.

More labels. Some people might want more axis labels, say, at 25,000 and 250,000 too. You already have the tricks for doing that.

Text for big numbers. At some point, depending partly on taste and partly on whether you run out of space, you might prefer to change the text for the biggest numbers to, say, `"1 million"` or `"1 m"`.

Change of units. Nothing except possibly taste or convention rules out a change of units, say, to 1,000 km^2, so that the text shown ranges from 2.5 to 1,000.

Horizontal labels. The y-axis labels would look better aligned horizontally. Horizontal text is much easier to read, so long as we do not lose too much space thereby.

More or fewer bins. Some people might want more or fewer bins. It may be easier to play with `bin()`, specifying number of bins, than with `width()`, which (remember!) must be expressed on the logarithmic scale.

Align bin bounds and labels. Some find it a little disconcerting whenever axis labels do not line up with bin boundaries. My advice is not to think that this is a problem. We will shortly see an example where it is easy to get.

Colors. Bar colors could be more attractive. Here in the *Stata Journal*, we are using `scheme sj`, but you need not make the same choice.

More explanation. For presentation or publication, we would want to add a good graph title or caption, but I will stop short of addressing what or how.

What the data are telling us is a good question, which we will also not pursue. If you know about Pareto distributions, for example, you will know that taking logs of a Pareto distribution gives you an exponential distribution, but let that be a throwaway remark.

2.6 Sandbox: Data on country populations

Figure 5 shows the populations of countries, again using data from Wikipedia. A data file is posted in the website directory associated with this issue. The data are as they come: see the extensive set of comments on Wikipedia on territories whose status is in dispute in some sense. We have skipped the pretense of discovering that a logarithmic scale is the right way to think here. We should know that in advance. That is precisely why the example is included.

```
. use country_populations, clear
(https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population)
. describe
Contains data from country_populations.dta
  obs:           240                          https://en.wikipedia.org/wiki/Li
                                                st_of_countries_and_dependencies
                                                _by_population
 vars:             4                          5 Oct 2017 13:58
 size:        28,080                          (_dta has notes)
--------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
--------------------------------------------------------------------------------
name            str54   %54s
population      double  %10.0g                Population
date            str18   %18s
source          str37   %37s
--------------------------------------------------------------------------------
Sorted by:
. clonevar log10_pop = population
. replace log10_pop = log10(log10_pop)
(240 real changes made)
```

```
. histogram log10_pop
(bin=15, start=1.7558749, width=.492403)
```

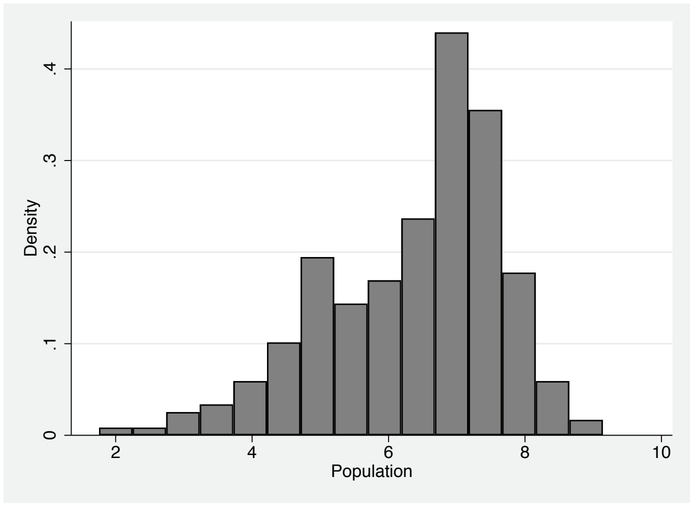

Figure 5. Distribution of country populations as reported on Wikipedia, 22 September 2017. Histogram shows populations on logarithmic scale. The histogram is a good start but needs improvement in several details.

The default—fortuitously but fortunately—illustrates a point small enough not to worry about but large enough to explain. If we tweak the start of binning to 1.5 and the bin width to 0.5, then some of the bin boundaries, namely, the integers 2 to 9, will be "nice" round numbers, as I hope you agree. There will be more on what defines niceness in the next section. That makes informative labeling quite easy. Much scope exists for tinkering with the mix of numbers and abbreviations (m for millions?) or for choosing fewer labels (surely not more?). I will flag (among other cosmetic changes) the tricks of extended tick length and pulling down the axis title. Figure 6 is the improved version.

```
. histogram log10_pop, start(1.5) width(0.5) xlabel(2 "100" 3 "1000" 4 "10000"
> 5 "100000" 6 "1 m" 7 "10 m" 8 "100 m" 9 "1000 m", tlength(*2))
> xscale(titlegap(*5)) frequency bfcolor(gs15) ylabel(, angle(horizontal))
(bin=16, start=1.5, width=.5)
```

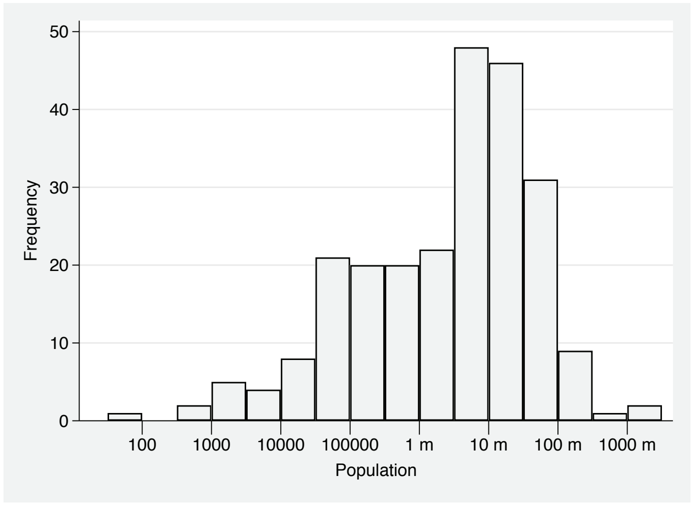

Figure 6. Distribution of country populations as reported on Wikipedia, 22 September 2017. Histogram shows populations on logarithmic scale. Note choices of bin start, bin width, tick length, and title position as indicated by the accompanying syntax.

You can experiment, as I did, with other bin widths, such as 1 or 0.25. As usual, there is a tradeoff between detail and simplicity. I stopped with the graph you see in figure 6.

3 Labeling

3.1 Default labeling can be awkward

The main strategy for logarithmic binning for histograms was to transform to a new variable, yet also to arrange that readers see axis labels and titles in the original units. Those original units are usually easier to think about. Logarithmic scales that are conventionally accepted and routinely used (say, decibels, pH for acidity, Richter scale for earthquake magnitude, stellar magnitude) do not need such transformations. Occasionally, readers do need to be reminded that on logarithm base-10 scales, a step of 1 corresponds to a factor of 10 and so the difference (really, the multiplier) can be a big deal.

Just using `xscale(log)` or `yscale(log)` is much easier when it is the right answer. Whenever that is done, we still need to think about axis labels. The same kinds of questions arise with many kinds of graphs, ranging from scatterplots and line graphs to more exotic or specialized kinds, but we will not need to look at many examples to think about the issue. In fact, we will keep going with the minor theme of univariate distributions.

A quantile plot using the official command `quantile` shows the problem that can arise. In passing, I will mention that `qplot` (Cox 1999, 2005, 2016) is more versatile, but we do not need it here. Figure 7 shows the country populations we have been looking at. Log scale warps the reference line for a uniform distribution that appears by default, so we blank it out as a distraction using no line color.

```
. quantile population, yscale(log) rlopts(lcolor(none))
```

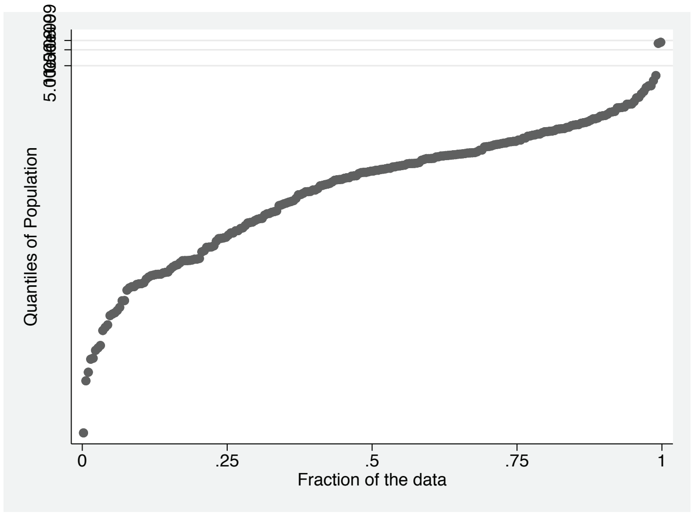

Figure 7. Quantile plot for country populations. The vertical axis labels on logarithmic scale are squeezed up and unreadable.

The problem is unreadable labels at one end of the scale. An obvious solution is to reach in and spell out a better choice of labels manually. Experienced users will often have done that. The contribution of this section is to offer a helper program that suggests some nice choices given the range of the variable (or, alternatively, a specified minimum and maximum).

3.2 Nicer labels can be specified

"Nice numbers" for axis labels have been discussed in several places. Heckbert (1990) commendably tried to make the problem, or its solution, as simple as possible. Ideas similar if not identical often underlie programs, both in Stata and elsewhere (for example, Hardin [1995]). In contrast, and equally commendably, Wilkinson (2005, 95–97) and Talbot, Lin, and Hanrahan (2010) have spelled out how fully automated choice entails a delicate tradeoff between several desiderata, chiefly simplicity, coverage, granularity, and legibility. Talbot, Lin, and Hanrahan (2010) give a full survey of the problem. The aim here is more modest, merely to make suggestions given users' preferences on style.

In this small literature, little attention has been paid to logarithmic scales. It is an easy start that, for example, integer powers of any base, such as

..., 1, 10, 100, 1,000, ...

or

..., 1, 2, 4, 8, ...

would be equally spaced on logarithmic scale. In what follows, these are called styles 1 and 2. The main issue is what to do if you want more labels, especially with the first set of choices. This is where the new command `niceloglabels` provides support.

Various styles of logarithmic labels go beyond integer powers with evident preferences for using nice numbers as labels, even at the cost of sacrificing equal intervals for slightly varying intervals. Thus two styles common in various literatures are

..., 1, 2, 5, 10, 20, 50, 100, ...

..., 1, 3, 10, 30, 100, ...

So multiplication steps are alternately $2\times$ and $2.5\times$ in the first set and alternately $3\times$ and $(10/3)\times$ in the second set. In what follows, these are called styles 125 and 13. The choice between these styles can be based partly on taste and partly on the fact that 125 typically offers more labels than 13 for the same relative range. What they have in common is approximately equal multiplicative steps combined with single significant figures. Recall that significant figures are what is left after setting aside leading and trailing zeros. Thus, 10^9 has one significant figure and 0.12 has two.

A different style runs

..., 1, 4, 7, 10, 40, 70, 100, ...

with the idea of equal additive steps within each power of 10, given that $4-1=7-4=10-7=3$. This is less common in my experience, but for an example, see Dupont (2009, 270).

Finally, I will mention

..., 1, 5, 10, 50, 100, ...

with a similar motivation but showing a preference for 1 and 5 as significant figures. This is called style 15.

If you use `niceloglabels`, you need to do a little work.

1. You need to specify the variable concerned or else a minimum and maximum defining a range.
2. You need to choose your idea of "nice" given a small catalog of available styles. As in everyday life, tastes can differ, so you need to specify yours.
3. You need to name a local macro to hold results.

Consider the variable `pop` shown in figure 7. Here are some examples of using `niceloglabels` to suggest labels.

First, we try the pure powers of 10 implied by style 1 for that variable. The output, echoed to the Results window, shows what many will have realized immediately: there are far too many zeros for a congenial display.

```
. niceloglabels pop, style(1) local(yla)
100 1000 10000 100000 1000000 10000000 100000000 1000000000
```

At that point, we can reach for an option to specify powers using the syntax defined in `help text`. Typing this kind of detail is especially tedious and error-prone, quite apart from the possibility that people need scripts to automate graph production.

```
. niceloglabels pop, style(1) local(yla) powers
100 "10{sup:2}"  1000 "10{sup:3}"  10000 "10{sup:4}"  100000 "10{sup:5}"
> 1000000 "10{sup:6}"  10000000 "10{sup:7}"  100000000 "10{sup:8}"
> 1000000000 "10{sup:9}"
```

In each case, the text displayed is echoed to a local macro. The name of that macro should be specified in a graph option. There is absolutely no need to retype the syntax or even copy and paste it. As this example shows, we can specify other details at the same time.

```
. quantile pop, yscale(log) ylabel(`yla´, angle(horizontal))
> rlopts(lcolor(none))
```

Figure 8 shows the result.

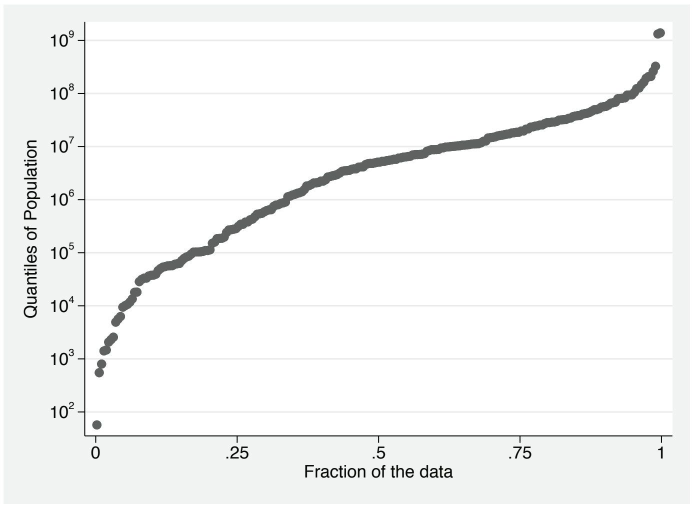

Figure 8. Quantile plot for country populations. The vertical axis labels on a logarithmic scale were produced using `niceloglabels` and are congenially spaced and readable. Compare figure 7.

4 The niceloglabels command

4.1 Syntax diagram

`niceloglabels` *varname* [*if*] [*in*] , `local(`*macname*`)` `style(`*style*`)` [`powers`]

`niceloglabels` *#1 #2*, `local(`*macname*`)` `style(`*style*`)` [`powers`]

4.2 Description

`niceloglabels` suggests axis labels that would look nice on a graph using a logarithmic scale. It can help when you choose `yscale(log)` or `xscale(log)`, or both, and wish to show nicer labels than the default. Results are put in a local macro for later use.

4.3 Remarks

There are two syntaxes. In the first, the name of a numeric variable must be given. The values selected must all be positive. In the second, two numeric values are given, which will be interpreted as indicating minimum and maximum of an axis range. Those two values can be given in any order, but as before, values must both be positive.

"Nice" is a little hard to define but easier to recognize. For example, it is a bonus if labels are exactly or approximately equally spaced on a logarithmic scale (or conversely, on the original scale), and it is a bonus if numbers are powers of 10 or 2 multiplied by small integers. Users must specify their preferred *style*, one from the following list:

1 means powers of 10 such as . . . , 0.1, 1, 10, 100, 1,000, . . .

13 means cycling such as . . . , 0.3, 1, 3, 10, 30, 100, 300, . . .

15 means cycling such as . . . , 0.5, 1, 5, 10, 50, 100, 500, . . .

125 means cycling such as . . . , 0.1, 0.2, 0.5, 1, 2, 5, 10, . . .

147 means cycling such as . . . , 0.1, 0.4, 0.7, 1, 4, 7, 10, . . .

2 means powers of 2 such as . . . , 1, 2, 4, 8, 16, . . .

When the ratio of maximum (max) and minimum (min) is an order of magnitude (power of 10) or less, none of these styles will suggest more than a few labels. In that case, you are almost certainly better off with labels equally spaced on the original scale, which is what Stata gives you anyway.

To make this concrete, here are the numbers of labels suggested when the minimum is $10 = 1e1 = 10^1$ and the power of the maximum is as tabulated in rows. Thus, the first row is for min = 10 and max = $100 = 1e2 = 10^2$, for which the labels suggested, for styles 2 147 125 15 13 1, are, respectively, 16 32 64; 10 40 70 100; 10 20 50 100; 10 50 100; 10 30 100; 10 100; hence, the numbers of labels are 3 4 4 3 3 2.

		style					
		2	147	125	15	13	1
	2	3	4	4	3	3	2
	3	7	7	7	5	5	3
power	4	10	10	10	7	7	4
of	5	13	13	13	9	9	5
max	6	17	16	16	11	11	6
	7	20	19	19	13	13	7
	8	23	22	22	15	15	8
	9	27	25	25	17	17	9

Powers of 2 make most sense in practice when the amounts to be shown are small positive integers or the problem has some combinatorial flavor, or both.

`niceloglabels` is conservative in that it typically will not suggest labels outside the data range. You could add such labels on the fly in your calls to later graphics commands. Technical hint: The small print behind "typically" is a fudging of minimum and maximum as a workaround for precision problems.

For an example of 147, see Dupont (2009, 270).

Note the suggestion by Cleveland (1985, 39; 1994, 39) of 3–10 labels on any axis.

4.4 Options

`local(`*macname*`)` inserts the specification of labels in local macro *macname* within the calling program's space. Hence, that macro will be accessible after `niceloglabels` has finished. This is helpful for later use with `graph` or other graphics commands. `local()` is required.

Anyone new to the idea and use of local macros should study the examples in the help carefully. `niceloglabels` creates a local macro, which is a kind of bag holding the text to be inserted in a `graph` command. The local macro is referred to in that graph command using the punctuation ` ´ around the macro name. Note that the opening (left) single quote and the closing (right) single quote are different. Other single quotation marks will not work. Do not be troubled by the closing single quote (') appearing as upright in many fonts.

`style(`*style*`)` specifies a style for axis labels. `style()` is required. See section 4.3.

`powers` specifies that labels be specified using syntax interpreted by `graph` as superscripts and ready to be used within a `ylabel()` or `xlabel()` option call. Thus, if the labels were 100, 1,000, and 10,000, the output would be

```
100 "10{sup:2}" 1000 "10{sup:3}" 10000 "10{sup:4}"
```

5 Tutorial: Logarithms as powers

Many friendly accounts of logarithms can be found in a great variety of styles. Abbott (1942) is clear and concise and only slightly old fashioned. Gullberg (1997) combines simple explanations with historical and biographical details. Axler (2009 or any later edition) is informatively discursive with many examples of applications. Maor (1994) is entertaining and enthusiastic on the larger history and mathematical context but also a little erratic (see the review in Blank [2001]).

Let's start by focusing on power notation. The 2 and 3 within 10^2 and 10^3 are powers or exponents. In the simplest case with such powers that are positive integers (whole numbers), this notation is a way to express repeated multiplication:

$$10 = 10^1, \quad 100 = 10 \times 10 = 10^2, \quad 1000 = 10 \times 10 \times 10 = 10^3$$

The power or exponent (in this case 1, 2, 3) is the number of times the base (in this case 10) appears in the product. Multiplication corresponds to adding powers, for example,

$$100 \times 1000 = 10^2 \times 10^3 = 10^5 = 100000$$

and division therefore to subtracting powers, for example,

$$1000/100 = 10^3/10^2 = 10^1 = 10$$

Therefore, $10^2/10^2 = 10^0 = 1$, and if we continue dividing by 10,

$$10^0/10^1 = 10^{-1} = 0.1, \quad 10^{-1}/10^1 = 10^{-2} = 0.01, \quad \text{etc.}$$

That extension gives a meaning to powers that are zero or negative integers.

We can extend this to intermediate positive and negative powers. If $100 = 10^2$ and $1000 = 10^3$, then we look for x in $200 = 10^x$, which is between 2 and 3 because 200 is between 100 and 1,000. In this broadening, x need not be an integer or whole number: it could be a real number with a fractional part. If $y = 10^x$, then we call x the logarithm or log of y to the base 10. That is, $x = \log_{10} y$. So

$$\log_{10} 100000 = 5, \quad \log_{10} 100 = 2, \quad \log_{10} 0.01 = -2$$

We use a calculator or computer function to get

$$\log_{10} 200 = 2.30103$$

or other logarithms with fractional parts. Logarithms to the base 10 are occasionally called common or Briggsian logarithms (after the English mathematician Henry Briggs, 1561–1630).

The general definition is that if $y = c^x$, then x is the logarithm or power of y to the base c, which must be positive. Two frequently used bases apart from 10 are 2 and $e \approx 2.71828$. Powers of 2 are met in binary arithmetic, and also in sedimentology, where particle size is often reported on the phi scale (ϕ), which is $-\log_2$ of particle size in mm. The number e has many special properties, particularly those encountered in differential and integral calculus, such as that the derivative with respect to x of e^x is also e^x. Logarithms to the base e are often called natural or Napierian logarithms, after the Scottish mathematician John Napier (1550–1617), and denoted $\log_e$ or ln. If we have logarithms to two different bases a and b, then

$$\log_a y = \log_a b \times \log_b y$$

$\log_a b$ is always a constant: for example, $\log_{10} e \approx 0.43429$.

We have two rules already, that multiplying corresponds to adding logarithms and dividing to subtracting them. In general,

$$\log(y_1 \times y_2) = \log y_1 + \log y_2$$

and

$$\log(y_1/y_2) = \log y_1 - \log y_2$$

so that

$$\log\left(y^2\right) = \log(y \times y) = \log y + \log y = 2 \log y$$
$$\log\left(y^3\right) = \log(y \times y \times y) = 3 \log y$$

and in fact

$$\log\left(y^k\right) = k\log y,\ \text{for any value of } k$$

Sometimes, we want to reverse the process and return to the original scale, which is called antilogging:

$$\text{antilog}(\log y) = y$$

The appropriate function will be c^x or $c^{\log y}$ for base c, for example, 10^x, 2^x, e^x for bases 10, 2, e.

If $x^2 = y$, then x is the square root of y, denoted $\sqrt{y}$,

$$\sqrt{y} \times \sqrt{y} = y^1$$

so the power or logarithm of the square root is given by

$$\log\left(\sqrt{y}\right) + \log\left(\sqrt{y}\right) = 2\log\left(\sqrt{y}\right) = 1$$

and is therefore 1/2, so that $\sqrt{y} = y^{1/2}$. Similarly, if $x^3 = y$, then x is the cube root of y or $\sqrt[3]{y} = y^{1/3}$.

Many processes are physically multiplicative (growth or decline depends on the amount present, such as growth under compound interest or radioactive decay) and are rendered mathematically additive by representation on logarithmic scales. Many variables vary over several orders of magnitude (which means powers of 10) but are compressed toward the lower end of the scale. Taking logs squeezes high values together relative to low values: hence, this reduces positive skewness of single variables and crowding near the origin of scatterplots, which are both common in data analysis. Another reason for taking logarithms is that power functions $y = ax^b$ are converted to straight lines because

$$\log y = \log a + \log x^b = \log a + b\log x$$

which is a straight line on a plot of $\log y$ against $\log x$. Hence, many curved relationships (convex or concave) can be straightened.

For any readers particularly interested in the history of ideas, I now add further notes and references. Kaunzer (1994) gives a scholarly and thorough outline of the history. The term *logarithm*, as a portmanteau creation based on Greek roots *logos* (here best translated as *ratio*) and *arithmos* or *number*, was introduced by John Napier. Napier's life and works have received much attention (Napier 1834; Knott 1915; Gladstone-Millar 2003; Havil 2014). Incidentally, Havil (2014, ix) lists 12 variant spellings of his family name, which helps to explain why variants such as "Naperian" persist to the present.

Some key points from the longer history deserve emphasis. The ideas of powers and exponents as notation and as easing multiplication, division, and rooting go back many centuries. Napier's major contribution was practical provision of tables of logarithms, yet his tables were at most implicitly of logarithms to base $1/e$: calling natural logarithms *Napierian* is thus a little stretch, or even a large one. Honors to Napier for introducing logarithms as calculation aids should be shared to some degree with Henry Briggs; the Swiss Jost Bürgi (1552–1632); and indeed several others who produced fuller tables until well into the twentieth century.

However, these major practical advances did not mean that the ideas of logarithms as merely exponents, the scope for many different bases, and ideas of logarithmic and exponential functions were born in modern form just like that. They grew erratically and unevenly but were well formed and well explained in the work of the great Swiss mathematician Leonhard Euler (1707–1783). His expository works (notably Euler [1748, 1770]) remain highly accessible to modern readers, so look out for versions in your first language (for example, Euler [1822, 1988, 1990]). Dunham (1999) gives a splendid introduction to Euler's mathematical work, while Calinger (2016) is a scholarly but highly readable biography.

6 Conclusion

In statistical computing (and in anything worthwhile?), details matter in getting the best results. Seeing or knowing that a logarithmic scale would be a good idea is, with a little experience, an easy first step in designing a graph. Binning for histograms and getting nice labels on graph axes can be trickier. This column has tried to provide some clear paths through the swamps of syntax and the thickets of taste and technique.

7 References

Abbott, P. 1942. *Teach Yourself Algebra*. London: English Universities Press.

Atkinson, A. B. 2015. *Inequality: What can be done?* Cambridge, MA: Harvard University Press.

Axler, S. 2009. *Precalculus: A Prelude to Calculus*. Hoboken, NJ: Wiley.

Bierce, A. 2002. *The Unabridged Devil's Dictionary*. Athens, GA: University of Georgia Press.

Blank, B. 2001. Reviewed Works: A History of Pi by Petr Beckmann; The Joy of Pi by David Blatner; The Nothing That Is by Robert Kaplan; The Story of a Number by Eli Maor; An Imaginary Tale by Paul Nahin; Zero: The Biography of a Dangerous Idea by Charles Seife. *College Mathematics Journal* 32: 155–160.

Cajori, F. 1929. *A History of Mathematical Notations. Volume II: Notation Mainly in Higher Mathematics*. Chicago: Open Court.

Calinger, R. S. 2016. *Leonhard Euler: Mathematical Genius in the Enlightenment*. Princeton, NJ: Princeton University Press.

Cleveland, W. S. 1985. *The Elements of Graphing Data*. Monterey, CA: Wadsworth.

———. 1993. *Visualizing Data*. Summit, NJ: Hobart.

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

Cox, N. J. 1999. gr42: Quantile plots, generalized. *Stata Technical Bulletin* 51: 16–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 113–116. College Station, TX: Stata Press.

———. 2005. Speaking Stata: The protean quantile plot. *Stata Journal* 5: 442–460.

———. 2008. Stata tip 59: Plotting on any transformed scale. *Stata Journal* 8: 142–145.

———. 2012. Speaking Stata: Transforming the time axis. *Stata Journal* 12: 332–341.

———. 2016. Software Updates: gr42_7: Quantile plots, generalized. *Stata Journal* 16: 813–814.

Dunham, W. 1999. *Euler: The Master of Us All.* Washington, DC: Mathematical Association of America.

Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.

Euler, L. 1748. *Introductio in Analysin Infinitorum.* Lausanne: Apud Marcum-Michaelem Bousquet et Socios.

———. 1770. *Vollständige Anleitung zur Algebra*. St. Petersburg: Kaiserliche Academie der Wissenschaften.

———. 1822. *Elements of Algebra*. London: Longman.

———. 1988. *Introduction to the Analysis of the Infinite: Book I*. New York: Springer.

———. 1990. *Introduction to the Analysis of the Infinite: Book II*. New York: Springer.

Gladstone-Millar, L. 2003. *John Napier: Logarithm John*. Edinburgh: National Museums of Scotland Publishing.

Gullberg, J. 1997. *Mathematics: From the Birth of Numbers*. New York: W. W. Norton.

Halmos, P. R. 1985. *I Want to be a Mathematician: An Automathography*. New York: Springer.

Hardin, J. W. 1995. dm28: Calculate nice numbers for labeling or drawing grid lines. *Stata Technical Bulletin* 25: 2–3. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 19–20. College Station, TX: Stata Press.

Havil, J. 2014. *John Napier: Life, Logarithms, and Legacy*. Princeton, NJ: Princeton University Press.

Heckbert, P. S. 1990. Nice numbers for graph labels. In *Graphics Gems*, ed. A. S. Glassner, 61–63. Cambridge, MA: Academic Press.

Kaunzer, W. 1994. Logarithms. In *Companion Encyclopedia of the History and Philosophy of the Mathematical Sciences*, ed. I. Grattan-Guinness, 210–228. London: Routledge.

Knott, C. G., ed. 1915. *Napier Tercentenary Memorial Volume*. London: Longmans, Green for Royal Society of Edinburgh.

Krumbein, W. C. 1934. Size frequency distributions of sediments. *Journal of Sedimentary Petrology* 4: 65–77.

Lewis, M. W., and K. E. Wigen. 1997. *The Myth of Continents: A Critique of Metageography*. Berkeley and Los Angeles, CA: University of California Press.

Maor, E. 1994. *e: The Story of a Number*. Princeton, NJ: Princeton University Press.

Milanovic, B. 2016. *Global Inequality: A New Approach for the Age of Globalization*. Cambridge, MA: Harvard University Press.

Moles, A. T., D. I. Warton, L. Warman, N. G. Swenson, S. W. Laffan, A. E. Zanne, A. Pitman, F. A. Hemmings, and M. R. Leishman. 2009. Global patterns in plant height. *Journal of Ecology* 97: 923–932.

Napier, M. 1834. *Memoirs of John Napier of Merchiston, His Lineage, Life and Times, With a History of the Invention of Logarithms*. Edinburgh: William Blackwood.

Pinker, S. 2018. *Enlightenment Now: The Case for Reason, Science, Humanism, and Progress*. New York: Viking.

Robbins, N. B. 2005. *Creating More Effective Graphs*. Hoboken, NJ: Wiley.

———. 2013. *Creating More Effective Graphs*. Wayne, NJ: Chart House.

Sloane, N. J. A., and S. Plouffe. 1995. *The Encyclopedia of Integer Sequences*. San Diego: Academic Press.

Steinhauser, A. 1875. *Lehrbuch der Mathematik für höhere Gewerbeschulen. Algebra*. Wien: Carl Gerold's Sohn.

Stringham, I. 1893. *Uniplanar Algebra. Being Part I of a Propædeutic to the Higher Mathematical Analysis*. San Francisco: Berkeley Press.

Talbot, J., S. Lin, and P. Hanrahan. 2010. An extension of Wilkinson's algorithm for positioning tick labels on axes. *IEEE Transactions on Visualization and Computer Graphics* 16: 1036–1043.

Velleman, D. 2013. How did the notation "ln" for "log base e" become so pervasive? https://math.stackexchange.com/questions/1694/how-did-the-notation-ln-for-log-base-e-become-so-pervasive/428560#428560.

Wilkinson, L. 2005. *The Grammar of Graphics*. 2nd ed. New York: Springer.

Williams, C. B. 1953. The relative abundance of different species in a wild animal population. *Journal of Animal Ecology* 22: 14–31.

———. 1964. *Patterns in the Balance of Nature and Related Problems in Quantitative Ecology*. London: Academic Press.

———. 1970. *Style and Vocabulary: Numerical Studies*. London: Charles Griffin.

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 16 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 "Speaking Stata" articles on graphics from 2004 to 2013 have been collected as *Speaking Stata Graphics* (2014, College Station, TX: Stata Press).