The Stata Journal (2018)
18, Number 2, pp. 345–356

Simulating the central limit theorem

Marshall A. Taylor
Department of Sociology
University of Notre Dame
Notre Dame, IN
mtaylo15@nd.edu

Abstract. Understanding the central limit theorem is crucial for comprehending parametric inferential statistics. Despite this, undergraduate and graduate students alike often struggle with grasping how the theorem works and why researchers rely on its properties to draw inferences from a single unbiased random sample. In this article, I outline a new command, `sdist`, that can be used to simulate the central limit theorem by generating a matrix of randomly generated normal or nonnormal variables and comparing the true sampling distribution standard deviation with the standard error from the first randomly generated sample. The user also has the option of plotting the empirical sampling distribution of sample means, the first random variable distribution, and a stacked visualization of the two distributions.

Keywords: st0525, sdist, central limit theorem, simulation, runiform(), teaching

1 Introduction

Virtually every introductory statistics course discusses the central limit theorem (CLT). Far from simply being an abstract mathematical concept that can be relegated to a footnote or a term in statistics history to which instructors pay a fleeting tribute, the CLT is the bedrock upon which parametric inferential statistics stand. Without the CLT, parametric tests—from the simple t test to advanced statistical models—make little sense as tools for estimating population dynamics. Put simply but seriously, the CLT is what allows frequentist statisticians to do what they do.

Despite the theorem's prevalence in the classroom, there is no guiding method for teaching the subject. The CLT can often be difficult to communicate effectively in a classroom setting with a textbook (Dyck and Gee 1998), and numerous pedagogical tools have been proposed to promote hands-on learning (for example, Aberson et al. [2000]; Dinov, Christou, and Sanchez [2008]; Matz and Hause [2008]; Price and Zhang [2007]; Schoenfelder et al. [2007]). A number of these tools involve field exercises, such as having students collect data outside the classroom. While innovative and kinesthetically engaging, these strategies can be difficult to implement if the material must be covered quickly (such as in statistics "boot camps" or short courses) or if other contextual factors preclude the instructor from being able to use such involved and time-intensive strategies. Further, given that the power of the CLT becomes more evident as the number of samples (and number of observations within the samples) grows asymptotically, strategies that involve the collection or distribution of physical materials are con-

strained by the number of material items at hand (for example, M&Ms [Dyck and Gee 1998] or rose blooms [Schoenfelder et al. 2007]). Furthermore, though a number of computer-based tools are available for simulating the CLT (for example, Caro [2015]; Rice Virtual Lab in Statistics [2017]; Dinov, Christou, and Sanchez [2008]), these still tend to be either restrictive in terms of the number of random samples that can be generated or web based and therefore not applicable on machines without an Internet connection. Finally, what both material- and computer-based methods lack is hands-on interaction with the statistical software that students will inevitably be using during (and hopefully after) their statistics course.[1]

In this article, I outline a simple new command, `sdist`, that can be used to simulate the CLT within the Stata environment. This is accomplished by 1) generating a matrix of randomly generated normal or nonnormal variables, 2) plotting the associated empirical sampling distribution of sample means, 3) comparing the true sampling distribution standard deviation with the standard error from the first randomly generated variable, and 4) producing a side-by-side comparison of the two distributions. The command allows the student to alter the number of random samples, the number of observations per sample, the type of distribution from which the samples are drawn, and the parameters of the distribution. The code is purposefully kept simple to promote student experimentation of the simulator outside the classroom.

In what follows, I first describe the simulation procedure performed by `sdist`. I then outline the syntax, options, and outputs of the command. I close with an illustration of `sdist` in action.

2 Simulation procedure

The `sdist` command compares an empirical sampling distribution of means from a matrix of randomly generated variables (samples) with an empirical distribution of one of the random variables used to generate the sampling distribution. Let $\mathbf{X}$ be an $n \times r$ matrix consisting of n observations and r random variables,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{21} & x_{31} & \cdots & x_{r1} \\ x_{12} & x_{22} & x_{32} & \cdots & x_{r2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & x_{3n} & \cdots & x_{rn} \end{bmatrix}$$

where each x_{ij} cell entry is a randomly generated number from some distribution (normal or nonnormal). Each cell entry is independent of every other cell.

1. The Statistical Consulting Group at the UCLA Institute for Digital Education and Research (2015) provides an interactive GUI-based CLT simulator in Stata called `clt`. Although `clt` offers more parent distribution types and is considerably faster than `sdist` when more samples are drawn, it does not offer the ability to visualize or report the mean, standard deviation, or standard error for one of the observed variables that goes into the sampling distribution of sample means. This feature, which `sdist` has, is particularly important when trying to illustrate the asymptotic normality characteristics of the CLT with nonnormal variables. `clt` also does not appear to be functional on Mac operating systems (or at least newer versions of them).

From here, a row vector, $\mathbf{x}$, of sample (column) means is generated from $\mathbf{X}$, where the ith sample mean in the vector is found with

$$\overline{x}_i = \frac{x_{i1} + x_{i2} + \cdots + x_{in}}{n}$$

This row vector is then transposed to make a column vector $\mathbf{x}^T$ of $\overline{x}_r$ means:

$$\mathbf{x}^T = \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \overline{x}_3 \\ \vdots \\ \overline{x}_r \end{bmatrix}$$

The frequency distribution of $\mathbf{x}^T$ is the empirical sampling distribution of r random-sample means, each generated from n observations.

The standard deviation of this sampling distribution, $\sigma_{\overline{x}}$, is then calculated as

$$\sigma_{\overline{x}} = \sqrt{\frac{\sum_{i=1}^{r}(\overline{x} - \mu_{\overline{x}})^2}{r}}$$

where $\mu_{\overline{x}}$ is the mean of the empirical sampling distribution.

This "true" standard deviation of the sampling distribution is then compared with the standard-error estimate from $x_1 : \text{se}_{\overline{x}} = s_1/\sqrt{n}$, where s_1 is the standard deviation of x_1. In addition to $\sigma_{\overline{x}}$ and $\text{se}_{\overline{x}}$, a Δ estimate is also reported,

$$\Delta = |\sigma_{\overline{x}} - \text{se}_{\overline{x}}|$$

where Δ is simply the absolute difference between the true standard deviation of the sampling distribution and its standard-error estimate—the value of which, per the asymptotic properties of the CLT, will shrink as r, n, or both increase.

3 The sdist command

`sdist` performs the above simulation using a combination of base Stata functions native to at least Stata 13.1. The reliance on base functions is intentional and meant to promote quick classroom implementation. The syntax for `sdist`, outlined below, was also kept simple for this purpose. A consequence of this emphasis on simplicity is that only a selection of graphical parameters from the `histogram` and `graph combine` commands are available for customization. Future updates to the command may increase the number of customizable graphical parameters.

3.1 Syntax

```
sdist [ , samples(#) obs(#) type(string) par1(#) par2(#) round(#)
    histplot saveplot1(string) saveplot2(string) repplot combine
    lcolor(string) fcolor(string) bckg(string) nlcolor(string) nlwidth(#)
    nlpattern(string) dots ]
```

3.2 Options

`samples(#)` specifies the number of r random samples to generate. The default is `samples(200)`.

`obs(#)` is the number of n observations per sample. The default is `obs(500)`.

`type(`*string*`)` is the type of distribution from which the random samples should be drawn. The default is `type(uniform)`, which generates random samples from a rectangular uniform distribution. Normal and Poisson distributions are also available, indicated by `type(normal)` and `type(poisson)`, respectively. The distributions are created through calls to Stata's random-number generators (see [FN] **Random-number functions**).

`par1(#)` is the first parameter to be specified depending on the distribution selected in `type()`. Because the default `type()` is the rectangular distribution, the default is the lower end of the $[a, b)$ interval. The samples are generated through the `runiform()` function (see [M-5] **runiform( )**), so the default for a is 0, but this can be changed. If `type(normal)` is selected, this parameter is the mean, with a default of `par1(0)`. This parameter does not specify anything if `type(poisson)` is selected; use `par2()` to specify the mean of the Poisson distribution instead.

`par2(#)` is the second parameter to be specified depending on the distribution selected in `type()`. Because the default `type()` is the rectangular distribution, the default is the higher end of the $[a, b)$ interval. The samples are generated through the `runiform()` function, so the default for b is (an approximation of) 1, but this can be changed. If `type(normal)` is selected, this parameter is the standard deviation, with a default of `par2(1)`. If `type(poisson)` is selected, this parameter is the mean, also with a default of `par2(0)`.

`round(#)` is the decimal point to which $\mu_{\overline{x}}$, $\sigma_{\overline{x}}$, $\overline{x}_1$, s_1, $\mathrm{se}_{\overline{x}}$, and Δ should be rounded. The default is `round(0.001)`.

`histplot` indicates whether histograms of the $\mathbf{x}^T$ and x_1 frequency distributions should be plotted. The plots are generated through Stata's `histogram` command (see [R] **histogram**). The default is no histogram.

`saveplot1(`*string*`)` indicates whether the $\mathbf{x}^T$ histogram should be saved and the name for the plot. The default is `saveplot1(plot1.gph)` if `repplot` is specified. This is ignored if `histplot` is not specified. The default is to not save the plot.

`saveplot2(`*string*`)` serves the same purpose as `saveplot1()`, but with reference to the x_1 histogram. The default is `saveplot2(plot2.gph)` if `repplot` is specified. This is ignored if `histplot` is not specified. The default is to not save the plot.

`repplot` specifies whether the saved histograms should replace existing saved histograms in the same directory with the same name. `repplot` will default to saving both plots if neither `saveplot1()` nor `saveplot2()` is specified, using `plot1.gph` and `plot2.gph` as the filenames, respectively. The default is to not replace plots.

`combine` indicates whether the two histograms should be stacked to form a third plot. This is a call to the `graph combine` command (see [G-2] **graph combine**). Both histograms have to be saved for the graphs to be combined, either by specifying both `saveplot1()` and `saveplot2()` simultaneously or by specifying `repplot` without either the `saveplot1()` or the `saveplot2()` option (though `repplot` can still be used in conjunction with both `saveplot1()` and `saveplot2()` if both are specified). Requiring that both `saveplot1()` and `saveplot2()` or `repplot` only be specified prevents the program from erroneously stacking histograms from different simulations. The default is to not stack the plots.

`lcolor(`*string*`)` indicates the outline color of the histogram bars. This is a call to the `lcolor()` option of the `histogram` command. The default is `lcolor(black)`.

`fcolor(`*string*`)` indicates the interior color of the histogram bars. This is a call to the `fcolor()` option of the `histogram` command. The default is `fcolor(gs6)`.

`bckg(`*string*`)` indicates the color of the graph region background. This is a call to the `graphregion(fcolor())` option (see [G-3] ***region_options***) of the `histogram` command. The default is `bckg(white)`.

`nlcolor(`*string*`)` indicates the color of the normal curve line. This is a call to the `normopts(lcolor())` option of the `histogram` command. The default is `nlcolor(black)`.

`nlwidth(`#`)` indicates the thickness of the normal curve line. This is a call to the `normopts(lwidth())` option of the `histogram` command. The default is `nlwidth(0.5)`.

`nlpattern(`*string*`)` indicates the pattern of the normal curve line. This is a call to the `normopts(lpattern())` option of the `histogram` command. The default is `nlpattern(solid)`.

`dots` indicates whether the program should show simulation progress using the `_dots` command. The default is no dots.

3.3 Output[2]

By design, `sdist` will not run if there are any data in memory. If data are in memory, the program will error out and inform users that they need to clear any data before running `sdist`. After users clear all data and execute the command, `sdist` returns a simple table with $\sigma_{\overline{x}}$, $\text{se}_{\overline{x}}$, and Δ. The table also includes a note on the Δ estimate, explaining why the difference between $\sigma_{\overline{x}}$ and $\text{se}_{\overline{x}}$ may be as large (or small) as it is (for example, if r or n is large or small). Below is an example of the output printed to the Results window.

```
. set seed 544
. sdist, histplot samples(500) type(poisson) par2(1)
> saveplot1(emp_dist.gph) saveplot2(single_dist.gph) repplot combine
```

	sd/se
sig_Xb	.045
se_Xb	.047
abs(diff)	.002

```
The difference between sig_Xb and se_Xb is .002. The larger
this difference, the poorer the single X variable standard error approximates
the standard deviation of the sampling distribution. This may be due to one
of two things: a small number of samples and/or a small sample size.
```

By issuing the `histplot` option, users generate ready-made plots of the $\mathbf{x}^T$ and x_1 distributions—complete with $\mu_{\overline{x}}$ and $\sigma_{\overline{x}}$ for the sampling distribution and $\overline{x}_1$, s_1, and $\text{se}_{\overline{x}}$ for the variable distribution. An example of the plot is provided in figure 1 below, which was generated from the above command. While figure 1 (the combined plot one gets after specifying the `combine` option) is not automatically saved, the two histograms composing it would have been saved because the user specified the `saveplot1()` and `saveplot2()` options. In this case, the hypothetical user chose to name the $\mathbf{x}^T$ histogram `emp_dist.gph` and the x_1 histogram `single_dist.gph`.

2. The examples in this article were created on a machine running Stata 14.2. The examples will output slightly different results when using a Stata version earlier than 14 because a new random-number generator was introduced with Stata 14 (Gopal 2016).

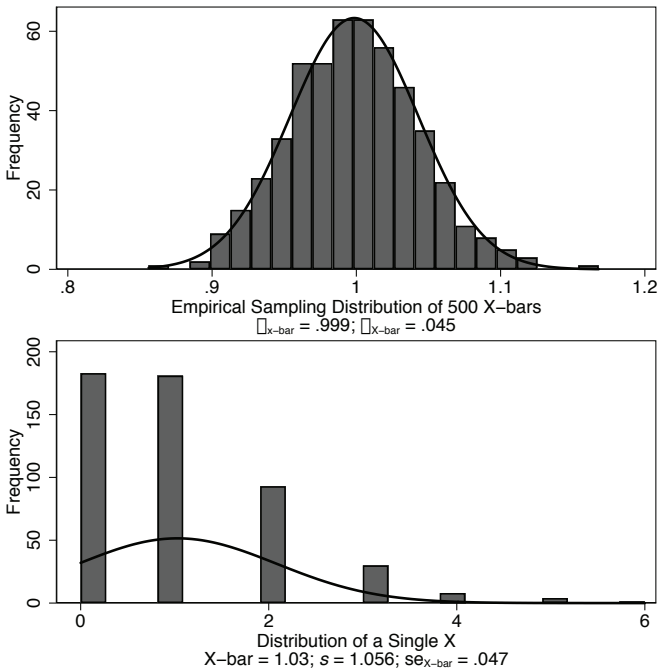

Figure 1. The empirical sampling distribution (top) and the first variable distribution (bottom) from a Poisson parent distribution

If the user had specified `repplot` without explicitly saving either of the histograms, `sdist` would have defaulted to the assumption that the user was wanting to save the plots (otherwise, he or she would not have attempted to replace other existing files by issuing `repplot` in the first place). Therefore, the two plots would have been saved with the generic names `plot1.gph` and `plot2.gph`. Note, however, that the `combine` option will never work when the two histograms are not saved in some way (through some combination of `saveplot1()`, `saveplot2()`, or `repplot`), because `combine` assumes that the plots have been saved. If `repplot` had been issued with `saveplot1()` but not `saveplot2()`, then only the first graph would overwrite any preexisting file with the same name in the directory. The reverse would also be the case if one issues `repplot` with `saveplot2()`.

4 Example

`sdist` requires an empty dataset to maximize sample-size flexibility and prevent other variables from inadvertently being implemented into the simulation procedure. If the user attempts to execute the `sdist` command with data loaded in memory, the command will error out with the following explanation:

```
. sdist
Save and/or clear existing data before running -sdist-.
r(4);
```

The random variables and any of their associated classes of objects are wiped from memory after the simulation is completed. The user may also use the `set seed` command before executing `sdist` to re-create any results.

As a hypothetical example, let us say an instructor wishes to illustrate the CLT property of asymptotic normality by comparing the differences between the standard deviation of an empirical sampling distribution of sample means and the standard error from a single sample using two sampling distributions of different sizes: one with an r of 500 and another with an r of 10,000 (both with a sample size of 2,000 and both from a rectangular parent distribution). To run the first simulation, the instructor would simply run `sdist` and set the `samples()` parameter to 500 and the `obs()` parameter to 2,000. The program would then return the following:

```
. set seed 2144
. sdist, histplot samples(500) obs(2000) repplot combine
```

	sd/se
sig_Xb	.007
se_Xb	.006
abs(diff)	.001

```
The difference between sig_Xb and se_Xb is .001. The larger
this difference, the poorer the single X variable standard error approximates
the standard deviation of the sampling distribution. This may be due to one
of two things: a small number of samples and/or a small sample size.
```

The output tells us that the standard deviation of the empirical sampling distribution, $\sigma_{\overline{x}}$, is 0.007, and the standard-error estimate of that standard deviation, $\text{se}_{\overline{x}}$, from the first sample in the simulation is 0.006. The absolute difference between these two numbers, Δ, highlights the fact that the standard error from just one of our samples—which is nonnormal (see figure 2 below, which we get after combining `histplot`, `repplot`, and `combine`)—is nonetheless a good estimate of the sampling distribution standard deviation that it seeks to approximate.

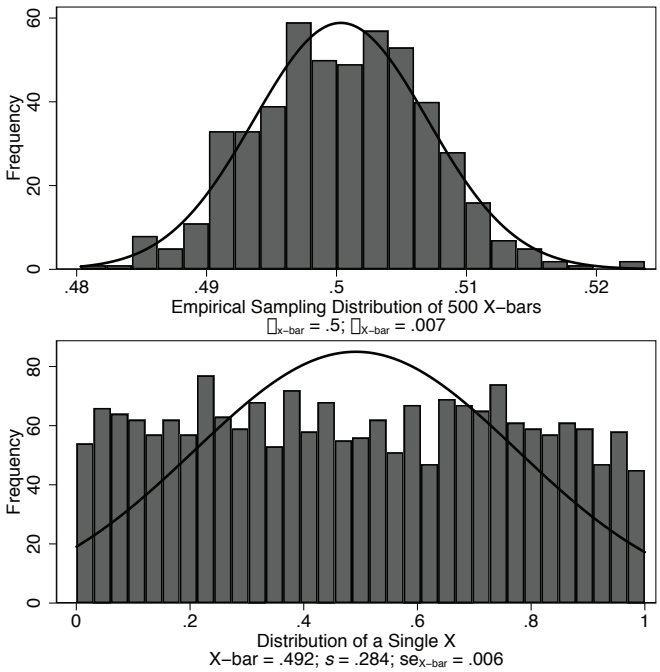

Figure 2. The empirical sampling distribution and variable distribution from a simulation with $r = 500$ and $n = 2000$

The key property of the CLT—that the sampling distribution of sample means for a variable with a sufficiently large-sample size will be approximately normal, regardless of the shape of the variable's population distribution or observed sample distribution—is visualized in the plots that can be generated through `sdist`. The main parameters of interest from the `sdist` table are also featured in the plot, along with the sampling distribution mean, the individual sample mean, and the individual sample standard deviation.

Though the example with an r of 500 does a decent job by itself of illustrating how large-sampling distributions with sufficient sample sizes approach normality, the CLT property of asymptotic normality is made that much clearer when juxtaposing figure 2 with the simulation featuring an r of 10,000:

```
. set maxvar 32767
. set seed 4816
. sdist, samples(10000) obs(2000) histplot saveplot1(emp_dist.gph)
> saveplot2(single_dist.gph) repplot combine
```

	sd/se
sig_Xb	.006
se_Xb	.006
abs(diff)	0

```
The difference between sig_Xb and se_Xb is 0. The larger
this difference, the poorer the single X variable standard error approximates
the standard deviation of the sampling distribution. This may be due to one
of two things: a small number of samples and/or a small sample size.
```

In addition to noting the smoothness around the normal curve with the larger simulation (see figure 3), the instructor could also point out the diminishing presence of outliers and explain how, when taking averages from multiple samples, the impact of influential observations diminishes as sample means cluster around the center of the sampling distribution—especially when the sampling distribution grows larger.

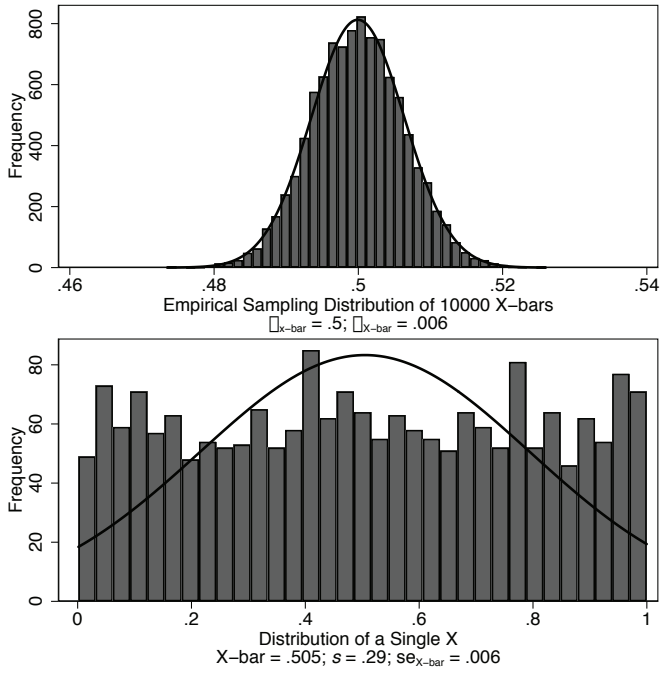

Figure 3. The empirical sampling distribution and variable distribution from a simulation with $r = 10000$ and $n = 2000$

5 Conclusion

The CLT is fundamental to statistical practice and education. Without quality knowledge of how and why it works, the inferential power of parametric statistics can be difficult to grasp. Though numerous pedagogical tools are available to promote hands-on learning, these strategies are often constrained by the availability of material resources or, if computer based, are not featured within the statistical computing environments within which students will inevitably have to gain experience. The command detailed here, `sdist`, addresses both of these shortcomings by 1) using simulations that are constrained only by the memory allowances of the user's Stata software and 2) using a simple syntax structure that promotes in-class and at-home experimentation among students new to the Stata environment.

There are, of course, aspects of `sdist` that can be improved. For instance, because the program requires variable space, larger simulations can be performed only in Stata/SE or Stata/MP. At a certain point in the simulation, Stata must be able to hold $r \times 3$ variables in memory. As such, the 10,000 sample example above required space for 30,000 variables—something that Stata/IC and Small Stata cannot provide.[3] This memory demand also means the processing speed decreases with larger simulations; thus, users are encouraged to prerun larger simulations prior to instruction. Hopefully, these limitations will be addressed in future iterations of the command.

The CLT is simultaneously deceptively simple and deceptively complex. `sdist` can hopefully provide those students and practitioners working within Stata an opportunity to experiment and play around with the theorem, figuring out what exactly makes it tick.

6 Acknowledgments

The author wishes to thank the editors, an anonymous reviewer, and Richard Williams for providing important and helpful advice on the command.

7 References

Aberson, C. L., D. E. Berger, M. R. Healy, D. J. Kyle, and V. L. Romero. 2000. Evaluation of an interactive tutorial for teaching the central limit theorem. *Teaching of Psychology* 27: 289–291.

Caro, R. J. 2015. Central limit theorem simulator. https://rpubs.com/RamiroJC/CLT_Slides.

Dinov, I. D., N. Christou, and J. Sanchez. 2008. Central limit theorem: New SOCR applet and demonstration activity. *Journal of Statistics Education* 16(2). http://ww2.amstat.org/publications/jse/v16n2/dinov.pdf.

3. The maximum number of random samples Stata/IC can handle is therefore about 682. Small Stata maxes out around 33.

Dyck, J. L., and N. R. Gee. 1998. A sweet way to teach students about the sampling distribution of the mean. *Teaching of Psychology* 25: 192–195.

Gopal, K. 2016. How to generate random numbers in Stata. The Stata Blog: Not Elsewhere Classified. https://blog.stata.com/2016/03/10/how-to-generate-random-numbers-in-stata/.

Institute for Digital Research and Education. 2015. clt: Central limit theorem demonstration. UCLA: Statistical Consulting Group. https://stats.idre.ucla.edu/stat/stata/ado/teach.

Matz, D. C., and E. L. Hause. 2008. "Dealing" with the central limit theorem. *Teaching of Psychology* 35: 198–200.

Price, B. A., and X. Zhang. 2007. The power of doing: A learning exercise that brings the central limit theorem to life. *Decision Sciences* 5: 405–411.

Rice Virtual Lab in Statistics. 2017. Sampling distributions. http://onlinestatbook.com/stat_sim/sampling_dist/.

Schoenfelder, E., R. Olson, M. Bell, and K. Tom. 2007. Stop and smell the roses: An activity for teaching the central limit theorem. *Psychology Learning & Teaching* 6: 80–84.

About the author

Marshall A. Taylor is a PhD candidate in the Department of Sociology at the University of Notre Dame and a doctoral affiliate with the Kellogg Institute for International Studies. His research rests at the intersection of culture and cognition, social movements, and computational social science. His dissertation focuses on theorizing and empirically examining the mechanisms through which white nationalist organizations in the U.S. South distribute their attention to various grievances and to other members of their social movement field. His work can be found in journals such as *Sociological Theory*, *Poetics*, *Journal of Classical Sociology*, *Deviant Behavior*, and the *Stata Journal*.