The Stata Journal (2003)
3, Number 4, pp. 373–385

The simulation extrapolation method for fitting generalized linear models with additive measurement error

James W. Hardin
Arnold School of Public Health
University of South Carolina
Columbia, SC 29208

Henrik Schmiediche
Department of Statistics MS-3143
Texas A&M University
College Station, TX 77843-3143

Raymond J. Carroll
Department of Statistics MS-3143
Texas A&M University
College Station, TX 77843-3143

Abstract. We discuss and illustrate the method of simulation extrapolation for fitting models with additive measurement error. We present this discussion in terms of generalized linear models (GLMs) following the notation defined in Hardin and Carroll (2003). As in Hardin, Schmiediche, and Carroll (2003), our discussion includes specified measurement error and measurement error estimated by replicate error-prone proxies. In addition, we discuss and illustrate three extrapolant functions.

Keywords: st0051, simulation extrapolation, measurement error, instrumental variables, replicate measures, generalized linear models

1 Introduction

This paper describes software for analyzing measurement error models. Software production by StataCorp was funded by a National Institutes of Health (NIH) Small Business Innovation Research Grant (SBIR). The goal of the work described in the grant is the production of software to analyze statistical models where one or more covariates are measured with error. The software development includes two major features. The first development feature is the development of the Stata program to support communication to dynamically linked user-written computer code. StataCorp was responsible for this development, and support for user-written code in the C/C++ programming languages was added to Stata version 8. Stata refers to compiled user-written code as plugins and maintains documentation on their web site at *http://www.stata.com/support/plugins*. See Hardin and Carroll (2003) for notational conventions and an introduction to the topic of measurement error models.

The project described was supported by Grant Number R44 RR12435 from the National Institutes of Health, National Center for Research Resources. The contents of this article are solely the responsibility of the authors and do not necessarily represent the official views of the National Center for Research Resources.

This paper introduces the simulation extrapolation (SIMEX) method, for addressing measurement error in generalized linear models. This technique was proposed by Cook and Stefanski (1995) and extended in Carroll et al. (1996) and Stefanski and Cook (1995). This method shares the simplicity of the regression calibration method and is suitable for problems with additive measurement error. SIMEX is a simulation-based method aimed at reducing bias caused by the inclusion of error-prone covariates. Estimates are obtained by adding additional measurement error; a type of resampling approach. This resampling uncovers the trend of measurement error. Once the trend is estimated, final estimates are obtained by extrapolating back to the case of no measurement error.

2 The SIMEX method

The SIMEX algorithm may be succinctly described as follows:

1. Fit the model to obtain the estimated coefficients $\widehat{\boldsymbol{\beta}}$ and an estimate of the measurement error variance $\widehat{\sigma}_u^2$.

2. Generate random pseudo errors for a scale factor θ times the estimated measurement error variance $\varepsilon \sim \mathrm{N}(0, \theta\widehat{\sigma}_u^2)$. These pseudo-errors are added to the original values of the error prone covariate. Fit the model to obtain $\widehat{\boldsymbol{\beta}}^{\{1,\theta_j\}}$. This is repeated r times to obtain the mean[†] coefficient vector $\overline{\widehat{\boldsymbol{\beta}}}^{\{\theta_j\}} = (1/r)\sum_{i=1}^{r}\widehat{\boldsymbol{\beta}}^{\{i,\theta_j\}}$.

3. The previous step is repeated for $j = 1, \ldots, k^*$ scale factors, where typically we use $\boldsymbol{\theta} = \{.5, 1, 1.5, 2\}$, though individual researchers may choose a longer list of scale factors. Using the typical list of scale factors, we have $k = 5$ estimated coefficient vectors since $k^* = 4$ for the list above, and we have the estimated coefficient vector from the initial step ($k = k^* + 1$).

4. For each regression coefficient β_m ($m = 1, \ldots, p$) in the model, we consider the estimated coefficient as a function of the scale factor θ_j for $j = 1, \ldots, k$. Formally, we specify a function $f()$ such that $\beta_m = f(\boldsymbol{\theta}, \beta_m^{\{\boldsymbol{\theta}\}})$. We estimate this relationship and then extrapolate back to final estimates $\beta_m = f(\theta_0 = -1, \beta_m^{\{\boldsymbol{\theta}\}})$ (no measurement error). Researchers are free to choose the form of the function $f()$, but we point out that there are relatively few—in this case 5—observations available to estimate the parameters of $f()$. The function $f()$ used to model the relationship between the estimated coefficient and θ is called the *extrapolant* function; the software described in the last section of this paper documents three different built-in functions from which to choose.

Subsequent to the initial model fitting, the next step of the SIMEX algorithm is the simulation step. In this step, we use simulation to create additional datasets with

[†]The median coefficient vector may be substituted; see section 5.

increasingly larger amounts of measurement error $(1+\theta)\widehat{\sigma}_u^2$; θ is a discrete set of values typically taken to be $\{0.5, 1.0, 1.5, 2.0\}$.

Although the SIMEX method is applicable to a large class of models, it is easiest to understand in the context of simple linear regression where the predictor is measured with error. We assume a model $\mathbf{Y} = \beta_0 + \beta_1 \mathbf{X}_u + \epsilon$. With measurement error, we do not observe $\mathbf{X}_u$ but instead $\mathbf{X}_w$, where $\mathbf{X}_w = \mathbf{X}_u + \mathbf{U}$ ($\mathbf{U}$ has mean zero and variance σ_u^2). In addition, we assume that $\mathbf{U}$ is independent of $\mathbf{X}_u$ and $\mathbf{Y}$. The following do-file generates data for which we can use the associated Stata software to illustrate these ideas.

```
clear
set seed 12345
set obs 100
gen xu = 5*invnorm(uniform())
gen w = xu + .5*invnorm(uniform())

gen y = 0 + 1*xu + invnorm(uniform())

local suu = .25
mat uinit = (‘suu’)

simex (y=) (xunknown: w), suuinit(uinit) seed(1)
```

The resulting estimated coefficients from running this do-file are given by

```
. simex (y=) (xunknown: w), suuinit(uinit) seed(1)
Simulation extrapolation                        No. of obs         =       100
Residual df  =        98                        Wald F(0,98)       =         .
                                                Prob > F           =         .

Variance Function: V(u) = 1                        [Gaussian]
Link Function    : g(u) = u                        [Identity]

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    xunknown |    1.013505          .       .       .            .           .
       _cons |   -.1889483          .       .       .            .           .
------------------------------------------------------------------------------
```

For the moment, we will ignore the fact that there are no estimated standard errors. The software has reported the final estimated coefficient vector, but here we want to illustrate the steps taken to reach those estimates. The `simex` command stores a matrix of the means of the estimated coefficients for each θ_j.

```
. mat list e(theta)
e(theta)[6,3]
         theta    xunknown       _cons
r1          -1   1.0135053   -.1889483
c1           0   1.0007818   -.1930212
c2          .5   .99797573   -.19624396
c3           1   .99244262   -.19533214
c4         1.5   .98341659   -.20282188
c5           2   .98346201   -.20240457
```

The first column is the scale factor θ_j for how much extra measurement error is added to the error-prone variable; in our example, this is the `w` variable. Remaining

columns show the average of the estimated coefficients from the r runs for each scale factor θ_j; by default, the software uses $r = 50$, but the user may specify a different number of runs. These final results are calculated by simulating added measurement error r times, fitting the model, and then calculating the mean coefficient vector.

The first row shows the extrapolation results (`theta`$=-1$). The second row displays the results of running the analysis where we ignore the measurement error (which we do only once). We can obtain the results corresponding to the row for `theta`$=0$ by simply running the analysis of interest (a linear regression in the present example) where we simply substitute the error-prone covariate `w` for the unknown covariate `x`.

```
. regress y w
```

Source	SS	df	MS	Number of obs =	100
				F(1, 98) =	1829.44
Model	2506.14071	1	2506.14071	Prob > F =	0.0000
Residual	134.25	98	1.36989796	R-squared =	0.9492
				Adj R-squared =	0.9486
Total	2640.39071	99	26.6706132	Root MSE =	1.1704

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
w	1.000782	.0233981	42.77	0.000	.954349	1.047215
_cons	-.1930212	.1173708	-1.64	0.103	-.4259397	.0398973

Regression calibration, as outlined in Hardin, Schmiediche, and Carroll (2003), attempts to estimate the unknown covariate and then run the analysis of interest using fitted values in place of the unknown covariate. SIMEX, on the other hand, simulates data in order to model the effect of measurement error on the fitted coefficients; this model is then used to extrapolate back to the results we would have if the covariate were known.

There are several ways to model the trend. By default, the software fits a quadratic model. Using the results from our do-file, we would consider the data

```
. list, table clean noobs
```

theta	xunknown	cons
0	1.000782	-.1930212
.5	.9979757	-.196244
1	.9924426	-.1953321
1.5	.9834166	-.2028219
2	.983462	-.2024046

For each of the fitted coefficients, we can model the trend over the `theta` variable. Since the default extrapolant function is a quadratic model, we generate the necessary data and then estimate the parameters of the extrapolant function using the `regress` command. Once the parameters of the extrapolant function are estimated, they are used to extrapolate (predict) to the case of no measurement error.

```
. gen theta2 = theta*theta
. regress xunknown theta theta2
      Source |       SS       df       MS              Number of obs =       5
-------------+------------------------------           F(  2,     2) =   14.73
       Model |  .000242399     2  .000121199           Prob > F      =  0.0636
    Residual |  .000016459     2  8.2294e-06           R-squared     =  0.9364
-------------+------------------------------           Adj R-squared =  0.8728
       Total |  .000258858     4  .000064714           Root MSE      =  .00287

------------------------------------------------------------------------------
    xunknown |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       theta |  -.0111026   .0063962    -1.74   0.225    -.0386234    .0164182
      theta2 |   .0006315   .0030668     0.21   0.856    -.0125638    .0138267
       _cons |   1.001771   .0026998   371.05   0.000     .9901549    1.013387
------------------------------------------------------------------------------
. local extrap = -1
. display _b[theta]*‘extrap’ + _b[theta2]*‘extrap’*‘extrap’ + _b[_cons]
1.0135053
. regress cons theta theta2
      Source |       SS       df       MS              Number of obs =       5
-------------+------------------------------           F(  2,     2) =    4.83
       Model |  .000064325     2  .000032163           Prob > F      =  0.1714
    Residual |  .000013309     2  6.6546e-06           R-squared     =  0.8286
-------------+------------------------------           Adj R-squared =  0.6571
       Total |  .000077634     4  .000019409           Root MSE      =  .00258

------------------------------------------------------------------------------
        cons |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       theta |  -.0044281   .0057518    -0.77   0.522    -.0291759    .0203197
      theta2 |  -.0003204   .0027578    -0.12   0.918    -.0121861    .0115453
       _cons |   -.193056   .0024278   -79.52   0.000    -.2035018   -.1826102
------------------------------------------------------------------------------
. display _b[theta]*‘extrap’ + _b[theta2]*‘extrap’*‘extrap’ + _b[_cons]
-.18894829
```

These steps highlight the results returned in the `e(theta)` matrix. The method for obtaining the coefficients is straightforward for the row where the scale factor is zero. Likewise, given the collection of (mean) coefficients for each scale factor under consideration and the form of the extrapolant function, it is easy to generate the results for the scale factor of negative one (no measurement error).

2.1 Extrapolants

The `simex` command supports three extrapolants to extrapolate back to the case of no measurement error:

1. The default *quadratic extrapolant*:

$$x^* = \beta_0 + \beta_1\theta + \beta_2\theta^2$$

This results in conservative corrections for attenuations and is (usually) numerically stable. Note that x^* is the `xunknown` variable in the previous examples—the mean of the simulation runs.

2. The *rational linear extrapolant*:

$$x^* = \beta_0 + \frac{\beta_1}{\beta_2 + \theta}$$

   This reproduces the usual method of moments estimators for the case of multiple linear regression with non-iid errors. This extrapolant is often more numerically unstable than the quadratic extrapolant. The option `rational` is used to specify the rational extrapolant function.

3. The simple *linear extrapolant*:

$$x^* = \beta_0 + \beta_1 \theta$$

   This is generally not preferable to the quadratic extrapolant, though there are cases where the linear extrapolant can do better than the quadratic extrapolant. This improvement occurs if the error is small and the extrapolation function is close to linear. The option `linear` is used to specify the linear extrapolant.

To calculate the rational extrapolant, SIMEX has a built-in nonlinear least-squares estimator. Should the rational extrapolant fail, the `simex` output will include a warning. The command then automatically switches to the quadratic extrapolant. The `simex` command options `nleps` and `nlreps` specify the tolerance and iteration limits for the nonlinear least-squares estimator.

By default, `simex` uses the mean x^* of the `srep` (default= 50) simulations. The `median` option causes `simex` to use the median x^* instead. Typically, it does not matter if the mean or median is used, but the median can protect against some single disaster in the simulation run.

Plotting extrapolants: simexplot

It is valuable for diagnostic and educational purposes to be able to see the extrapolation that resulted in the simex parameter estimate. The command `simexplot` will plot a visual representation of how the parameter estimates are derived by simultaneously showing the results and extrapolation for each of the SIMEX estimates.

The command `simexplot`, without any arguments, can be typed after a successful `simex` run to plot the extrapolation of all the parameters. Alternatively, the user can specify one or more covariates of interest. In this way, you can generate individual plots for the simex estimates. Note, however, that the extrapolation line is only drawn for the `quadratic` and `linear` extrapolants.

3 Simulated data example

The following example may be replicated by interested readers following the data generation and model-fitting commands outlined. Here, we illustrate the techniques and results for measurement error analysis with multiple covariates measured with error.

Our generated dataset comprises 500 observations with multiple covariates. This setting allows us to illustrate the extrapolation techniques of the SIMEX algorithm. The data are generated, and the analysis (with graph) may be replicated by interested readers by issuing the commands illustrated. Readers should first save important data in memory before following these steps. The `x1` and `x2` covariates are scaled to illustrate the independence of scale for the algorithm.

```
. set seed 1
. set more off
. set obs 500
obs was 0, now 500
. gen x1 = uniform()*10
. gen x2 = uniform()*5
. gen x3 = uniform()
. gen x4 = uniform()
. gen x5 = uniform()
. gen err = invnorm(uniform())
. gen y =  1*x1 + 2*x2 + 3*x3 + 4*x4 + 5 + err
. gen a1 = x3 + 0.25*invnorm(uniform())
. gen a2 = x3 + 0.25*invnorm(uniform())
. gen b1 = x4 + 0.25*invnorm(uniform())
. gen b2 = x4 + 0.25*invnorm(uniform())
. simex (y=x1 x2) (w3: a1 a2) (w4: b1 b2), mess(2) brep(99) seed(1)
Simulation extrapolation                        No. of obs         =       500
                                                Bootstraps reps    =        99

Residual df  =       495                        Wald F(4,495)      =   1890.71
                                                Prob > F           =    0.0000

Variance Function: V(u) = 1                        [Gaussian]
Link Function    : g(u) = u                        [Identity]
```

y	Coef.	Bootstrap Std. Err.	t	P>|t|	[95% Conf.	Interval]
x1	.998104	.0185552	53.79	0.000	.9616474	1.034561
x2	2.053648	.0405524	50.64	0.000	1.973972	2.133324
w3	2.761487	.2335827	11.82	0.000	2.302551	3.220423
w4	3.888912	.2330361	16.69	0.000	3.43105	4.346774
_cons	5.100601	.1890554	26.98	0.000	4.729151	5.472051

The `simexplot` command produces an illustrative graph of the estimated coefficients.

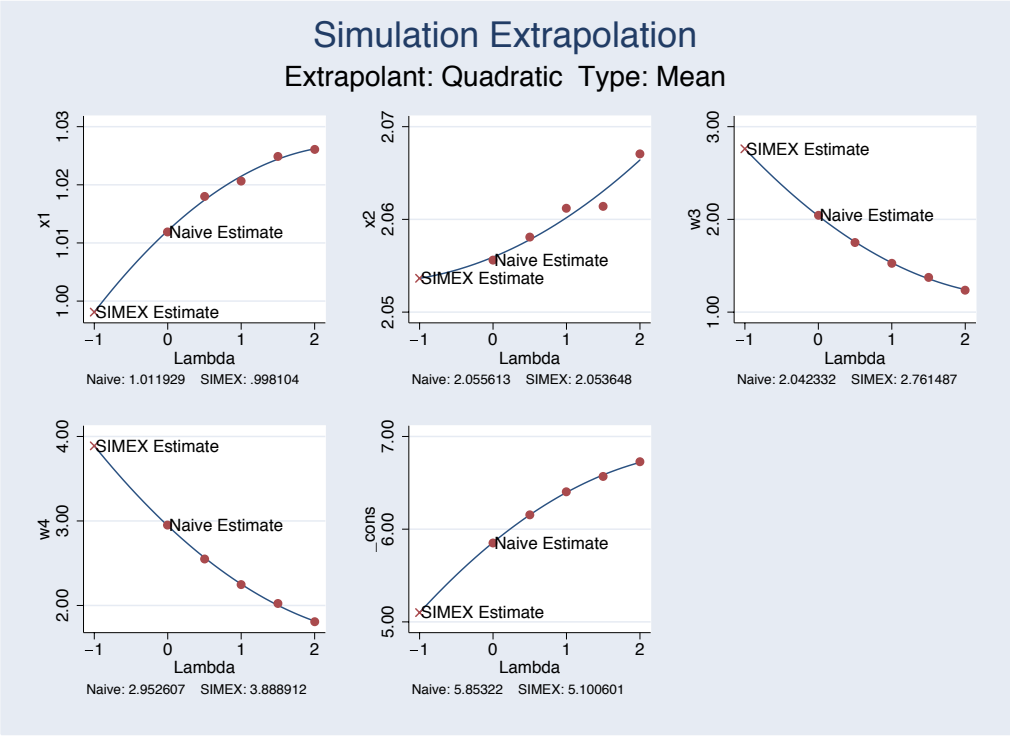

Figure 1: SIMEX results for a measurement error model. The model is for generated data following $y = x_1 + 2x_2 + 3x_3 + 4x_4 + 5$. In fitting the model, we have replicate error-prone measurements for the unobserved x_3 and x_4 variables. The graph illustrates the extrapolated point estimates for all covariates in the fitted model. The label w_3 is for the unobserved x_3 variable, and the label w_4 is for the unobserved x_4. The `SIMEX Estimate` label corresponds to -1 on the horizontal axis, while the `Naive Estimate` label corresponds to 0 on the horizontal axis. We have two error-prone replicate measures for each of the unobserved covariates in this fitted model. With multiple covariates, naive fitted covariates may be biased in either direction as illustrated.

4 NHANES example

Using the National Health and Examination Survey (NHANES) data, we investigate the presence of breast cancer `qbc` as a function of other covariates; `qage` is the age in years of the patient, `pir` is the poverty index ratio, `qbmi` is the body mass index, `alcohol` is indicator for whether the individual uses alcohol, `famhist` is an indicator of whether there is a family history of breast cancer, and `agemen` is the age at menarche. Two additional covariates, `qcalorie` and `qsatfat`, are the individual's recall measurements recorded for saturated caloric fat intake.

First, we fit a logistic regression, ignoring the measurement error. In this example, we simply include the replicate measures in `qcalorie` and `qsatfat` as two additional covariates and ignore the measurement error therein. Next, we fit a SIMEX model calculating bootstrap standard errors. In this model, we specify that `qcalorie` and `satfat` are replicate measures for an unknown covariate measure of the saturated fat intake of the individual. With 3,145 observations, 50 replications per value of $\boldsymbol{\theta}$, and 199 bootstrap replications, the model fitting takes some time to complete (approximately 6 minutes on our Linux system as listed in the output).

```
. local y "qbc"
. local z "qage pir qbmi alcohol famhist agemen"
. local x "qcalorie qsatfat"
. logit ‘y’ ‘z’ ‘x’, nolog
Logit estimates                                   Number of obs   =       3145
                                                  LR chi2(8)      =      29.11
                                                  Prob > chi2     =     0.0003
Log likelihood = -278.47466                       Pseudo R2       =     0.0497
```

qbc	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
qage	.0659472	.01951	3.38	0.001	.0277084	.1041861
pir	.123113	.0772283	1.59	0.111	-.0282516	.2744776
qbmi	-1.237347	2.499628	-0.50	0.621	-6.136529	3.661834
alcohol	.4374029	.287022	1.52	0.128	-.12515	.9999557
famhist	.6615556	.4433177	1.49	0.136	-.2073311	1.530442
agemen	-.1589137	.2700326	-0.59	0.556	-.6881679	.3703405
qcalorie	-1.351118	1.998842	-0.68	0.499	-5.268776	2.56654
qsatfat	-2.371214	2.0104	-1.18	0.238	-6.311526	1.569098
_cons	-5.649387	1.126337	-5.02	0.000	-7.856966	-3.441807

```
. simex (‘y’ = ‘z’) (w: ‘x’), fam(bin) brep(199) seed(12394)
Estimated time to perform bootstrap: 5.42 minutes.
Simulation extrapolation                          No. of obs         =       3145
                                                  Bootstraps reps    =        199
Residual df  =      3137                          Wald F(7,3137)     =       6.13
                                                  Prob > F           =     0.0000
Variance Function: V(u) = u(1-u)                     [Bernoulli]
Link Function    : g(u) = log(u/(1-u))               [Logit]
```

qbc	Coef.	Bootstrap Std. Err.	t	P>|t|	[95% Conf.	Interval]
qage	.0649982	.0156628	4.15	0.000	.0342878	.0957086
pir	.1298667	.0803848	1.62	0.106	-.0277454	.2874788
qbmi	-1.542769	2.651615	-0.58	0.561	-6.741844	3.656306
alcohol	.4492951	.275521	1.63	0.103	-.0909245	.9895147
famhist	.6751815	.4762267	1.42	0.156	-.2585659	1.608929
agemen	-.1473869	.2937225	-0.50	0.616	-.7232946	.4285209
w	-4.944333	2.472668	-2.00	0.046	-9.792544	-.0961208
_cons	-5.233584	1.073305	-4.88	0.000	-7.338036	-3.129132

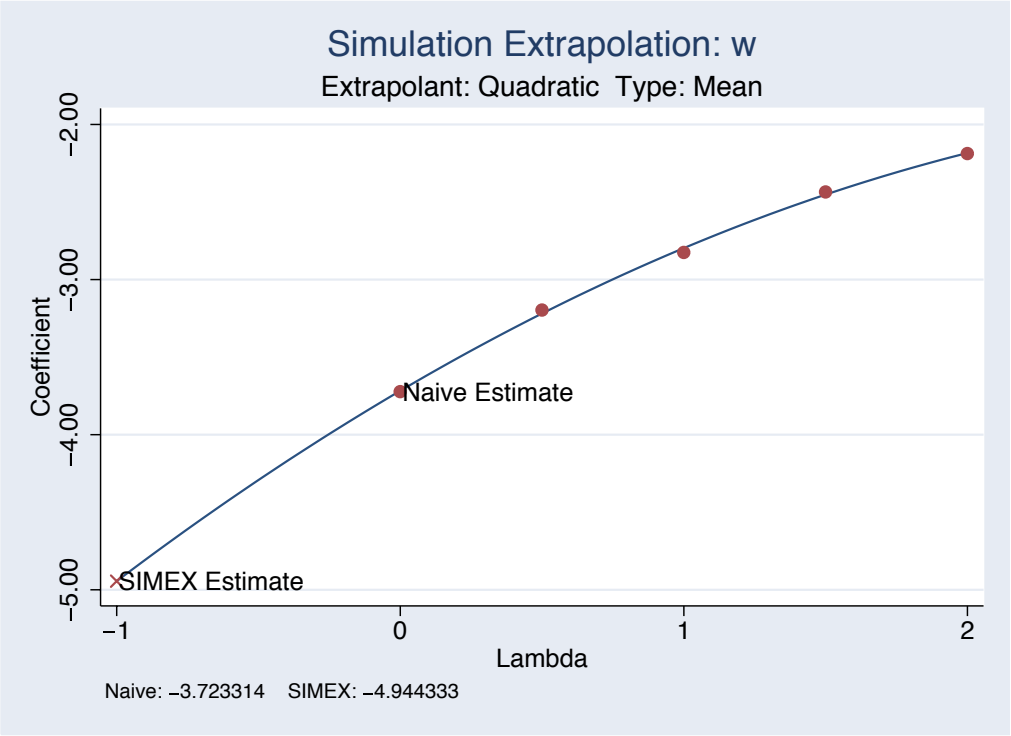

Figure 2: SIMEX results for a measurement error model. The plot is for the covariate measured with error and shows the mean coefficient estimates along with the extrapolated value for no measurement error. We do not have replicate measures in this dataset, so for illustration purposes, the model assumes that `qcalorie` (hundreds of calories) and `qsatfat` (hundreds of grams of fat) are replicate measures for the true saturated fat intake.

5 Formal Stata syntax

`simex` (*depvar* [*varlist*]) [(*depvar* [*varlist*]) ...(*depvar* [*varlist*])] [`if` *exp*] [`in` *range*] [`message`(#) `family`(*string*) `link`(*string*) `ltolerance`(#) `iterate`(#) `theta`(*matrixnum*) `srep`(#) `median` `linear` `rational` `nleps`(#) `nlrep`(#) `level`(#) `suuinit`(*matrixname*) `bstrap` `brep`(#) `btrim`(#) `seed`(#) `saving`(*string*) `replace`]

General options

`message`(#) specifies the desired (observed) level of printed messages of the plugin module. Users can specify this option to suppress or request warning and informational messages.

0) Display nothing, not even fatal error messages.

1) Display fatal error messages only.

2) Display warning messages (default).

3) Display informational messages.

4) Display more informational messages.

Note that the ado-code that handles the interface to the plugin may still print error messages regardless of the message level setting.

The message command can also be used to see intermediate details of the internal calculations of the code. These were used by the authors to debug the code. The notation and mnemonics used are not documented and may not correspond to anything in the printed documentation. Furthermore the numbers may be in a raw and unadjusted format that is difficult to interpret.

5, 6, 7) Display details with increasing verbosity.

Message levels are cumulative.

`family(`*string*`)` specifies the distribution of the dependent variable. The `gaussian` family is the default. The choices and valid family and link combinations are the same as for Stata's `glm` command.

`link(`*string*`)` specifies the link function; the default is the canonical link for the specified `family`. The choices and valid family and link combinations are the same as for Stata's `glm` command.

`ltolerance(#)` specifies the convergence criterion for the change in deviance between iterations. The default is `ltolerance(1e-6)`.

`iterate(#)` specifies the maximum number of iterations allowed in fitting the model. The default is `iterate(100)`. It is rare that one needs to increase this.

`theta(`*matrixnum*`)` specifies the $\boldsymbol{\theta}$ we will use for our simex. The default is `theta=(0, .5, 1, 1.5, 2)`.

`srep(#)` specifies the number of replications (simulations) for each theta. The default is `srep(50)`.

`median` specifies that the median extrapolant function should be used for extrapolating coefficient estimates. The default is the mean.

`linear` specifies that the linear quadratic extrapolant function should be used for extrapolating coefficient estimates. The default is quadratic regression.

`rational` specifies that the rational extrapolant function should be used for extrapolating coefficient estimates. The default is quadratic regression. When you use the `rational` extrapolant, the options `nlrep()` and `nleps()` are also available.

`nleps(#)` specifies the tolerance value to use in the optimization of the rational linear extrapolant. The default is `nleps(0.0001)`.

`nlrep(#)` specifies the number of replications to allow in the optimization of the rational linear extrapolant. The default is `nlrep(50)`.

Standard error options

`level(#)` specifies the confidence level, in percent, for confidence intervals of the coefficients.

`suuinit(`*matrixname*`)` specifies the measurement error covariance matrix. This is calculated from the replications in the measurement error variables if it is not specified.

Bootstrap options

`bstrap` specifies that bootstrap standard errors should be calculated. The bootstrap is internal to the code for the regression calibration command. The estimated time to perform the bootstrap will be displayed should the bootstrap require more than 30 seconds.

`brep(#)` specifies the number of bootstrap samples generated to calculate the bootstrap standard errors of the fitted coefficients. The default is `brep(199)`.

`btrim(#)` specifies the amount of trimming applied to the collection of bootstrap samples prior to calculation of the bootstrap standard errors. The default is `btrim(.02)`, meaning that 1% of the samples (rounded) will be trimmed at each end.

When the bootstrap is run with `mess(3)`, an informational message similar to this one will display

```
Average number of iterations per   GLM call: 3.0
Maximum number of iterations for a GLM call: 3
Minimum number of iterations for a GLM call: 3
Trimming total of 4 bootstrap replications (2.0%).
Maximum change in standard errors due to trimming: 2.4%
```

indicating that 4 samples (2 on each end) were trimmed and that this trimming resulted in a 2.4% change in magnitude of one of the standard errors. All other standard errors changed less than 2.4%. This simple diagnostic gives an indication of how trimming influenced the bootstrap standard errors.

`seed(#)` specifies a random number seed used in generating random samples for the bootstrap calculations. This option has no effect if bootstrapping is not specified. Its main purpose is to allow repeatability of bootstrap results. The default is `seed(0)`, which will seed the random number generator using the system clock.

`saving(`*string*`)` specifies the file to which the bootstrap results will be saved.

`replace` replaces the file specified in the `saving()` option (if that file already exists).

5.1 simexplot

`simexplot` [*varlist*]

The command `simexplot` will plot the effect of measurement error on the parameter estimate. It gives a visual representation on how the parameter estimates are derived by showing the extrapolation back to -1. Note that if the rational extrapolant was used, no extrapolant line is drawn.

The optional *varlist* is used to specify the parameters that are to be plotted. If no *varlist* is specified, all the `simex` parameters are plotted.

6 References

Carroll, R. J., H. Küchenhoff, F. Lombard, and L. A. Stefanski. 1996. Asymptotics for the SIMEX estimator in structural measurement error models. *Journal of the American Statistical Association* 91(433): 242–250.

Cook, J. and L. A. Stefanski. 1995. A simulation extrapolation method for parametric measurement error models. *Journal of the American Statistical Association* 89: 1314–1328.

Hardin, J. W. and R. J. Carroll. 2003. Measurement error, GLMs, and notational conventions. *Stata Journal* 3(4): 328–340.

Hardin, J. W., H. Schmiediche, and R. J. Carroll. 2003. The regression-calibration method for fitting generalized linear models with additive measurement error. *Stata Journal* 3(4): 360–371.

Stefanski, L. A. and J. Cook. 1995. Simulation extrapolation: The measurement error jackknife. *Journal of the American Statistical Association* 90(432): 1247–1256.

About the Authors

James W. Hardin (jhardin@gwm.sc.edu), is an Associate Research Professor, Department of Epidemiology and Biostatistics, and a Research Scientist, Center for Health Services and Policy Research, Arnold School of Public Health, Carolina Plaza Suite 1120, University of South Carolina, Columbia, SC 29208, USA.

Henrik Schmiediche (henrik@stat.tamu.edu), is a Senior Lecturer and Senior Systems Analyst, Department of Statistics, MS 3143, Texas A&M University, College Station, TX 77843-3143, USA.

Raymond J. Carroll (carroll@stat.tamu.edu) is a Distinguished Professor, Department of Statistics, MS 3143, Texas A&M University, College Station, TX 77843-3143, USA.

Research by StataCorp was supported by the National Institutes of Health (NIH) Small Business Innovation Research (SBIR) Grant (2R44RR12435-02).