The Stata Journal (2022)
22, Number 2, pp. 446–459

DOI: 10.1177/1536867X221106436

Speaking Stata: The largest five—A tale of tail values

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

Abstract. How do you work with the largest five, or smallest five, or any other fixed number of values in a tail of a distribution? In this column, I give examples of problems and code for basic calculations as a prelude to graphics, tables, and more detailed analysis. The main illustration is analysis of concentration among firms or companies, with wider discussion mentioning hydrology, climatology, cryptography, and ecology. The examples allow a tutorial covering sorting and ranking and using `if` and `in` to select observations, `by:` as a framework for groupwise calculations, indicator variables as a mode of selection, and `egen` as a Swiss Army knife with many handy functions.

Keywords: dm0108, by:, egen, if, in, ranking, sorting, tails, concentration, diversity, inequality

1 Introduction

Faced with a distribution, researchers may want to look systematically at (say) the 5 largest or the 5 smallest values. Here 5 is evidently just an example: the same principles apply if the number chosen is 3, 10, 30, or whatever else. Further, 1 is a notable special case.

In economics or finance, we may want to look at the 5 firms with the largest number of employees or sales volume or market value or the 5 best- or 5 worst-performing stocks. In hydrology and climatology, we might want to look at the 5 largest floods (river discharges) or the 5 driest years in terms of rainfalls. Extremes can be interesting and important, to say the least. Think of the Amazon, the world's largest river on most criteria, or Amazon, a rather large firm.

Stata practice here starts with some very simple devices but necessarily gets more complicated as we face up to common difficulties, including wanting to do this systematically for subsets of the data; coping with missing values; desiring to go beyond simple summaries in further work; and wanting to work efficiently with large datasets.

The idea of "the largest 5" or any variation on that theme is easy to explain to lay people or beginning students. The idea may still be helpful at other levels. If 5 seems arbitrary, there are easy answers: try some other choice instead or as well, or do something else entirely if your problem needs a different solution.

In what follows, it is assumed that ties are not a problem. If the 5 largest values are all 42, that is what it is. For ideas on how to handle distinct values, often called unique values, you could start with Cox and Longton (2008). In practice, when people want to do this, the 5 largest or smallest values commonly are distinct and come from a straggly tail of a skewed distribution.

We will stop a long way short of the statistics of extremes. On that, Gumbel (1958) is the classic account. More recent books include Embrechts, Klüppelberg, and Mikosch (1997), Coles (2001), Beirlant et al. (2004), Reiss and Thomas (2007), and Resnick (2007). Davison and Huser (2015) gave a recent review.

2 Easy commands

Starting with easy commands, let us see how far we can get with `summarize`, `collapse`, `sort`, `list`, and `keep` (see [R] **summarize**, [D] **collapse**, [D] **sort**, [D] **list**, and [D] **drop**) in particular.

`summarize` always reports the minimum and maximum. With its `detail` option, it lists the 4 highest and 4 lowest values, without saving any of those results except the minimum and maximum. `summarize` can be more useful once observations have been identified as satisfying a particular criterion, as we will see shortly.

`collapse` lets you collapse to a reduced dataset that could include minimum and maximum values, and whatever else you want, but `collapse` lacks handles for other values in each tail of a distribution. It can still be helpful after other calculations.

`sort` on a variable sorts the dataset from lowest to highest values on that variable. For example, with `auto.dta`, we could go

```
. sysuse auto
. summarize price
. sort price
```

`auto.dta` is a small and generally well-behaved dataset. There are no missing values on `price`, so everything is as simple as it could be. We can now look at the tails of the distribution of `price` using `list`. Here are some basic possibilities, starting with a device you should know already and ending with one you may not have seen before.

```
. list price in 1/5
. list price in -5/L
. list price if inrange(_n, 1, 5) | inrange(_n, _N-4, _N)
```

The `in` qualifier specifies observation numbers and allows one range of observation numbers. The observation numbers depend on the current sort order, which for this problem is precisely the point. So after `sort price`, the qualifier `in 1/5` identifies the observations with the 5 lowest values of `price`, and the qualifier `in -5/L` identifies those with the 5 highest values.

Note especially the use of negative integers to specify observations counting from the bottom of the dataset upward. Stata allows `-1` and indeed the letter `l` to indicate the last observation, but I recommend `L` (also meaning last). `L` means one fewer keystroke than `-1`. Lower case `l` is all too easily misread as the numeral `1`. Using negative numbers surely beats calculations on the fly, such as this: I have 74 observations, so I want to see the last 5, which means `in 70/74`.

Note that the answer is not `in 69/74`. A little thought or experiment underlines that such code identifies six observations. My experience is that one never stops making such off-by-one or fencepost errors and thus needing to fix them. If the term is new to you, a quick Google on "fencepost errors" leads to informative sources.

The last code example just given using `inrange()` is more complicated than you may need, but the idea is sometimes useful. Here `_n` is the observation number (running from 1 to 74 in `auto.dta`), and `_N` is the total number of observations (so 74 in `auto.dta`). We need `_N - 4` at the bottom of the dataset to specify where observations in the top 5 start. In short, `if` used like this is a way to work around restrictions on the `in` syntax. We can specify multiple observation number ranges using the `|` operator, meaning logical or: *this* `|` *that* is true if either *this* or *that* is true. See `help operators` for more detail on logical and other operators.

Sorting has many benefits. We could now use `list` to look at other variables too on the observations with extreme values, including name, place, or time identifiers and other categorical or measured variables. We could now use `summarize` with an `in` qualifier. A detail often overlooked is that `summarize` calculates the sum as well as other results: although not displayed in results, the sum is available immediately afterward as the saved result `r(sum)`.

You could also use `keep` to produce a reduced dataset consisting only of observations in either or both tails. This idea too is more useful in more challenging contexts.

3 Comparing subsets

People wanting this usually seek much more. The context is often wanting to look systematically at how "the largest 5" vary in some way, say, how some results vary from year to year. In economics or finance, we might have companies in different economic sectors for a period of years and want to consider variations in some measures between sectors and between years. In environmental science, we might have monitoring stations in different regions over a period of years. And so on.

One more step in technique yields many useful results. The step is to use `generate` and `egen` (see [D] **generate** and [D] **egen**) as workhorses within a framework provided by the `by` (see [D] **by**) prefix. The perspective is now that new variables are needed, so that we can graph, table, and further analyze our results most easily.

Let's dive into a more challenging example and then explain the new tricks being used. Suppose we want to look at concentration of economic activity as it varies over

time. The Grunfeld data are a good sandbox. Let's use market value `mvalue` as a size variable and look at the total `mvalue` for the largest 5 companies, and that as a fraction of the total `mvalue` for all companies, as they vary from year to year. The latter is often called a "concentration ratio". The Grunfeld data include 10 companies, which is not a problem: nowhere will we wire in 10 as part of the code.

```
. webuse grunfeld, clear
. bysort year (mvalue): generate byte highest = (_N - _n) < 5
. by year: egen double total5 = total(highest * mvalue)
. by year: egen double total = total(mvalue)
. by year: egen count5 = total(highest)
. by year: egen count = count(mvalue)
. by year: generate conc5 = total5 / total
. format mvalue total5 total %3.2f
. format conc5 %4.3f
. tabdisp year, cellvar(count count5 total5 total conc5)
```

The results for our dataset will be shown toward the end of this section.

First off, let me flag that this code capitalizes on there being no missing values in the variables concerned. We will hold until a later section a discussion of how to deal with any missing values.

Next, note that 5 is wired into the code at just one place, although if you used a different integer, it would also be prudent to change some new variable names accordingly. This is not an attempt at very general code. It includes specific details arising from a particular dataset, but we are slowly working our way toward respectable code for a do-file.

To understand the code as a whole, let us work backward from what we want, which is the total market value of the largest 5 companies; the total market value; the ratio of those two totals, which we know will lie between 0 and 1. Preferring that concentration ratio to be a percentage between 0 and 100 is an alternative.

The easiest of these to get at is the total market value for each year, which goes in the new variable `total`.

For that and the partial total for the largest 5, the code uses `double` variables. Totals can become very big, so we decide to worry a little about holding total values to maximum precision.

After we `sort` first by `year` and then by `mvalue`, the largest 5 values are in the last 5 observations. The detail used here is to create an indicator or dummy variable that is 1 if the observation is one of the largest 5 and 0 otherwise. It is hard to overstate how useful such variables can be (Cox 2016; Cox and Schechter 2019).

The precise criterion compares the observation number `_n` with the total number of observations `_N`. It is crucial that—under the aegis of the `by:` prefix—these are calculated within each group of observations specified, in this case a distinct `year` (Cox 2002). Spelling it out, we want the observations with numbers equal to `_N`, `_N` − 1, `_N` − 2, `_N` − 3, and `_N` − 4.

Consider this statement:

```
. by year: egen double total5 = total(highest * mvalue)
```

Multiplying by `highest` means that we are adding terms that are $1 \times$ `mvalue` when in the largest 5 together with terms that are $0 \times$ `mvalue` when not in the largest 5. That boils down easily to the total of `mvalue` for the observations that make up the largest 5 in any subset. Getting a clean expression for selection of observations and multiplication is a nice possibility when an indicator variable is to hand.

The code could have been

```
. by year: egen double total5 = total(mvalue) if highest == 1
```

or even

```
. by year: egen double total5 = total(mvalue) if highest
```

In either case, the result for observations with `highest == 0` would be missing. The indicator variable approach appeals, I suggest, as yielding less clumsy code. It reduces the need to keep track of which observations have nonmissing values of `total5` and which have missing values.

The variable `count5`, which is in `egen` terms `total(highest)`, will similarly be the sum of 1 for each value in the largest 5 and 0 otherwise, which is precisely a count of values in the largest 5, as desired. I have two reasons for calculating this variable.

First, to check on the counting. The result in our dataset should always be 5. But always be on the lookout for off-by-one errors, in which our code could yield 4 or 6. Indeed, worse errors might occur. While working toward this column, I made a careless slip of this kind but caught it quickly because I was counting too.

Second, to look ahead toward the possibility of unbalanced panels with unequal numbers of observations for each year (or time generally). Suppose that there were only 1, 2, 3, or 4 nonmissing values of `mvalue` in each year. Then `total5` would still contain their total; it would also equal `total`; and the concentration ratio `total5 / total` would be identically 1. Keeping a count in the variable `count5` of the number of values in the "largest 5" lets us watch out for such instances.

A further possibility is that there are only 5 nonmissing values in each year, which would also imply a ratio of 1. For that and other reasons, we also have a variable `count`, which is the number of nonmissing values available in each subset. See the help for `egen` on its function `count()` if you want more detail.

Note on Stata terminology. `egen` is a framework for running its own routines, called "functions". Several are bundled with official Stata, and many more have been written by users. Such functions are not the same as functions in Stata proper or in Mata (Cox 2011b). Sorry if this is confusing or irritating, but three different names for varieties of the same idea might well have been just as confusing or irritating.

The results for our new variable are identical for each observation in the same subset, so in this example for each year. That smacks of redundancy but is otherwise convenient. Readers familiar with frames (introduced in Stata 16) will be aware of another approach, and indeed the problem could also be recast using Mata. Identical results for each observation in each year, however, do mean that we want to see each distinct line of results only once. It is one of many small virtues of `tabdisp` (see [P] **tabdisp**) used for tabular display that this behavior can be automatic.

Repeating the `tabdisp` command from earlier in this section, we can now think more about how to show the results.

```
. tabdisp year, cellvar(count count5 total5 total conc5)
```

Year	count	count5	total5	total	conc5
1935	10	5	6319.60	7074.71	0.893
1936	10	5	9838.40	10795.74	0.911
1937	10	5	12479.60	13522.90	0.923
1938	10	5	7632.10	8471.42	0.901
1939	10	5	9726.30	10802.54	0.900
1940	10	5	10335.30	11340.17	0.911
1941	10	5	9946.50	10879.48	0.914
1942	10	5	7972.80	8808.46	0.905
1943	10	5	8993.80	9951.64	0.904
1944	10	5	9205.50	10277.72	0.896
1945	10	5	10282.40	11401.05	0.902
1946	10	5	10831.20	12039.64	0.900
1947	10	5	8140.30	9272.97	0.878
1948	10	5	7841.80	8969.80	0.874
1949	10	5	7973.10	9175.54	0.869
1950	10	5	8410.80	9770.55	0.861
1951	10	5	10474.70	12080.84	0.867
1952	10	5	10755.10	12543.82	0.857
1953	10	5	12839.60	14777.81	0.869
1954	10	5	12585.20	14379.42	0.875

Before the `tabdisp` command, we assigned display formats to the variables here with noninteger values. Specifying two decimal places for the totals just echoed the resolution of the original data. For any purpose other than checking results, more rounding would be fine. Other way round, three decimal places for the ratio is, I guess, about what people often want to show.

I wanted to show the counts explicitly because some people with messier datasets really need to display them. For this kind of dataset, we should check that counts are as they should be but then suppress the display in a presentation or publication. It would also be a good idea to have checks in the code using `assert` (see [D] **assert**) such as

```
. assert count == 10
. assert count5 == 5
```

whenever constancy is expected: it is expected here because we think we have balanced panels and no missing values. If either of those commands failed, you would need to find out why. If the explanation made sense, you would then remove the `assert` commands

from the code but ensure that the counts are available to readers of your research. Either way, it pays to be `assert`ive (Gould 2003).

`tabdisp` remains a neglected command (Cox 2003, 2012). It can be just what you need for quick and not so dirty display of variables, including results variables. It is billed as a programmer's command, but it is easy to use interactively and in do-files.

A serious project would not stop here. You should first fire up a line graph and then think what next to do. Because each result is repeated for observations in the same year, it would be good practice to go

```
. egen tag = tag(year)
. line conc5 year if tag, sort
```

4 Which measures?

This section is a digression on the statistical content and context of the example.

People interested in measuring concentration should be concerned with the strengths and limitations of fraction of total composed by the 5 largest values as one of several possible measures. It is enough for the present purpose that some researchers find it simple and useful, but is it too simple, and is it as useful as other measures? A great deal has been written in this territory, and there is much scope for yet more.

Let us sketch one larger context and give a few simple examples of other possibilities. We have in market value a variable quantifying size or frequency or abundance that must be zero or positive. The distribution of such a variable, say y, can be reexpressed as proportions $y/\sum y =: p$. Such a framework fits many applications, such as letter frequencies in elementary cryptography (Holden 2017; Dooley 2018; Dunin and Schmeh 2020) or abundance of species or taxa generally in ecology (several references in Cox [2005]). In this framework, our measure is just $\sum_{j=1}^{5} p_j$, where ranks j follow a convention that 1 means largest and there is no correction for ties.

Again, the choice 5 is arbitrary here and a matter of judgment. We can see this measure as one of a family, given choice of J in $\sum_{j=1}^{J} p_j$. In ecology, $J = 1$—so simply p_1—has often been used, typically named the Berger–Parker index (Berger and Parker 1970).[1]

Two of several other approaches may be singled out. First comes $\sum p^2 =: R$ or its complement $1 - R$, which is usually named for someone who did not first discover or invent it. Repeat rate, named by Alan M. Turing (Good 1953, 1982), and match probability (MacKay 2003, 267) are good evocative names for R.

1. Naming ideas after people is a generous way to honor the predecessors on whose shoulders we stand and at whose feet we sit. Unfortunately, the implied history is often wrong: the names used may name neither the first inventors nor those who deserve most credit. Beyond that, such names can impede communications, particularly between disciplines.

Then there is $\sum p \ln(1/p) =: H$, usually named entropy, "a quantity which occurs in many contexts, is invoked in many more, and is given justifications ranging from the axiomatic to the metaphysical" (Whittle 1992, 272).

R and H are not directly comparable, but $1/R$ and $\exp(H)$ have an attractive interpretation as an equivalent number of equally common categories. To see that, note that with k identical proportions, so that each proportion is $1/k$, R reduces to $1/k$ and H reduces to $\ln k$.

There are large and sometimes repetitive literatures in several disciplines on how best to measure and analyze concentration (or diversity, or inequality, to mention only two related terms) for a ranked distribution p_j. There are many examples of how quite simple ideas can be helpful. There are also futile debates about which measure is best (in the absence of criteria for "best") and naive exercises trusting that all the information in a distribution can be captured by a single scalar. Comments within Cox (2005), although in several senses utterly standard, seem to retain their force. New ideas and frameworks continue to emerge (for example, Leinster [2021]).

None of the above is to deny that direct analysis of a size variable y, through, say, moment- or quantile-based summaries, may not be a better idea. As Jeffreys (1961, x)[2] remarked, "It is sometimes considered a paradox that the answer depends not only on the observations but on the question; it should be a platitude."

5 Missing values

We postponed discussion of missing values.

Missing values have to be sorted to somewhere in the data. The only systematic choices could be that numeric missing values are regarded as 1) lower than the largest negative value possible or 2) higher than the largest positive value possible. Stata developers made the second choice. It follows that, after sorting, any numeric missing values will compose some of if not all the "largest five"—in the sense that they will be `list`ed last, or shown last in the Data Editor.

A solution to this problem is just to segregate missing values so that they do not enter the calculation. That is easier than might be feared once you have created an indicator variable for being missing, which then allows a different sort order.

2. Completists may be happy to know that the remark occurs on page vii of the 1948 edition and on page vi of the 1939 edition.

We do not need this refinement for the Grunfeld data, but it does no harm. Here—once we have been more careful about creating an indicator variable `highest`—most of the code is unchanged.

```
. webuse grunfeld, clear
. generate byte OK = !missing(mvalue)
. bysort OK year (mvalue): generate byte highest =  OK & ((_N - _n) < 5)
. bysort year: egen double total5 = total(highest * mvalue)
. by year: egen double total = total(mvalue)
. by year: egen count5 = total(highest)
. by year: egen count = count(mvalue)
. by year: generate conc5 = total5 / total
. format mvalue total5 total %3.2f
. format conc5 %4.3f
. tabdisp year, cellvar(count count5 total5 total conc5)
```

Note the use of the logical operator and (`&`). To be regarded as one of the highest, a value must not be missing and in the top 5. *this* `&` *that* is true only if both *this* and *that* are true.

6 Ranking instead

Another idea—which may have already occurred to you—is that for our main example of measuring concentration using the largest 5, what we seek are just values with ranks 1 to 5, so long as we can rank in a way that matches the calculation. We can do that directly in Stata.

Here is revised code.

```
. webuse grunfeld, clear
. bysort year: egen rank = rank(-mvalue), unique
. by year: egen double total5 = total((rank <= 5) * mvalue)
. by year: egen double total = total(mvalue)
. by year: egen count5 = total(rank <= 5)
. by year: egen count = count(mvalue)
. by year: generate conc5 = total5 / total
. format mvalue total5 total %3.2f
. format conc5 %4.3f
. tabdisp year, cellvar(count count5 total5 total conc5)
```

We need to negate the argument to `rank()`—note the negative sign before `mvalue`—to ensure that the values are ranked from 1 for largest onward. We also need the `unique` option to disable the default treatment of ties, which is to assign to tied values the average of ranks that would otherwise have been assigned. These are details to take care of. See the help for the `rank()` function of `egen` for details of various flavors of rank.

A positive feature of the `rank()` function in `egen` is that missing values are ignored automatically. That is welcome.

You may like this approach more than any so far.

7 Other measures

Naturally, we may want other measures too, whether standard or not. Mean, median, and geometric mean are three attractive summaries for many size measures. The geometric mean applies only to values that are all positive, which is obvious enough here but which may bite for your own application.

Let's build on the last section and show code for those three. The output is suppressed here but easily reproducible. Again, a line graph is ready to hand. Recall the advice from Section 3 about tagging just one observation for each year.

```
. webuse grunfeld, clear
. bysort year: egen rank = rank(-mvalue), unique
. by year: egen double mean5 = mean(cond(rank <= 5, mvalue, .))
. by year: egen double median5 = median(cond(rank <= 5, mvalue, .))
. by year: egen double gmean5 = mean(cond(rank <= 5, log(mvalue), .))
. replace gmean5 = exp(gmean5)
. format *n5 %4.0f
. tabdisp year, cellvar(*n5)
```

`egen` already has functions for mean and median, as you would hope and expect. The trick needed is to limit calculations to observations with ranks 1 to 5. We could use an `if` qualifier here, which would have few real disadvantages, but choose instead to show off a favorite device, using `cond()` within a function call to ignore whatever you want to ignore (Cox 2011a). See Kantor and Cox (2005) if you remain puzzled by `cond()`.

Two wider principles about `egen` calls deserve a flag. Many of its functions feed on expressions, which can be more complicated than a bare variable name. We can use any Stata functions here (although, perhaps surprisingly, not other `egen` functions). Further, `egen` functions tend to ignore missing values to the extent appropriate. Both principles are being used in the code above for mean and median. First, we instruct `egen` to treat data assigned ranks 6 upward as if they were missing. Then, we take a mean or median over values, which calculation will ignore missings.

Official Stata lacks an `egen` function for geometric means, although one has long been available in the `egenmore` package on the SSC Archive. A good excuse for any lack is how easy it is to get geometric means in two steps: first, take the mean of logged values, and then, exponentiate that mean. A similar trick would yield harmonic means in two steps.

Wainer (2005, 107) makes a neat point arising out of an analysis of Olympic performances. "The bronze medal time is the median of the top five finishers. The winning time can sometimes not be representative, whereas taking a median of the top five yields a more statistically stable measure." The point has further applications not only within the wide, wide world of sports but also beyond. See also Wainer, Njue, and Palmer (2000).

8 The smallest five

What about the smallest five? Adapting what has been said already should be simple or can be regarded as an exercise.

The convention in `egen, rank()` that lowest values rank 1 up is convenient in this case. Note that missing values can still bite, just not as often.

9 Efficiency matters?

With really large datasets, we might need to think more about getting results speedily. Here the main caution is that `egen` code includes extra bells and whistles that you can avoid if you know what you are doing. So, for example, a total can be obtained using the Stata function `sum()`, which yields cumulative or running sums. Here we revisit the last such line:

```
. by year: egen double total5 = total((rank <= 5) * mvalue)
```

A non-`egen` solution could be

```
. by year: generate double total5 = sum((rank <= 5) * mvalue)
. by year: replace total5 = total5[_N]
```

That's two lines of code. How could it be faster? If you look inside the previous code with

```
. viewsource egen.ado
. viewsource _gtotal.ado
```

you will see that the two-step is at the heart of the `total()` function any way, but if we avoid `egen`, we avoid many lines that parse user input and cope with yet other details that can be omitted in a direct solution.

10 Conclusion

This column will naturally be most interesting and useful if the examples are close to what you want to do. Beyond that, I have a larger tale to tell.

A serious subject need not entail a solemn or stuffy style. But banter about coding tricks should not distract from a positive theme. Again and again, problems that are simple to understand can be coded up with fairly simple Stata code using a small set of devices and methods, such as

- sorting and ranking
- `if` and `in` to select observations
- `by:` as a framework for groupwise calculations

- indicator variables as a mode of selection
- `egen` as a Swiss Army knife with many handy functions

Pólya (1957, 208) quipped, on behalf of the "traditional mathematics professor", "What is the difference between method and device? A method is a device you use twice." In Stata, as elsewhere, expertise grows as you learn whichever tricks are devices, or even better methods, that you can use again and again in different problems.

11 Acknowledgments

The original seeds of this column were various threads on Statalist asking related questions. Statalist remains the best source of public information on what researchers using Stata want to do.

12 References

Beirlant, J., Y. Goegebeur, J. Segers, and J. Teugels. 2004. *Statistics of Extremes: Theory and Applications*. New York: Wiley.

Berger, W. H., and F. L. Parker. 1970. Diversity of planktonic foraminifera in deep-sea sediments. *Science* 168: 1345–1347. https://doi.org/10.1126/science.168.3937.1345.

Coles, S. 2001. *An Introduction to Statistical Modeling of Extreme Values*. London: Springer.

Cox, N. J. 2002. Speaking Stata: How to move step by: step. *Stata Journal* 2: 86–102. https://doi.org/10.1177/1536867X0200200106.

———. 2003. Speaking Stata: Problems with tables, Part I. *Stata Journal* 3: 309–324. https://doi.org/10.1177/1536867X0300300308.

———. 2005. Speaking Stata: The protean quantile plot. *Stata Journal* 5: 442–460. https://doi.org/10.1177/1536867X0500500312.

———. 2011a. Speaking Stata: Compared with .... *Stata Journal* 11: 305–314. https://doi.org/10.1177/1536867X1101100210.

———. 2011b. Speaking Stata: Fun and fluency with functions. *Stata Journal* 11: 460–471. https://doi.org/10.1177/1536867X1101100308.

———. 2012. Speaking Stata: Output to order. *Stata Journal* 12: 147–158. https://doi.org/10.1177/1536867X1201200109.

———. 2016. Speaking Stata: Truth, falsity, indication, and negation. *Stata Journal* 16: 229–236. https://doi.org/10.1177/1536867X1601600117.

Cox, N. J., and G. M. Longton. 2008. Speaking Stata: Distinct observations. *Stata Journal* 8: 557–568. https://doi.org/10.1177/1536867X0800800408.

Cox, N. J., and C. B. Schechter. 2019. Speaking Stata: How best to generate indicator or dummy variables. *Stata Journal* 19: 246–259. https://doi.org/10.1177/1536867X19830921.

Davison, A. C., and R. Huser. 2015. Statistics of extremes. *Annual Review of Statistics and Its Application* 2: 203–235. https://doi.org/10.1146/annurev-statistics-010814-020133.

Dooley, J. F. 2018. *History of Cryptography and Cryptanalysis: Codes, Ciphers, and Their Algorithms*. Cham, Switzerland: Springer.

Dunin, E., and K. Schmeh. 2020. *Codebreaking: A Practical Guide*. London: Robinson.

Embrechts, P., C. Klüppelberg, and T. Mikosch. 1997. *Modelling Extremal Events for Insurance and Finance*. Berlin: Springer.

Good, I. J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika* 40: 237–264. https://doi.org/10.2307/2333344.

———. 1982. Diversity as a concept and its measurement: Comment. *Journal of the American Statistical Association* 77: 561–563. https://doi.org/10.2307/2287710.

Gould, W. 2003. Stata tip 3: How to be assertive. *Stata Journal* 3: 448. https://doi.org/10.1177/1536867X0400300414.

Gumbel, E. J. 1958. *Statistics of Extremes*. New York: Columbia University Press.

Holden, J. 2017. *The Mathematics of Secrets: Cryptography from Caesar Cyphers to Digital Encryption*. Princeton, NJ: Princeton University Press.

Jeffreys, H. 1961. *Theory of Probability*. 3rd ed. London: Oxford University Press.

Kantor, D., and N. J. Cox. 2005. Depending on conditions: A tutorial on the cond() function. *Stata Journal* 5: 413–420. https://doi.org/10.1177/1536867X0500500310.

Leinster, T. 2021. *Entropy and Diversity: The Axiomatic Approach*. Cambridge: Cambridge University Press.

MacKay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press.

Pólya, G. 1957. *How to Solve It: A New Aspect of Mathematical Method*. 2nd ed. Princeton, NJ: Princeton University Press.

Reiss, R.-D., and M. Thomas. 2007. *Statistical Analysis of Extreme Values with Applications to Insurance, Finance, Hydrology, and Other Fields*. 3rd ed. Basel: Birkhäuser.

Resnick, S. I. 2007. *Heavy-Tail Phenomena: Probabilistic and Statistical Modeling*. New York: Springer.

Wainer, H. 2005. *Graphic Discovery: A Trout in the Milk and Other Visual Adventures*. Princeton, NJ: Princeton University Press.

Wainer, H., C. Njue, and S. Palmer. 2000. Assessing time trends in sex differences in swimming & running. *Chance* 13(1): 10–15. https://doi.org/10.1080/09332480.2000.10542184.

Whittle, P. 1992. *Probability via Expectation*. 3rd ed. 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 16 commands in official Stata. He was an author of several inserts in the *Stata Technical Bulletin* and is Editor-at-Large 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).