The Stata Journal (2010)
10, Number 3, pp. 369–385

simsum: Analyses of simulation studies including Monte Carlo error

Ian R. White
MRC Biostatistics Unit
Institute of Public Health
Cambridge, UK
ian.white@mrc-bsu.cam.ac.uk

Abstract. A new Stata command, `simsum`, analyzes data from simulation studies. The data may comprise point estimates and standard errors from several analysis methods, possibly resulting from several different simulation settings. `simsum` can report bias, coverage, power, empirical standard error, relative precision, average model-based standard error, and the relative error of the standard error. Monte Carlo errors are available for all of these estimated quantities.

Keywords: st0200, simsum, simulation, Monte Carlo error, normal approximation, sandwich variance

1 Introduction

Simulation studies are an important tool for statistical research (Burton et al. 2006), but they are often poorly reported. In particular, to understand the role of chance in results of simulation studies, it is important to estimate the Monte Carlo (MC) error, defined as the standard deviation of an estimated quantity over repeated simulation studies. However, this error is often not reported: Koehler, Brown, and Haneuse (2009) found that of 323 articles reporting the results of a simulation study in *Biometrics, Biometrika*, and the *Journal of the American Statistical Association* in 2007, only 8 articles reported the MC error.

This article describes a new Stata command, `simsum`, that facilitates analyses of simulated data. `simsum` analyzes simulation studies in which each simulated dataset yields point estimates by one or more analysis methods. Bias, empirical standard error (SE), and precision relative to a reference method can be computed for each method. If, in addition, model-based SEs are available, then `simsum` can compute the average model-based SE, the relative error in the model-based SE, the coverage of nominal confidence intervals, and the power to reject a null hypothesis. MC errors are available for all estimated quantities.

2 The simsum command

2.1 Syntax

`simsum` accepts data in wide or long format.

In wide format, data contain one record per simulated dataset, with results from multiple analysis methods stored as different variables. The appropriate syntax is

`simsum` *estvarlist* [*if*] [*in*] [`, true(`*expression*`)` *options*]

where *estvarlist* is a *varlist* containing point estimates from one or more analysis methods.

In long format, data contain one record per analysis method per simulated dataset, and the appropriate syntax is

`simsum` *estvarname* [*if*] [*in*] [`, true(`*expression*`)` `methodvar(`*varname*`)` `id(`*varlist*`)` *options*]

where *estvarname* is a variable containing the point estimates, `methodvar(`*varname*`)` identifies the method, and `id(`*varlist*`)` identifies the simulated dataset.

2.2 Options

Main options

`true(`*expression*`)` gives the true value of the parameter. This option is required for calculations of bias and coverage.

`methodvar(`*varname*`)` specifies that the data are in long format and that each record represents one analysis of one simulated dataset using the method identified by *varname*. The `id()` option is required with `methodvar()`. If `methodvar()` is not specified, the data must be in wide format, and each record represents all analyses of one simulated dataset.

`id(`*varlist*`)` uniquely identifies the dataset used for each record, within levels of any by-variables. This is a required option in the long format. The `methodvar()` option is required with `id()`.

`se(`*varlist*`)` lists the names of the variables containing the SEs of the point estimates. For data in long format, this is a single variable.

`seprefix(`*string*`)` specifies that the names of the variables containing the SEs of the point estimates be formed by adding the given prefix to the names of the variables containing the point estimates. `seprefix()` may be combined with `sesuffix(`*string*`)` but not with `se(`*varlist*`)`.

sesuffix(*string*) specifies that the names of the variables containing the SEs of the point estimates be formed by adding the given suffix to the names of the variables containing the point estimates. `sesuffix()` may be combined with `seprefix(`*string*`)` but not with `se(`*varlist*`)`.

Data-checking options

`graph` requests a descriptive graph of SEs against point estimates.

nomemcheck turns off checking that adequate memory is free. This check aims to avoid spending calculation time when `simsum` is likely to fail because of lack of memory.

`max(#)` specifies the maximum acceptable absolute value of the point estimates, standardized to mean 0 and standard deviation 1. The default is `max(10)`.

`semax(#)` specifies the maximum acceptable value of the SE as a multiple of the mean SE. The default is `semax(100)`.

`dropbig` specifies that point estimates or SEs beyond the maximum acceptable values be dropped; otherwise, the command halts with an error. Missing values are always dropped.

`nolistbig` suppresses listing of point estimates and SEs that lie outside the acceptable limits.

`listmiss` lists observations with missing point estimates or SEs.

Calculation options

`level(#)` specifies the confidence level for coverages and powers. The default is `level(95)` or as set by `set level`; see [R] **level**.

`by(`*varlist*`)` summarizes the results by *varlist*.

`mcse` reports MC errors for all summaries.

`robust` requests robust MC errors (see section 4) for the statistics `empse`, `relprec`, and `relerror`. The default is MC errors based on an assumption of normally distributed point estimates. `robust` is only useful if `mcse` is also specified.

modelsemethod(`rmse`|`mean`) specifies whether the model SE should be summarized as the root mean squared value (`modelsemethod(rmse)`, the default) or as the arithmetic mean (`modelsemethod(mean)`).

`ref(`*string*`)` specifies the reference method against which relative precisions will be calculated. With data in wide format, *string* must be a variable name. With data in long format, *string* must be a value of the method variable; if the value is labeled, the label must be used.

Options specifying degrees of freedom

The number of degrees of freedom is used in calculating coverages and powers.

`df(`*string*`)` specifies the degrees of freedom. It may contain a number (to apply to all methods), a variable name, or a list of variables containing the degrees of freedom for each method.

`dfprefix(`*string*`)` specifies that the names of the variables containing the degrees of freedom be formed by adding the given prefix to the names of the variables containing the point estimates. `dfprefix()` may be combined with `dfsuffix(`*string*`)` but not with `df(`*string*`)`.

`dfsuffix(`*string*`)` specifies that the names of the variables containing the degrees of freedom be formed by adding the given suffix to the names of the variables containing the point estimates. `dfsuffix()` may be combined with `dfprefix(`*string*`)` but not with `df(`*string*`)`.

Statistic options

If none of the following options are specified, then all available statistics are computed.

`bsims` reports the number of simulations with nonmissing point estimates.

`sesims` reports the number of simulations with nonmissing SEs.

`bias` estimates the bias in the point estimates.

`empse` estimates the empirical SE, defined as the standard deviation of the point estimates.

`relprec` estimates the relative precision, defined as the inverse squared ratio of the empirical SE of this method to the empirical SE of the reference method. This calculation is slow; omitting it can reduce run time by up to 90%.

`modelse` estimates the model-based SE. See `modelsemethod()` above.

`relerror` estimates the proportional error in the model-based SE, using the empirical SE as the gold standard.

`cover` estimates the coverage of nominal confidence intervals at the specified level.

`power` estimates at the specified level the power to reject the null hypothesis that the true parameter is zero.

Output options

`clear` loads the summary data into memory.

`saving(`*filename*`)` saves the summary data into *filename*.

`nolist` suppresses listing of the results and is allowed only when `clear` or `saving()` is specified.

`listsep` lists results using one table per statistic, giving output that is narrower and better formatted. The default is to list the results as a single table.

`format(`*string*`)` specifies the format for printing results and saving summary data. If `listsep` is also specified, then up to three formats may be specified: 1) for results on the scale of the original estimates (`bias`, `empse`, and `modelse`), 2) for percentages (`relprec`, `relerror`, `cover`, and `power`), and 3) for integers (`bsims` and `sesims`). The default is the existing format of the (first) estimate variable for 1 and 2 and `%7.0f` for 3.

`sepby(`*varlist*`)` invokes this `list` option when printing results.

`abbreviate(#)` invokes this `list` option when printing results.

`gen(`*string*`)` specifies the prefix for new variables identifying the different statistics in the output dataset. `gen()` is only useful with `clear` or `saving()`. The default is `gen(stat)` so that the new identifiers are, for example, `statnum` and `statcode`.

3 Example

This example is based on, but distinct from, a simulation study comparing different ways to handle missing covariates when fitting a Cox model (White and Royston 2009). One thousand datasets were simulated, each containing normally distributed covariates x and z and a time-to-event outcome. Both covariates had 20% of their values deleted independently of all other variables so the data became missing completely at random (Little and Rubin 2002). Each simulated dataset was analyzed in three ways. A Cox model was fit to the complete cases (`CC`). Then two methods of multiple imputation using chained equations (van Buuren, Boshuizen, and Knook 1999), implemented in Stata as `ice` (Royston 2004, 2009), were used. The `MI_LOGT` method multiply imputes the missing values of x and z with the outcome included as $\log(t)$ and d, where t is the survival time and d is the event indicator. The `MI_T` method is the same except that $\log(t)$ is replaced by t in the imputation model. The results are stored in long format, with variable `dataset` identifying the simulated dataset number, string variable `method` identifying the method used, variable `b` holding the point estimate, and variable `se` holding the SE. The data start like this:

(*Continued on next page*)

	dataset	method	b	se
1.	1	CC	.7067682	.14651
2.	1	MI_T	.6841882	.1255043
3.	1	MI_LOGT	.7124795	.1410814
4.	2	CC	.3485008	.1599879
5.	2	MI_T	.4060082	.1409831
6.	2	MI_LOGT	.4287003	.1358589
7.	3	CC	.6495075	.1521568
8.	3	MI_T	.5028701	.130078
9.	3	MI_LOGT	.5604051	.1168512

They are then summarized thus:

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
dataset	3000	500.5	288.7231	1	1000
method	0				
b	3000	.5054995	.1396257	-.1483829	1.004529
se	3000	.1375334	.0183683	.0907097	.2281933

`simsum` produces the following output:

```
. simsum b, se(se) methodvar(method) id(dataset) true(0.5) mcse
> format(%6.3f %6.1f %6.0f) listsep
Reshaping data to wide format ...
Starting to process results ...
Non-missing point estimates
```

CC	MI_LOGT	MI_T
1000	1000	1000

Non-missing standard errors

CC	MI_LOGT	MI_T
1000	1000	1000

Bias in point estimate

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
0.017	0.005	0.001	0.004	-0.001	0.004

Empirical standard error

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
0.151	0.003	0.132	0.003	0.134	0.003

% gain in precision relative to method CC

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
.	.	31.0	3.9	26.4	3.8

RMS model-based standard error

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
0.147	0.001	0.135	0.001	0.134	0.001

Relative % error in standard error

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
-2.7	2.2	2.2	2.3	-0.4	2.3

Coverage of nominal 95% confidence interval

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
94.3	0.7	94.9	0.7	94.3	0.7

Power of 5% level test

CC	(MCse)	MI_LOGT	(MCse)	MI_T	(MCse)
94.6	0.7	96.9	0.5	96.3	0.6

Some points of interest include the following:

- Table 3: `CC` has small-sample bias away from the null.
- Tables 4 and 5: `CC` is inefficient compared with `MI_LOGT` and `MI_T`.
- Comparing tables 4 and 6 shows that model-based SEs are close to the empirical values. This is shown more directly in table 7.
- Table 8: Coverage of nominal 95% confidence intervals also seems fine, which is not surprising in view of the lack of bias and good model-based SEs.
- Table 9: `CC` lacks power compared with `MI_LOGT` and `MI_T`, which is not surprising in view of its inefficiency.

If different formatting of the results is required, the results can be loaded into memory using the `clear` option and can then be manipulated.

4 Formulas

Assume that the true parameter is β and that the ith simulated dataset ($i = 1, \ldots, n$) yields a point estimate $\widehat{\beta}_i$ with SE s_i. Define

$$\begin{aligned}
\overline{\beta} &= \frac{1}{n}\sum_i \widehat{\beta}_i \\
V_{\widehat{\beta}} &= \frac{1}{n-1}\sum_i \left(\widehat{\beta}_i - \overline{\beta}\right)^2 \\
\overline{s^2} &= \frac{1}{n}\sum_i s_i^2 \\
V_{s^2} &= \frac{1}{n-1}\sum_i \left(s_i^2 - \overline{s^2}\right)^2
\end{aligned}$$

Performance of $\widehat{\beta}$: Bias and empse

Bias is defined as $E(\widehat{\beta}_i) - \beta$ and estimated by

$$\begin{aligned}
\text{estimated bias} &= \overline{\beta} - \beta \\
\text{MC error} &= \sqrt{V_{\widehat{\beta}}/n}
\end{aligned} \tag{1}$$

Precision is measured by the empirical standard deviation $\text{SD}\left(\widehat{\beta}_i\right)$ and is estimated by

$$\begin{aligned}
\text{empirical standard deviation} &= \sqrt{V_{\widehat{\beta}}} \\
\text{MC error} &= \sqrt{V_{\widehat{\beta}}/2(n-1)}
\end{aligned}$$

assuming $\widehat{\beta}$ is normally distributed, as then $(n-1)V_{\widehat{\beta}}/\text{var}\left(\widehat{\beta}\right) \sim \chi^2_{n-1}$.

Estimation method comparison: relprec

In a small change of notation, consider two estimators $\widehat{\beta}_1$ and $\widehat{\beta}_2$ with values $\widehat{\beta}_{1i}$ and $\widehat{\beta}_{2i}$ in the ith simulated dataset. The relative gain in precision for $\widehat{\beta}_2$ compared with $\widehat{\beta}_1$ is

$$\begin{aligned} \text{relative gain in precision} &= V_{\widehat{\beta}_1}/V_{\widehat{\beta}_2} \\ \text{MC error} &\approx \frac{2V_{\widehat{\beta}_1}}{V_{\widehat{\beta}_2}}\sqrt{\frac{1-\rho_{12}^2}{n-1}} \end{aligned}$$

where ρ_{12} is the correlation of $\widehat{\beta}_1$ with $\widehat{\beta}_2$.

The MC error expression can be proved by observing the following: 1) $\text{var}\left(\log V_{\widehat{\beta}_1}\right) = \text{var}\left(\log V_{\widehat{\beta}_2}\right) = 2/(n-1)$; 2) $\text{var}\left\{\log\left(V_{\widehat{\beta}_1}/V_{\widehat{\beta}_2}\right)\right\} = 4(1-\rho_V)/(n-1)$ where $\rho_V = \text{corr}\left(V_{\widehat{\beta}_1}, V_{\widehat{\beta}_2}\right)$; and 3) $\rho_V = \rho_{12}^2$. Result 3 may be derived by observing that $V_{\widehat{\beta}} \approx 1/n\sum_i\left(\widehat{\beta}_i - \beta\right)^2$ so that under a bivariate normal assumption for $\left(\widehat{\beta}_1, \widehat{\beta}_2\right)$,

$$\begin{aligned} n\ \text{cov}\left(V_{\widehat{\beta}_1}, V_{\widehat{\beta}_2}\right) &\approx \text{cov}\left\{\left(\widehat{\beta}_1-\beta_1\right)^2, \left(\widehat{\beta}_2-\beta_2\right)^2\right\} \\ &= \text{cov}\left\{\left(\widehat{\beta}_1-\beta_1\right)^2, E\left[\left(\widehat{\beta}_2-\beta_2\right)^2 \,\middle|\, \widehat{\beta}_1\right]\right\} \\ &= \text{cov}\left\{\left(\widehat{\beta}_1-\beta_1\right)^2, \rho_{12}^2 V_{\widehat{\beta}_2}/V_{\widehat{\beta}_1}\left(\widehat{\beta}_1-\beta_1\right)^2\right\} \\ &= 2\rho_{12}^2 V_{\widehat{\beta}_1} V_{\widehat{\beta}_2} \end{aligned}$$

where the third step follows because $\left(\widehat{\beta}_2-\beta_2\right) \,\middle|\, \widehat{\beta}_1$ is normal with mean $\rho_{12}\sqrt{V_{\widehat{\beta}_2}/V_{\widehat{\beta}_1}}\left(\widehat{\beta}_1-\beta_1\right)$ and constant variance.

Performance of model-based SE s_i: modelse and relerror

The average model-based SE is (by default) computed on the variance scale, because standard theory yields unbiased estimates of the variance, not of the standard deviation.

$$\begin{aligned} \text{average model-based SE } \overline{s} &= \sqrt{\overline{s^2}} \\ \text{MC error} &\approx \sqrt{V_{s^2}/4n\overline{s^2}} \end{aligned}$$

using the Taylor series approximation $\text{var}\,(X) \approx \text{var}\left(X^2\right)/4E(X)^2$.

We can now compute the relative error in the model-based SE as

$$\text{relative error} = \overline{s}/\sqrt{V_{\widehat{\beta}}} - 1 \tag{2}$$

$$\text{MC error} \approx \left(\overline{s}/\sqrt{V_{\widehat{\beta}}}\right)\sqrt{V_{s^2}/\left(4n\overline{s}^4\right) + 1/2(n-1)} \tag{3}$$

assuming that $\overline{s}$ and $V_{\widehat{\beta}}$ are approximately uncorrelated and using a further Taylor approximation.

However, if the `modelsemethod(mean)` option is used, the formulas are

$$\text{average model-based SE } \overline{s} = \frac{1}{n}\sum_i s_i$$

$$\text{MC error} = \sqrt{\frac{1}{n}\sum_i \left(s_i - \overline{s}\right)^2}$$

with consequent adjustments to equations (2) and (3).

Joint performance of $\widehat{\beta}$ and s_i: Cover and power

Let $z_{\alpha/2}$ be the critical value from the normal distribution, or (if the number of degrees of freedom has been specified) the critical value from the appropriate t distribution. The coverage of a nominal $100(1-\alpha)\%$ confidence interval is

$$\text{coverage } C = \frac{1}{n}\sum_i 1\left(\left|\,\widehat{\beta}_i - \beta\,\right| < z_{\alpha/2}s_i\right)$$

$$\text{MC error} = \sqrt{C(1-C)/n}$$

where $1(\cdot)$ is the indicator function. The power of a significance test at the α level is

$$\text{power } P = \frac{1}{n}\sum_i 1\left(\left|\,\widehat{\beta}_i\,\right| \geq z_{\alpha/2}s_i\right)$$

$$\text{MC error} = \sqrt{P(1-P)/n}$$

Robust MC errors

Several of the MC errors presented above require a normality assumption. Alternative approximations can be derived using an estimating equations method. The empirical standard deviation, $\sqrt{V_{\widehat{\beta}}}$, can be written as the solution $\widehat{\theta}$ of the equation

$$\sum_i \left\{\frac{n}{n-1}\left(\widehat{\beta}_i - \overline{\beta}\right)^2 - \widehat{\theta}^2\right\} = 0$$

The relative precision of $\widehat{\beta}_2$ compared with $\widehat{\beta}_1$ can be written as the solution $\widehat{\theta}$ of

$$\sum_i \left\{ \left(\widehat{\beta}_{1i} - \overline{\beta}_1\right)^2 - \left(\widehat{\theta} + 1\right) \left(\widehat{\beta}_{2i} - \overline{\beta}_2\right)^2 \right\} = 0$$

The relative error in the model-based SE can be written as the solution $\widehat{\theta}$ of

$$\sum_i \left\{ s_i^2 - \left(\widehat{\theta} + 1\right)^2 \left(\widehat{\beta}_i - \overline{\beta}\right)^2 \right\} = 0$$

provided that the `modelsemethod(rmse)` method is used. (If `modelsemethod(mean)` is specified, it is ignored in computing robust MC errors.) Ignoring the uncertainty in the sample means $\overline{\beta}$, $\overline{\beta}_1$, and $\overline{\beta}_2$, each estimating equation is of the form

$$\sum_i \left\{ T_i - f\left(\widehat{\theta}\right) B_i \right\} = 0$$

so the sandwich variance (White 1982) is given by

$$\widehat{\text{var}} \left\{ f\left(\widehat{\theta}\right) \right\} \approx \sum_i \left\{ T_i - f\left(\widehat{\theta}\right) B_i \right\}^2 \left(\sum_i B_i \right)^{-2}$$

and using the delta method,

$$\widehat{\text{var}} \left(\widehat{\theta}\right) \approx \widehat{\text{var}} \left\{ f\left(\widehat{\theta}\right) \right\} / f'\left(\widehat{\theta}\right)^2$$

Finally, as an attempt to allow for uncertainty in the sample means, we multiply the sandwich variance by $n/(n-1)$. A rationale is that this agrees exactly with (1) if the method is applied to the MC error of the bias. However, most simulation studies are large enough that this correction is unimportant.

5 Evaluations

Most of the formulas used by `simsum` to compute MC errors involve approximations, so I evaluated them in two simulation studies.

5.1 Multiple imputation, revisited

First, I repeated 250 times the simulation study described in section 3. The data have the same format as before, with a new variable, `simno`, identifying the 250 different simulation studies. I ran `simsum` twice. In the first run, each quantity and its MC error was computed in each simulation study:

```
. simsum b, true(0.5) methodvar(method) id(dataset) se(se) mcse by(simno)
> bias empse relprec modelse relerror cover power nolist clear
Reshaping data to wide format ...
Starting to process results ...
Results are now in memory.
```

The data are now held in memory, with one record for each statistic for each of the 250 simulation studies. The statistics are identified by the values of a newly created numerical variable `statnum`, and the different simulation studies are still identified by `simno`. The variables `bCC`, `bMI_LOGT`, and `bMI_T` contain the analysis results for the three methods. MC errors in variables are suffixed with `_mcse`. In the second run, these values are treated as ordinary output from a simulation study, and the average calculated MC error is compared with the empirical MC error.

```
. simsum bCC bMI_LOGT bMI_T, sesuffix(_mcse) by(statnum) mcse gen(newstat)
> empse modelse relerror nolist clear

Warning: found 250 observations with missing values
Starting to process results ...
Results are now in memory.
```

The 250 observations with missing values refer to the relative precisions, which are missing for the reference method (`CC`). Average calculated MC errors for each statistic are compared in table 1 with empirical MC errors. The calculated MC errors are naturally similar to those reported in the single simulation study above (some values have been multiplied by 1,000 for convenience). Empirical MC errors are close to the model-based values. The only exception is for coverage, where the model-based MC errors appear rather small for methods `CC` and `MI_LOGT`. This is likely to be a chance finding, because there is no doubt about the accuracy of the model-based MC formula for this statistic.

Table 1. Simulation study comparing three ways to handle incomplete covariates in a Cox model: Comparison of average calculated MC error (Calc) with empirical MC error (Emp) for various statistics

Statistic[1]	`CC` method			`MI_LOGT` method			`MI_T` method		
	Emp	Calc	% error[2]	Emp	Calc	% error[2]	Emp	Calc	% error[2]
Bias × 1000	4.79	4.74	−1.1 (4.4)	4.23	4.17	−1.3 (4.4)	4.22	4.18	−1.0 (4.4)
EmpSE × 1000	3.37	3.36	−0.3 (4.5)	3.11	2.95	−5.2 (4.3)	3.11	2.96	−4.9 (4.3)
RelPrec	.	.	.	4.21	3.97	−5.7 (4.2)	4.13	3.97	−4.1 (4.3)
ModSE × 1000	0.52	0.50	−3.1 (4.3)	0.59	0.59	0.3 (4.5)	0.60	0.59	−3.1 (4.4)
RelErr	2.16	2.22	2.9 (4.6)	2.40	2.34	−2.6 (4.4)	2.43	2.33	−4.2 (4.3)
Cover	0.62	0.70	13.5 (5.1)	0.61	0.68	11.3 (5.0)	0.67	0.68	1.8 (4.6)
Power	0.74	0.73	−1.4 (4.4)	0.59	0.59	−0.4 (4.5)	0.60	0.59	−2.4 (4.4)

[1] Statistics are abbreviated as follows: Bias, bias in point estimate; EmpSE, empirical SE; RelPrec, % gain in precision relative to method `CC`; ModSE, RMS model-based SE; RelErr, relative % error in SE; Cover, coverage of nominal 95% confidence interval; Power, power of 5% level test.
[2] Relative % error in average calculated SE, with its MC error in parentheses.

5.2 Nonnormal joint distributions

In a second evaluation, I simulated 100,000 datasets of size $n = 100$ from the model $X \sim N(0, 1)$, $Y \sim \text{Bern}(0.5)$. I then estimated the parameter β in the logistic regression model

$$\text{logit } P(Y = 1 \mid X) = \alpha + \beta X \tag{4}$$

in two ways: 1) $\widehat{\beta}_{\text{LR}}$ was the maximum likelihood estimate from fitting the logistic regression model (4), and 2) $\widehat{\beta}_{\text{LDA}}$ was the estimate from linear discriminant analysis (LDA), fitting the linear regression model

$$X \mid Y \sim N\left(\gamma + \delta X, \sigma^2\right)$$

and taking $\widehat{\beta}_{\text{LDA}} = \widehat{\delta}/\widehat{\sigma}^2$.

The 100,000 datasets were divided into 100 simulation studies each of 1,000 simulated datasets. The quantities described above and their SEs were calculated for each simulation study, except that power for testing $\beta = 0$ was not computed because this null hypothesis was true. Finally, the empirical MC error of each quantity across simulation studies was compared with the average MC error estimated within each simulation study.

Results are shown in table 2. The calculated MC error is adequate for all quantities except for the relative precision of LDA compared with logistic regression, for which the calculated SE is some three times too small. This appears to be due to the nonnormal joint distribution of the parameter estimates shown in figure 1. The robust MC errors perform well in all cases.

Table 2. Simulation study comparing LDA with logistic regression: Comparison of empirical with average calculated MC errors for various statistics

Quantity	Method	Mean	MC error		
			Empirical	Average calculated	
				Normal	Robust
Bias × 1000	Logistic	0.41	6.79	6.71	.
	LDA	0.41	6.66	6.57	.
Empirical SE × 1000	Logistic	212.00	4.78	4.74	5.07
	LDA	207.86	4.69	4.65	4.97
% gain in precision	Logistic	.	.	.	.
	LDA	4.027	0.124	0.048	0.131
Model SE × 1000	Logistic	207.32	0.51	0.51	.
	LDA	203.12	0.48	0.47	.
% error in model SE	Logistic	−2.16	2.13	2.20	2.26
	LDA	−2.23	2.18	2.20	2.30
% coverage	Logistic	95.36	0.60	0.66	.
	LDA	94.70	0.64	0.71	.

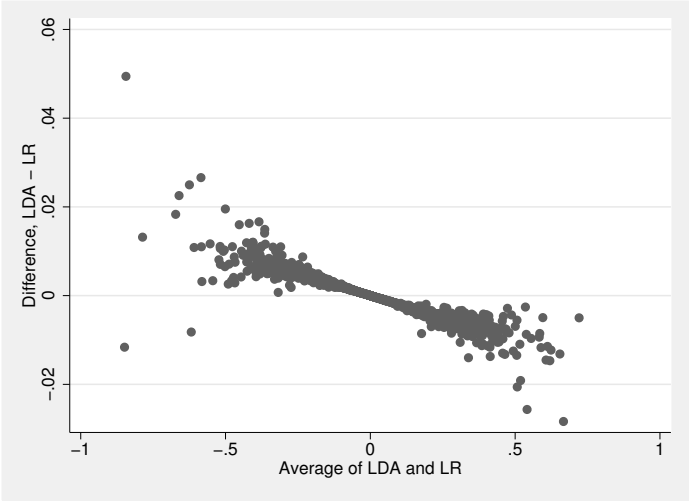

Figure 1. Scatterplot of the difference $\widehat{\beta}_{\mathrm{LDA}} - \widehat{\beta}_{\mathrm{LR}}$ against the average $\left(\widehat{\beta}_{\mathrm{LDA}} + \widehat{\beta}_{\mathrm{LR}}\right)/2$ in 2,000 simulated datasets

6 Discussion

I hope that `simsum` will help statisticians improve the reporting of their simulation studies. In particular, I hope `simsum` will help them think about and report MC errors. If MC errors are too large to enable the desired conclusions to be drawn, then it is usually straightforward to increase the sample size, a luxury rarely available in applied research.

For three statistics (empirical SE, and relative precision and relative error in model-based SE), I have proposed two approximate MC error methods, one based on a normality assumption and one based on a sandwich estimator. The MC error should only be taken as a guide, so errors of some 10–20% in calculating the MC error are of little importance. In most cases, both MC error methods performed adequately. However, the normality-based MC error was about three times too small when evaluating the relative precision of two estimators with a highly nonnormal joint distribution (figure 1). It is good practice to examine the marginal and joint distributions of parameter estimates in simulation studies, and this practice should be used to guide the choice of MC error method.

Other methods are available for estimating MC errors. Koehler, Brown, and Haneuse (2009) proposed more computationally intensive techniques that are available for implementation in R. Other software (Doornik and Hendry 2009) is available with an econometric focus.

7 Acknowledgment

This work was supported by MRC grant U.1052.00.006.

8 References

Burton, A., D. G. Altman, P. Royston, and R. L. Holder. 2006. The design of simulation studies in medical statistics. *Statistics in Medicine* 25: 4279–4292.

Doornik, J. A., and D. F. Hendry. 2009. *Interactive Monte Carlo Experimentation in Econometrics Using PcNaive 5*. London: Timberlake Consultants Press.

Koehler, E., E. Brown, and S. J.-P. A. Haneuse. 2009. On the assessment of Monte Carlo error in simulation-based statistical analyses. *American Statistician* 63: 155–162.

Little, R. J. A., and D. B. Rubin. 2002. *Statistical Analysis with Missing Data*. 2nd ed. Hoboken, NJ: Wiley.

Royston, P. 2004. Multiple imputation of missing values. *Stata Journal* 4: 227–241.

———. 2009. Multiple imputation of missing values: Further update of ice, with an emphasis on categorical variables. *Stata Journal* 9: 466–477.

van Buuren, S., H. C. Boshuizen, and D. L. Knook. 1999. Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in Medicine* 18: 681–694.

White, H. 1982. Maximum likelihood estimation of misspecified models. *Econometrica* 50: 1–25.

White, I. R., and P. Royston. 2009. Imputing missing covariate values for the Cox model. *Statistics in Medicine* 28: 1982–1998.

About the author

Ian R. White is a program leader at the MRC Biostatistics Unit in Cambridge, United Kingdom. His research interests focus on handling missing data, noncompliance, and measurement error in the analysis of clinical trials, observational studies, and meta-analysis. He frequently uses simulation studies.