The Stata Journal

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

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

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

http://www.stata-journal.com

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

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

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

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year university library subscription	$125	1-year university library subscription	$165
2-year university library subscription	$215	2-year university library subscription	$295
3-year university library subscription	$315	3-year university library subscription	$435
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

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

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

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

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

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

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

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

The Stata Journal (2013)
13, Number 1, pp. 21–38

Versatile sample-size calculation using simulation

Richard Hooper
Centre for Primary Care and Public Health
Queen Mary, University of London
London, UK
r.l.hooper@qmul.ac.uk

Abstract. I present a new Stata command, `simsam`, that uses simulation to determine the sample size required to achieve a given statistical power for any hypothesis test under any probability model that can be programmed in Stata. `simsam` returns the smallest sample size (or smallest multiple of 5, 10, or some other user-specified increment) so that the estimated power exceeds the target. The user controls the precision of the power estimate, and power is reported with a confidence interval. The sample size returned is reliable to the extent that if `simsam` is repeated, it will, nearly every time, give a sample size no more than one increment away.

Keywords: st0282, simsam, sample size, power, simulation

1 Introduction

The statistical power of a research study is the probability that it will find evidence of an important effect. Power depends on what we mean by "important", on what counts as evidence, and on how the study is designed; but given these specifications, power depends on parameters of the design such as the sample size. In the life sciences, choosing a sample size on the basis of the power it achieves is an important ethical consideration when planning research (Newell 1978; Altman 1980).

Many methods for calculating sample size, including those for comparing independent samples using a t test or χ^2 test, rely on an approximation for the relationship between sample size and power (Florey 1993; Campbell, Julious, and Altman 1995). Although the exact power achieved by a given sample size can be estimated for any hypothesis test using Monte Carlo simulation (Feiveson 2002; Eng 2004), using simulation to determine the sample size required to achieve a given power is slightly more complicated, necessitating that power be estimated at different sample sizes to find the one at which the target power is attained. This computational burden may be the reason simulation is not used more routinely as a tool for sample-size calculation. A general approach for determining sample size by simulation in Stata has been described by Feiveson (2002, 2009), and a sample-size calculator of this kind has been developed by Browne, Golalizadeh Lahi, and Parker (2009) for random-effects models in MLwiN and R, but practical and versatile software tools have not to date been widely available.

I describe a new Stata command, `simsam`, that uses a novel, iterative algorithm that uses simulation to determine the sample size required to achieve a given power. The user controls the precision with which power is estimated, but in early iterations, the algorithm uses less precision to make more rapid progress. `simsam` assumes that code for generating and analyzing a single dataset is provided in a separate program; thus `simsam` can calculate sample size for any method of analysis under any probability model that can be programmed in Stata. In fact, although the term "sample size" is used throughout this article, `simsam` can be used to determine any design parameter that, when increased (with all other parameters fixed), causes the power to increase. There could be more than one parameter, such as duration of recruitment and duration of follow-up or number of clusters and number of participants per cluster.

2 The simsam command

2.1 Syntax

`simsam` *subcommand n_option_name*`, inc(#) prec(#) [power(#) alpha(#)`
`detect(`*options*`) null(`*options*`) assuming(`*options*`) start(#) iter(#)`
`notable pvalue(`*name*`) level(#) ]`

or

`simsam continue [, inc(#) prec(#) null(`*options*`) iter(#) notable]`

2.2 Options

`inc(#)` specifies the sample-size increment. `simsam` returns a sample size that is a multiple of the increment. This option is required except with `simsam continue`, where if `inc()` is not specified, it is assumed to be the same as in the previous `simsam` command. `inc()` must be an integer (`simsam` will only consider integer values for the sample size).

`prec(#)` specifies the precision of the final estimate of power. This option is required except with `simsam continue`, where if `prec()` is not specified, it is assumed to be the same as in the previous `simsam` command. The precision is the half-width of the confidence interval for power, whose level is set by the `level()` option. The actual confidence interval is calculated using an exact binomial method, so it will not always match the specified precision exactly.

`power(#)` specifies the target for statistical power as a decimal fraction. The default is `power(0.9)`.

`alpha(#)` specifies the significance level. The default is `alpha(0.05)`.

`detect(`*options*`)` lists the options to *subcommand* that specify the effect to be detected.

`null(`*options*`)` lists the options to *subcommand* that specify the null model.

`assuming(`*options*`)` lists any other options to *subcommand* that specify additional assumptions.

`start(#)` specifies the starting value for sample size in the iterative algorithm. The default is `start(100)`. The algorithm can generally find its way to any required sample size, large or small, from a starting value of 100, but sometimes a judicious choice of starting value will ensure quicker convergence.

`iter(#)` specifies the maximum number of iterations. The default is `iter(10)`, which will be sufficient in many applications. The largest number that can be specified is `iter(99)`.

`notable` suppresses the output table containing results from each iteration.

`pvalue(`*name*`)` identifies the returned scalar containing the p-value. The default is `pvalue(p)`, which means that after execution of *subcommand*, `simsam` looks for the p-value in `r(p)`.

`level(#)` specifies the confidence level for estimates of power. The default is `level(99)` to ensure good coverage and to emphasize the distinction from sample-based estimates, which would usually be quoted with 95% confidence. The level can be set to any integer between 90 and 99, but precision will usually be controlled using the `prec()` option, so `level()` can be kept at its default.

2.3 Saved results

`simsam` saves the following scalars and macros in `r()`. Some of these are used by `simsam continue` to pick up where `simsam` left off; therefore, users should not use another `rclass` command after `simsam` if they intend to use `simsam continue`.

Scalars

`r(n)`	required sample size	`r(tryn)`	sample size to try next
`r(p)`	estimate of power at sample size n	`r(nullp)`	estimate of power under null
`r(pl)`	lower confidence limit for power	`r(nullpl)`	lower confidence limit for power under null
`r(pu)`	upper confidence limit for power	`r(nullpu)`	upper confidence limit for power under null
`r(alpha)`	significance level	`r(level)`	confidence level for estimates of power
`r(power)`	target power		
`r(inc)`	increment	`r(prec_inc)`	precision and increment ratio
`r(prec)`	precision		
`r(reps)`	number of replications used to estimate power		

Macros

`r(subcomm)`	*subcommand*	`r(assuming)`	additional options
`r(noption)`	name of option for sample size	`r(ntried)`	sample sizes tried so far, at full precision
`r(pvalue)`	where *p*-value is returned by *subcommand*	`r(exitcond)`	exit condition
`r(detect)`	specification of the effect to be detected	`r(phase)`	phase of algorithm (heuristic or step-down)
`r(null)`	specification of the null model		

3 The subcommand

In the `simsam` syntax, *subcommand* is the name of a program containing code to generate and analyze a single dataset. *n_option_name* is the name of an option to *subcommand* that controls the sample size. Essential features of the *subcommand* program are that

(a) options follow standard syntax, and there are no arguments before the comma;

(b) data in memory are cleared before the new dataset is generated; and

(c) it is an r-class command, returning the *p*-value as a scalar.

Where speed is of the essence, the program should also be as lean as possible. Requirement (b) also allows *subcommand* to be used with Stata's `simulate` command.

▷ Example

The following program could be used as a subcommand for `simsam` to calculate sample size for a two-sample *t* test (though a faster method in this case would be to use Stata's `sampsi` command). The program generates normally distributed observations in two independent samples and then applies a *t* test. It has required options `d()` (mean difference), `sd()` (standard deviation in each group), and `npergrp()` (sample size per group).

```
. program define s_ttest, rclass
  1.         version 12.0
  2.         syntax , D(real) SD(real) NPERGRP(integer)
  3.         drop _all
  4.         set obs `=2*`npergrp´´
  5.         gen group=mod(_n, 2)
  6.         gen x=rnormal(`d´*group, `sd´)
  7.         capture noisily ttest x, by(group)
  8.         return scalar p=r(p)
  9. end
```

◁

`simsam` assumes by default that the *p*-value will be returned in `r(p)`, but an alternative can be specified with the `pvalue()` option. For example, `pvalue(p_exact)` tells `simsam` to look for the *p*-value in `r(p_exact)`. This is useful if the user wants to choose from more than one returned *p*-value, for example, exact and approximate or one-tailed and two-tailed. `simsam` also assumes that if the *p*-value is missing—for example, if an

error was captured during the analysis and a p-value was not returned—the result is to be considered nonsignificant. This allows for situations where legitimately arising data cannot be analyzed—for example, a logistic regression on a contingency table with a zero cell. Once a new program has been checked for syntax errors, errors in the analysis should be routinely captured so that they do not halt the progress of `simsam`. In particular, `capture noisily` should be captured, as in the example above, so that running *subcommand* by itself produces noisy output.

4 Basic use

In basic use, there are two required options to `simsam`: `inc()`, which specifies the increment for sample size, and `prec()`, which sets the precision of the final estimate of power (see section 2.2). A certain degree of precision relative to the increment is necessary for `simsam` to converge reliably on a solution, though this will also depend on what the required sample size turns out to be; if during its execution `simsam` suspects that there is a problem with precision, it will halt and give further advice (this essentially involves checking whether successive iterations of sample size are determined to within one increment, subject to the uncertainty in the power estimate; see section 8.4).

Options for target power and significance and their default values are the same as for Stata's `sampsi` command. The other options likely to be specified are `detect()` and `assuming()`. Their use is best illustrated with an example.

▷ Example

Suppose you want to use the `s_ttest` command defined above to calculate the sample size required to detect a mean difference of 0.5 between two independent groups using a t test, with 80% power at the 5% significance level, assuming a standard deviation in each group of 1.0. As will be seen in a later example, you can play with the increment and precision once an initial solution has been obtained; to start, you may prefer a fairly wide increment and precision—in this case an increment of 10 and a precision of 1%.

(Note: All examples in this article were run in Stata version 12. Because of changes to Stata's random-number generator, results can be version-dependent even with the same seed specified.)

```
. version 12.0
. set seed 20120301
. simsam s_ttest npergrp, power(0.8) alpha(0.05) detect(d(0.5))
> assuming(sd(1.0)) inc(10) prec(0.01)
```

```
iteration  npergrp              power (99% CI)
        1      100 ..........  0.9200 (0.8239, 0.9737)
        2       70 ..........  0.8110 (0.7771, 0.8418)
        3       70 ..........  0.8368 (0.8274, 0.8459)
        4       60 ..........  0.7766 (0.7661, 0.7870)
```

```
       npergrp = 70
        achieves 83.68% power (99% CI 82.74, 84.59)
          at the 5% significance level
    to detect
             d = 0.5
     assuming
            sd = 1.0
If continuing, use prec/inc < 2.8e-03
```

The final results show that a sample size of 70 per group achieves an estimated power of 83.68%. This is the smallest multiple of 10 for which estimated power exceeds 80%.

◁

In the example, options listed inside `detect()` and `assuming()` are all passed as options to `s_ttest`, along with the option `npergrp(`n`)`, where n is the working sample size at each iteration. In this situation, `detect()` and `assuming()` are essentially interchangeable (other than influencing how parameters are reported in the final output).

The output table updates as each iteration is completed. It can be suppressed with the `notable` option. The line of dots in each row marks the progress of the iteration. The interval between dots represents one tenth of the time required for the whole iteration. This interval gets longer in later iterations; that is, the dots appear more slowly as the iterations continue (this differs from the `simulate` command).

The algorithm used by `simsam` to determine sample size is discussed in more detail in the next section.

5 The algorithm

5.1 Background

To find the minimum sample size at which power exceeds a given threshold, we have to estimate power at more than one sample size. In the Stata examples given by Feiveson (2002, 2009) and in the MLPowSim software developed by Browne, Golalizadeh Lahi, and Parker (2009), the user specifies an upper and lower limit for sample size together with a sample-size increment; the algorithm considers every sample size in turn, starting at the lower limit and increasing by the given increment until the upper limit is reached.

Williams, Ebel, and Wagner (2007) suggested a more efficient binary search algorithm in which half the possible sample sizes are ruled out at each iteration. Starting with a range of possible sample sizes (n_1, n_2), this algorithm estimates the power at the mid-point of the range n_{mid} and then repeats the process either on the range (n_1, n_{mid}) or on the range (n_{mid}, n_2), depending on whether the power at n_{mid} is above or below the required power. Jung (2008) suggested a binary search in which the number of replications varies: it starts with 100 replications at each sample size and increases to 1,000 replications once the algorithm has narrowed down the search. `simsam` develops these ideas further using an algorithm that combines 1) an intelligent or heuristic search for sample size and 2) a progressively increasing number of replications.

5.2 Heuristic search

A binary search can rule out half the sample sizes at each iteration because the relationship between sample size and power can be assumed to be an increasing one. If we could assume something about the form of the relationship between sample size and power, it might be possible to hone in more rapidly on the desired solution. If we knew the exact relationship, we could go directly to the solution; but even if we can only guess an approximate relationship, we should still be able to find a better guess at sample size and then apply the same process iteratively to converge on a solution.

`simsam` makes a guess based on a normal approximation: it assumes the hypothesis test is based on a normally distributed estimate of effect size with standard error decreasing as one over the square root of sample size. With this assumption, `simsam` can formulate the relationship between power and sample size from a single estimate of power at a single sample size and in this way jump to a new, improved guess at sample size (see section 8.1).

5.3 Increasing the number of replications

`simsam` starts with 100 replications in the first iteration and multiplies this by 10 at each iteration until the required maximum is reached, after which it continues using the maximum number of replications at each iteration. The maximum is calculated from the precision specified by the user for the power estimate (see section 8.2). Suppose that the maximum number of replications is 1,000,000. With this scheme, we can afford to spend the first four iterations converging on a solution. As long as we aim to have converged by the time we reach 1,000,000 replications, then the earlier iterations will only have consumed 111,100 replications in total—less than 1/9 of the 1,000,000 required to estimate the power at the final iteration. This is much more efficient than using the full number of replications to evaluate every different sample size.

5.4 Terminating the search

The algorithm requires a robust approach to decide when to stop. It proceeds in two phases: a heuristic phase as described above, followed by a step-down phase in which the strategy changes to an incremental search.

During the heuristic phase, once the maximum number of replications is reached, the algorithm maintains a list of the sample sizes it has already tried, a record of the smallest estimated power exceeding the target, and the sample size that achieved this (the best sample size so far). When the algorithm determines that the next sample size is one it has already tried, it reverts to the "best" sample size and switches over from the heuristic phase to the step-down phase.

In the step-down phase, the algorithm reduces the sample size by one increment at each iteration. It stops when 1) it reaches a sample size it has already tried (for which the estimated power must have been below the target); 2) it reaches a sample size of zero; or 3) it estimates the power to be below the target. The step-down phase ensures we have found the minimum required sample size by confirming that the power at a smaller sample size is below the target.

▷ Example

The processes acting behind the scenes in the example above can now be explained. We are trying to estimate a power of 80% with a 99% confidence interval of $+/-1\%$. This requires 10,620 replications. The first iteration uses 100 replications of sample size 100 and estimates the power to be 92.00%. This results in a revised guess of 70 for the sample size. The second iteration uses 1,000 replications of sample size 70 and estimates the power to be 81.10%. This again results in a guess of 70 (rounded up to the nearest 10) for the required sample size. The third iteration uses the maximum number of replications (10,620—the algorithm skips 10,000 because this is almost the maximum anyway) with sample size 70 and estimates the power to be 83.68%.

The revised guess at the required sample size is 70 again (this is not shown in the table but can be inferred from what `simsam` does next). Because this is the second time a sample size of 70 has been suggested at the maximum number of iterations, the algorithm switches over to the step-down phase and looks instead at a sample size of 60. Power in this case is estimated to be 77.66%. Because the estimated power at sample size 70 is 83.68% and at sample size 60 is 77.66%, `simsam` concludes that the smallest sample size to the nearest 10 for which power exceeds 80% is 70.

◁

6 Continuing after a previous command

`simsam` may halt before a solution has been obtained either because it suspects a problem that might prevent convergence or because it has completed a prespecified number of iterations. In fact, `simsam` will cease iterating under one of five exit conditions:

1. the algorithm has converged according to the criteria defined in the previous section;
2. the number of iterations specified in the `iter()` option has been completed, the default being 10 (the largest number that can be specified is 99);
3. `simsam` suspects that given the precision relative to the increment specified, the sample size cannot be reliably determined to within one increment;
4. the estimated power is unexpectedly low (if the power is less than the significance level, this could indicate a problem with *subcommand*—in particular, a power of 0 may indicate the program is consistently failing to return a p-value); or
5. the working sample size is continually increasing, but the power is not being controlled as expected.

After any of these exit conditions, `simsam` may be restarted using the `simsam continue` command. `simsam continue` uses saved results to continue where `simsam` left off and does not have any required options, though it does allow the user to alter the increment and the precision before continuing. After exit conditions 1 and 3, `simsam` estimates the precision relative to the increment that is required if the user wants to continue.

Cases 4 and 5 are not considered errors because they could arise legitimately (for example, through sampling error or poor choice of starting value), but using `simsam continue` in these cases is unlikely to solve the underlying problem, and the program is liable simply to halt again. Continuing after `simsam` has converged, without changing the increment or precision, will just cause the previous result to be output again. In fact, `simsam continue` is likely to be most useful in three situations, which are illustrated with examples below: 1) stopping and restarting after a fixed number of iterations; 2) obtaining a rough solution before proceeding to a higher-precision one; and 3) amending the precision or increment to allow convergence.

▷ Example

```
. version 12.0
. set seed 20120301
. simsam s_ttest npergrp, power(0.8) alpha(0.05) detect(d(0.5)) assuming(sd(1))
> inc(10) prec(0.01) iter(2)
```

iteration	npergrp	power (99% CI)
1	100	0.9200 (0.8239, 0.9737)
2	70	0.8110 (0.7771, 0.8418)

```
Warning: did not converge within 2 iterations
. simsam continue, iter(2)
```

iteration	npergrp	power (99% CI)
1	70	0.8368 (0.8274, 0.8459)
2	60	0.7766 (0.7661, 0.7870)

```
       npergrp = 70
        achieves 83.68% power (99% CI 82.74, 84.59)
          at the 5% significance level
    to detect
             d = 0.5
     assuming
            sd = 1
If continuing, use prec/inc < 2.8e-03
```

◁

▷ Example

```
. version 12.0
. set seed 20120301
. simsam s_ttest npergrp, power(0.8) alpha(0.05) detect(d(0.5)) assuming(sd(1))
> inc(10) prec(0.01)
```

iteration	npergrp	power (99% CI)
1	100	0.9200 (0.8239, 0.9737)
2	70	0.8110 (0.7771, 0.8418)
3	70	0.8368 (0.8274, 0.8459)
4	60	0.7766 (0.7661, 0.7870)

```
       npergrp = 70
        achieves 83.68% power (99% CI 82.74, 84.59)
          at the 5% significance level
    to detect
             d = 0.5
     assuming
            sd = 1
If continuing, use prec/inc < 2.8e-03
```

```
. simsam continue, inc(1) prec(0.001)
```

iteration	npergrp	power (99% CI)
1	70	0.8383 (0.8352, 0.8412)
2	64	0.8010 (0.8000, 0.8020)
3	63	0.7949 (0.7939, 0.7959)

```
       npergrp = 64
        achieves 80.10% power (99% CI 80.00, 80.20)
          at the 5% significance level
    to detect
             d = 0.5
     assuming
            sd = 1
If continuing, use prec/inc < 3.1e-03
```

◁

▷ Example

```
. version 12.0
. set seed 20120301
. simsam s_ttest npergrp, power(0.8) alpha(0.05) detect(d(0.5)) assuming(sd(1))
> inc(1) prec(0.01)
```

iteration	npergrp	power (99% CI)
1	100	0.9200 (0.8239, 0.9737)

```
Warning: npergrp not reliably determined to within one increment
If continuing, use prec/inc < 2.8e-03
. simsam continue, inc(10) prec(0.01)
```

iteration	npergrp	power (99% CI)
1	70	0.8110 (0.7771, 0.8418)
2	70	0.8368 (0.8274, 0.8459)
3	60	0.7766 (0.7661, 0.7870)

```
       npergrp = 70
        achieves 83.68% power (99% CI 82.74, 84.59)
          at the 5% significance level
    to detect
             d = 0.5
     assuming
            sd = 1
If continuing, use prec/inc < 2.8e-03
```

◁

7 Validating a sample-size calculation

One of the advantages of `simsam` as a system for sample-size calculation is that it is relatively easy to share and to validate. This is useful, for example, if a sample-size calculation is to be included in a grant proposal.

7.1 Introduction to the example

Suppose there is a proposal for a randomized trial of a community intervention—the AARDVARK trial—that is to be randomized by household and delivered either to one adult in the household or to two if there is a married or cohabiting couple living there. The investigators assume the proportion of households with couples is 0.3; the mean difference in outcome between intervention and control groups is 0.5; the standard deviation within each group is 1.0; and the intracluster correlation is 0.5. They intend to analyze their data using random-effects regression, testing the group difference with a likelihood-ratio test. They report that a sample size of 58 households per group achieves 80% power at the 5% significance level and submit the following command, which they used with `simsam` to calculate sample size:

```
. program define aardvark, rclass
 1.         version 12.0
 2.         syntax, PCOUPLE(real) ICC(real) D(real) SD(real) NHOUSPERGRP(integer)
 3.         drop _all
 4.         set obs `=2*`nhouspergrp´´
 5.         scalar sigmaa=sqrt(`icc´)*`sd´
 6.         scalar sigmae=sqrt(1-`icc´)*`sd´
 7.         gen group=mod(_n,2)
 8.         gen k=1+(runiform()<`pcouple´)
 9.         gen ybar=rnormal(`d´*group, sigmaa)
10.         gen housid=_n
11.         expand k
12.         gen y=rnormal(ybar, sigmae)
13.         capture noisily {
14.                 xtreg y group, i(housid) mle
15.                 estimates store model1
16.                 xtreg y, i(housid) mle
17.                 estimates store model0
18.                 lrtest model1 model0
19.         }
20.         return scalar p=r(p)
21. end
```

7.2 Repeating the calculation and estimating the power under the null hypothesis

Because `simsam` only converges on a solution if it can do so reliably, repeating the `simsam` command will nearly always give the same answer (or, at worst, a sample size one increment away). Thus one way for the funding panel to validate the investigators' calculation is simply to repeat it. `simsam` also allows the user to estimate the power under the null hypothesis at the required sample size. This can be specified as an additional option, `null()`, when repeating the sample-size calculation. The example below takes some time to run (over 24 hours, depending on system specification), but in the timescale of a typical grant application, this should be no great burden (and without the efficient algorithm used by `simsam`, it might take weeks to determine the sample size by simulation).

```
. version 12.0
. set seed 20120301
. simsam aardvark nhouspergrp, power(0.8) alpha(0.05) null(d(0)) detect(d(0.5))
> assuming(sd(1.0) icc(0.5) pcouple(0.3)) inc(1) prec(0.001)
```

iteration	nhousp~p	power (99% CI)
1	100	..... 0.9900 (0.9280, 0.9999)
2	43	..... 0.6400 (0.5998, 0.6788)
3	63	..... 0.8443 (0.8347, 0.8535)
4	56	..... 0.7919 (0.7885, 0.7952)
5	58	..... 0.8024 (0.8014, 0.8033)
6	57	..... 0.7954 (0.7944, 0.7964)
null	58	..... 0.0523 (0.0512, 0.0533)

```
   nhouspergrp = 58
        achieves 80.24% power (99% CI 80.14, 80.33)
          at the 5% significance level
    to detect
             d = 0.5
     assuming
            sd = 1.0
           icc = 0.5
       pcouple = 0.3
     under null:  5.23% power (99% CI  5.12,  5.33)
If continuing, use prec/inc < 3.4e-03
```

This reproduces the investigators' solution of 58 households per group. The power under the null is estimated to be 5.23%. In general, we would expect power under the null to equal significance. Here we must consider the validity of the *subcommand* program (see section 7.3). If we are satisfied with this (and excluding sampling variation as an explanation), we must conclude that a likelihood-ratio test in this situation is slightly biased. If the sample-size calculation is repeated with a nominal significance level of 4.8% to bring the true significance level closer to 5%, then the required sample size becomes 59 households per group.

One note of caution: If power is defined as the probability of correctly rejecting the null hypothesis under a given alternative, then in a two-sided context, we need to be clear what we mean by "correctly rejecting". Power in the two-sided case is often calculated as the probability that the two-tailed p-value is significant and the observed effect is in the correct direction. For `simsam` to calculate this "directional" power, we need the subcommand to return the relevant one-sided p-value multiplied by two rather than the two-sided p-value. In this case, the power under the null would be half the significance level. In practical terms, this distinction turns out to be of little importance: when the directional power is 80%, the nondirectional power is only around 0.0001% greater using a normal approximation.

With the `null()` option in use, the distinction between `detect()` and `assuming()` is clearer. To calculate the required sample size, you pass all options in `detect()` and `assuming()` to *subcommand*. To calculate power under the null, you pass all options in `null()` and `assuming()` to *subcommand*. In other words, `assuming()` contains assumptions that hold under both the null and the alternative.

The `null()` option can also be used with `simsam continue`. If the `inc()` and `prec()` options are not set, then power under the null is calculated for the previous `simsam` solution.

7.3 Validating the subcommand

The command `aardvark` certainly appears to have face validity; that is, it seems to do what is intended. To investigate further what the command is doing, we could run it separately and follow up with additional checks:

```
. version 12.0
. set seed 20120301
. quietly aardvark, pcouple(0.3) icc(0.5) d(0.5) sd(1.0) nhouspergrp(58)
. table group, contents(mean y sd y)
```

group	mean(y)	sd(y)
0	-.1695986	.9529981
1	.4627651	1.021282

```
. bysort group: loneway y housid
```

```
-> group = 0
                    One-way Analysis of Variance for y:
                                          Number of obs =        73
                                              R-squared =    0.9582
    Source                SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between housid         62.658676     57     1.099275       6.04     0.0002
Within housid           2.732109     15     .1821406
------------------------------------------------------------------------
Total                  65.390785     72    .90820535

         Intraclass       Asy.
         correlation      S.E.       [95% Conf. Interval]
         ---------------------------------------------------
            0.80037     0.07926       0.64501     0.95572

         Estimated SD of housid effect           .8545366
         Estimated SD within housid              .4267793
         Est. reliability of a housid mean        0.83431
              (evaluated at n=1.26)
```

```
-> group = 1
                    One-way Analysis of Variance for y:
                                          Number of obs =        76
                                              R-squared =    0.9120
    Source                SS         df      MS            F     Prob > F
------------------------------------------------------------------------
Between housid         71.341771     57      1.25161       3.27     0.0037
Within housid          6.8844844     18    .38247136
------------------------------------------------------------------------
Total                  78.226255     75    1.0430167

         Intraclass       Asy.
         correlation      S.E.       [95% Conf. Interval]
         ---------------------------------------------------
            0.63477     0.12855       0.38282     0.88673

         Estimated SD of housid effect           .8153182
         Estimated SD within housid              .6184427
         Est. reliability of a housid mean        0.69442
              (evaluated at n=1.31)
```

We could also run simulations involving `aardvark` using the `simulate` command to investigate its behavior further.

7.4 Comparison with approximate methods

If there were no clustering by household, a conventional sample-size calculation would indicate that 63 participants per group were required (Campbell, Julious, and Altman 1995). If we use a standard adjustment for the "design effect" introduced by clustering (see Donner, Birkett, and Buck [1981]), calculated using the average cluster size of 1.3 and the intracluster correlation of 0.5, the required sample size per group becomes 73 participants, or 56 households. Variability in cluster size will reduce the power. An

alternative adjustment that is based on the coefficient of variation of the distribution of cluster size and is conservative is discussed by Eldridge, Ashby, and Kerry (2006). Using this adjustment, we obtain a required sample size of 78 participants per group, or 60 households. From this, we would suspect that the true required sample size was somewhere between 56 and 60 households per group, which agrees with the solution obtained by `simsam`.

8 Methods and formula

8.1 Successive iterations of sample size in the heuristic phase

In the case where the null and alternative hypotheses differ by one degree of freedom, we can imagine the hypothesis test to be based on an unbiased estimate of the effect size δ, which is roughly normally distributed with standard error $\sigma/\sqrt{n}$, n being the sample size. Under this assumption of normality, the relationship between sample size n and type II error probability β is

$$\frac{\delta}{\sigma/\sqrt{n}} = z_{1-\alpha/2} + z_{1-\beta}$$

where z_p is the pth percentile of the standard normal distribution, and α is the type I error probability, that is,

$$n = (\sigma/\delta)^2 \left(z_{1-\alpha/2} + z_{1-\beta}\right)^2$$

or equivalently

$$\frac{n}{\left(z_{1-\alpha/2} + z_{1-\beta}\right)^2} = \text{constant}$$

Suppose at iteration i we are considering sample size n_i, at which we estimate the power to be $1 - \widehat{\beta}_i$. Then we can use the above equation to calculate a new sample size n_{i+1} aimed at achieving the target power $1 - \beta^*$:

$$n_{i+1} = n_i \left(\frac{z_{1-\alpha/2} + z_{1-\beta^*}}{z_{1-\alpha/2} + z_{1-\widehat{\beta}_i}}\right)^2 = n_i f\left(\widehat{\beta}_i\right)$$

`simsam` rounds this up to the next multiple of the chosen increment.

8.2 Number of replications

The maximum number of replications r is calculated from the target power $1-\beta^*$ (or the significance if the power under the null is being calculated), the precision $\delta\beta$ (the half-width of the confidence interval), and the confidence level $1-\lambda$ using standard methods for estimating proportions:

$$r = \beta^* (1-\beta^*) \left(\frac{z_{1-\lambda/2}}{\delta\beta} \right)^2$$

`simsam` rounds this number up to the nearest multiple of 10.

8.3 Confidence intervals for power

Confidence intervals for power are calculated by the Stata command `cii` using an exact binomial method; see [R] **ci**.

8.4 Required ratio of precision to increment

After each iteration, `simsam` assesses whether the required sample size can be reliably determined to within one increment. To do this, it imagines the power to be estimated as $1-\beta^*$ (confidence interval $[1-\beta^*-\delta\beta, 1-\beta^*+\delta\beta]$) at the most up-to-date guess at required sample size n_{i+1} and looks at what the difference would be between the next iteration of sample size at the upper and lower confidence limits, respectively. This difference should be less than the increment m for there to be confidence that the next iteration of sample size does not change by more than one increment:

$$\{ f(\beta^* + \delta\beta) - f(\beta^* - \delta\beta) \} n_{i+1} < m$$

If this condition is not met, `simsam` halts. In this case, and also when it converges on a solution, `simsam` provides the user with a suggested precision-to-increment ratio. It does this by approximating the left-hand side of the above inequality using the derivative of f with respect to β, which is $\dot{f}$, and using its best guess at sample size $\widehat{n}$ (either n_{i+1} if halted prematurely or the solution on which it has converged). Thus

$$2\delta\beta \dot{f}(\beta^*) \widehat{n} < m$$

that is,

$$\frac{\delta\beta}{m} < 1 / \left\{ 2\widehat{n} \dot{f}(\beta^*) \right\} = \frac{\left(z_{1-\alpha/2} + z_{1-\beta^*} \right) e^{-z_{1-\beta^*}^2/2}}{4\sqrt{2\pi}\widehat{n}}$$

9 References

Altman, D. G. 1980. Statistics and ethics in medical research: III How large a sample? *British Medical Journal* 281: 1336–1338.

Browne, W. J., M. Golalizadeh Lahi, and R. M. A. Parker. 2009. A guide to sample size calculations for random effect models via simulation and the MLPowSim software package. http://www.bristol.ac.uk/cmm/software/mlpowsim/mlpowsim-manual.pdf.

Campbell, M. J., S. A. Julious, and D. G. Altman. 1995. Estimating sample sizes for binary, ordered categorical, and continuous outcomes in two group comparisons. *British Medical Journal* 311: 1145–1148.

Donner, A., N. Birkett, and C. Buck. 1981. Randomization by cluster: Sample size requirements and analysis. *American Journal of Epidemiology* 114: 906–914.

Eldridge, S. M., D. Ashby, and S. Kerry. 2006. Sample size for cluster randomized trials: Effect of coefficient of variation of cluster size and analysis method. *International Journal of Epidemiology* 35: 1292–1300.

Eng, J. 2004. Sample size estimation: A glimpse beyond simple formulas. *Radiology* 230: 606–612.

Feiveson, A. H. 2002. Power by simulation. *Stata Journal* 2: 107–124.

———. 2009. FAQ: How can I use Stata to calculate power by simulation? http://www.stata.com/support/faqs/statistics/power-by-simulation/.

Florey, C. D. 1993. Sample size for beginners. *British Medical Journal* 306: 1181–1184.

Jung, S.-H. 2008. Sample size calculation for paired survival data: A simulation method. *Journal of Statistical Computation and Simulation* 78: 85–92.

Newell, D. J. 1978. Type II errors and ethics (letter). *British Medical Journal* 4: 1789.

Williams, M. S., E. D. Ebel, and B. A. Wagner. 2007. Monte Carlo approaches for determining power and sample size in low-prevalence applications. *Preventive Veterinary Medicine* 82: 151–158.

About the author

Richard Hooper is a senior lecturer in medical statistics at the Centre for Primary Care and Public Health in the Blizard Institute at Queen Mary, University of London, where he is an adviser for Research Design Service London, and he is a senior statistician with the Pragmatic Clinical Trials Unit.