Removing Bias — The SIMEX Procedure

Thomas Struppeck, FCAS, MAAA

Abstract: SIMEX (Simulation-Extrapolation) is a very general technique that helps to correct for bias in estimates caused by errors in measurements of predictors. The method is well established in statistical practice, but seems to not be as widely known in actuarial circles. Using ordinary least squares regression as an example, the method is illustrated using some simple R code. The reader is encouraged to run the code (which is highlighted in grey) in R themselves to get hands-on experience using the method.

Keywords: SIMEX, MCSIMEX, Naïve Estimate, Unbiasedness, Bias

INTRODUCTION

Linear regression and its generalizations are commonly used tools when analyzing data. One variable, the response, is explained in terms of the other variables (the explanatory variables or regressors). The coefficient on the regressor is called its slope, and the goal of regression is to obtain a good estimate of it. Classically, regressors are assumed to have been measured with no error. When this assumption is violated, slope estimates can be biased. This has been known for some time and various methods have been developed to estimate and remove this bias. One such method, SIMEX, was introduced by Cook and Stefanski in their 1994 paper [CS]. One disadvantage of SIMEX is that an *a priori* estimate of the variance of the measurement errors is needed, and this is not always available.

This paper first presents an example of how the bias in ordinary least squares (OLS) regression arises in practice. Then this example is used to introduce the SIMEX procedure in a simple case. The example illustrates how observed noisy data can be used to estimate the regression line that accurate data would give if it could be observed or captured or obtained.

Historically, actuaries chose exposure and classification variables that were "…objective and relatively easy and inexpensive to obtain and verify" [WM]. The variables that required more judgment were used for underwriting or tiering. With the advent of more sophisticated rating methodologies, this line has come blurred. Now a great many variables may be used in a pricing model — and it is possible that there are missing values. If these values are imputed or modeled externally, the imputed value can be thought of as an observation with measurement error. As an example, "rural miles driven" might be the actual variable of interest, but only "total miles driven" is available. If a study indicates that for a given territory, half of all miles driven are rural, we might use half of "total miles driven" as a proxy for "rural miles driven". This proxy behaves like a measurement with error, and regressions performed on the proxy will have slope parameters biased towards zero when compared to the true slope parameters for the underlying variable.

Methods like SIMEX can help remove bias from parameter estimates when using such data. For insurance data, the analyst has available the data that was recorded, not the actual data. The data that is recorded is called the observed data. The difference between the actual data and the observed data is called the noise. As we will see, the presence of noise may or may not introduce bias in the slope estimates depending on whether the noise is independent of the actual data or not. If we have an estimate of the variance of the noise, we can correct for it in our slope estimates.

Insurance data has often been rounded (say to the nearest thousand) or binned (say, 0-5, 6-10, 11-15) and while rounding or binning does introduce noise, we will see that it does not introduce bias. Other examples of insurance data such as miles driven (auto) or replacement cost for a building (homeowners) could be estimates (or proxies) and using these estimates (instead of the true values which are generally unavailable) can introduce bias. Estimates abound in reserving, consider for example case reserves based on legal opinions of potential jury awards; the difference between the actual award and the estimate could be viewed as noise.

ORDINARY LEAST SQUARES REGRESSION, THE GAUSS-MARKOV THEOREM, AND ATTENUATION

Ordinary Least Squares regression (OLS) is a commonly used tool that is very well behaved and very well understood, at least if the assumptions are all met. For instance, the Gauss-Markov Theorem says that in a regression if the errors have mean zero, have constant variance, and are uncorrelated, then the Best Linear Unbiased Estimator (more commonly, BLUE) of the slope coefficients is given by the OLS estimate. This says that (among other things) the slope estimates are <u>un</u>biased, but in practice the slope estimates often are biased — how can this be?

The answer can be seen by looking closely at the assumptions:

1) The errors have mean zero.
2) The errors have constant variance.
3) The various errors are uncorrelated.

It is common for all three of these to be satisfied. The problem is with the word "errors".

The model being fit is: $Y_i = \beta X_i + \alpha + \varepsilon_i$

The "errors" are the ε_i, which are assumed to have mean 0 (or else we could change the intercept, α), to have the same variance for all *i*, and to be independent (and hence, uncorrelated.) Notice that the error is entirely associated with the dependent variable, Y_i. What is meant by that is that the independent variable, X_i, is assumed to be measured perfectly accurately. In practice this may or may not be the case. Some regressors, such as policy limit or deductible may be known with certainty, but others, such as replacement cost, may have significant measurement error. If the error is independent of the actual value, then there will be attenuation (bias towards zero). Since the bias is towards zero, slope estimates will have their significance underestimated. That can cause otherwise significant regressors to appear insignificant. Surprisingly, if the error and true value are correlated, there may be no bias.

Let's look at an example:

```
set.seed (1001) # initialize the random number generator
x_actual <- runif(1000,10,20) # generate 1000 uniform (10,20) random variates
y_actual <- 3*x_actual + 2 # create the 1000 corresponding values of y
plot(x_actual,y_actual) # should all lie exactly on a line
abline(2,3,col="red")
(mod_a_a <- lm(y_actual~x_actual)) # coefficient estimates are exactly the correct (true) values
```

The reader is encouraged to run the sample blocks of code into R to follow along. Each block that involves generating random numbers will start with a call to set.seed[1].

We have generated 1000 uniform variates, X_actual, in the interval from (10,20). We then compute Y_actual = 3X_actual + 2. The points (X_actual,Y_actual) lie exactly along a line with slope 3. Indeed the regression shows the slope to be exactly 3. We will now add some noise to the Y_actual variables.

```
set.seed(1002)
# add some random noise to y_actual ... this is the y value that we observe
y_observed1 <- y_actual+rnorm(1000,0,5)
(mod_a_o1 <- lm(y_observed1~x_actual)) # the "extra" parentheses make R print the results
```

The slope coefficient on X_actual is estimated to be 2.911

```
summary(mod_a_o1)
```

Using the summary command, we can see the standard error of this estimate is 0.05237. We know that the true value of the slope is 3.000, which lies in the 95%-confidence interval.

Next, using the same Y-actual, we generate a different Y_observed (different noise) and estimate

[1] That function, set.seed, re-initializes the random number generator in R. This is done so that the random numbers that reader is using will match the ones used to make the examples. Different seeds are used each time so that no unintended correlations are introduced.

the new slope coefficient.

```
set.seed(1003) # cannot use the previous value or we will duplicate the old result!
# add some random noise to y_actual ... this is the y value that we observe
y_observed2 <- y_actual+rnorm(1000,0,5)
mod_a_o2 <- lm(y_observed2~x_actual)
summary(mod_a_o2)
```

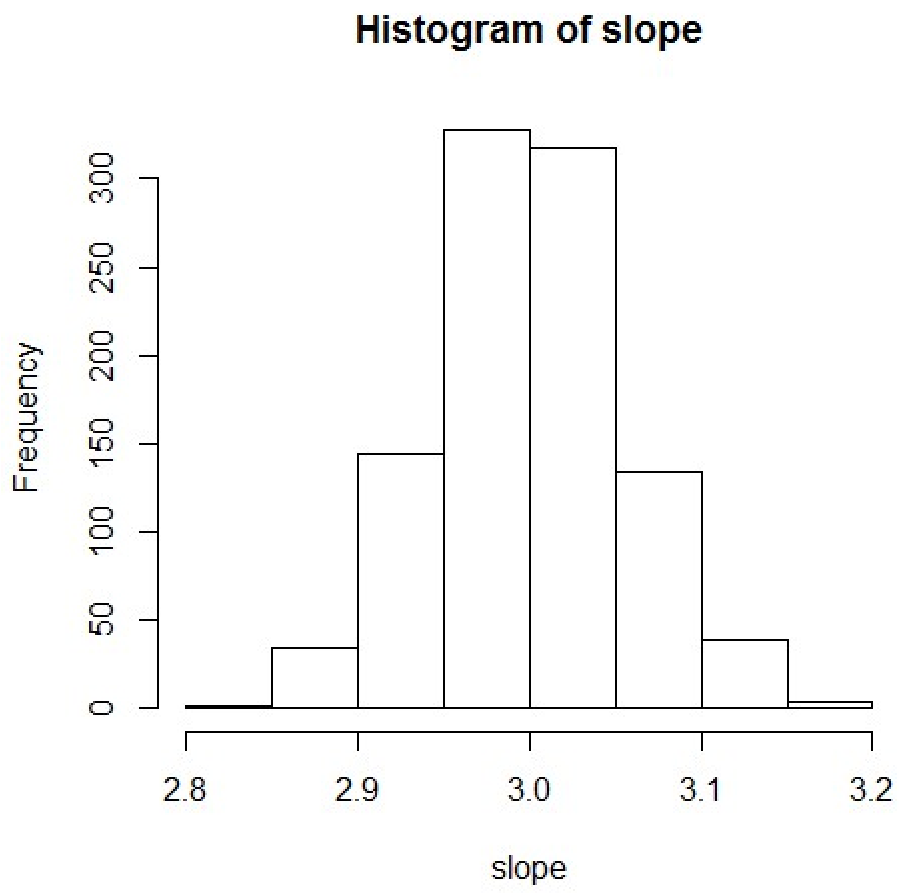
This time our slope estimate for X_actual was 2.985 which is closer to 3.000. To get a feeling for the sampling distribution of the slope coefficient, we will run 1000 regressions and record the slope estimates in a vector.

```
set.seed(1004)
slope <- rep(0,1000) # create a zero-filled vector of length 1000
for (i in 1:1000) {
    y_observed <- y_actual + rnorm(1000,0,5)
    mod_loop <- lm(y_observed~x_actual)
    slope[i]<- mod_loop$coeff[2]
}
summary(slope)
```

The average slope estimate is 3.000. That was luck, but most other seeds will also yield average estimates near the true value of 3.000. That is because the Gauss-Markov theorem tells us that ordinary least squares regression gives unbiased estimates.

Here is the histogram of the sampling distribution of the slope estimate:

```
hist(slope)
```



Now we will leave the noise added to Y fixed, and add some noise to X. By fixing the noise added to Y, we will be able to see how adding noise to X changes the result. (The variance of the noise added to X is twenty-five times smaller than that of the noise we added to Y, i.e. the standard deviation is one fifth as much.)

```
set.seed (1005)
y_observed3 <- y_actual+rnorm(1000,0,5) # We will leave this fixed from now on
x_observed1 <- x_actual+rnorm(1000,0,1)
(mod_o_o1 <- lm(y_observed3~x_observed1))
```

The slope estimate this time is way off, 2.681.

In this example, we have added normally distributed noise with mean 0 and standard deviation 1. The variance of the noise is what matters, not the exact distribution, so for example uniformly distributed noise on $[-\sqrt{3}, \sqrt{3}]$ (which also has standard deviation equal to one) should give similar results.

The estimate that we obtained is way outside the histogram from above. Perhaps we were unlucky? We will empirically estimate distribution of the slopes as we did before:

```
set.seed(1006)
for (i in 1:1000) {
    x_observed <- x_actual+rnorm(1000,0,1)
    mod_loop <- lm(y_observed3~x_observed)
    slope[i] <- mod_loop$coeff[2]
}
summary(slope)
```

The average estimate is 2.715, considerably smaller than the true value, 3.000. This bias toward zero is called attenuation bias. Our sample sizes (1,000) have been fairly large. Even larger sample sizes do not help in this situation. In other words, the slope estimates from OLS can be inconsistent[2] when there are errors in the regressors. Also, since the bias is towards zero, p-values and significance of regressors will be understated which, in a multivariable setting, could lead to a suboptimal choice of regressors.

The more noise that there is in the regressors, the worse that attenuation bias gets. What we would like to do is remove the noise from the data, but of course, that is impossible. We can however add more noise! Doing so increases the bias even more. We relate the amount of increase in the bias to the amount of additional (simulated) noise that we added. Once that relationship is established we use it to extrapolate back from the original noisy data to the ideal situation where there is no noise. That process, SIMulation followed by EXtrapolation, is the heart of the SIMEX algorithm. After a brief section on the theory, we will look at a hypothetical insurance data set and use SIMEX on it.

[2] An estimator is called inconsistent if it does not converge to the true value even as sample sizes go to infinity.

Observed values and Actual values

If we only have the noisy values, the variable X_actual can be thought of as a latent variable. The noisy values are the observed variable, so let's call these noisy values $X_{observed}$.

We can think of our model this way:

$$Y_{observed} = \beta X_{actual} + \alpha + \varepsilon_i$$

$$X_{observed} = X_{actual} + \delta_i$$

Conditional on the actual X value, the observed Y values have mean $\beta X_{actual} + \alpha$. There is noise ε_i added to Y, and we will assume that this noise has mean 0 and variance ${\sigma_Y}^2$.

We would like to estimate β and ${\sigma_Y}^2$, but we only observe the $Y_{observed}$ and the $X_{observed}$. While we do not actually directly observe the δ_i, often we will know something about the process that generates the δ_i. For example, if we are only know the year of an event which is equally likely to have occurred anytime during the year, we would probably use the mid-point of the year as the observed date; this is intended to make the error, as we have defined it, have mean 0. Or we may know from previous experience how far off, on average, the observed value is from the true value, *i.e.* the variance.

We will assume that δ_i has mean 0 and variance σ_δ^2. If we know its variance, and we know how it covaries with the actual X value, then we can, in the case of ordinary least-squares regression, actually compute the degree of attenuation.

Knowing how the errors covary with the measurements is not as unlikely as it might seem. The most common cases are perfect negative correlation and zero correlation. If the actual values are rounded or binned, the error and the actual value will be perfectly negatively correlated (for example, 4.2 gets always gets rounded to 4.0 and always produces an error of -0.2.) In other cases, assuming correlation zero is reasonable. As an example, consider the estimated replacement cost of a building. If we had ten appraisers evaluate the building, we would presumably get ten different estimates and these estimates would be centered around the actual value. We could estimate the variance of these estimates and use that as the variance of the error. The correlation between the estimate and the actual value would in this case be zero, assuming that a random appraiser from those ten did the measurement.

So, if repeated observations of a particular actual value always give the same observed value (*e.g.*

rounding) then the correlation will be negative 1. If on the other hand, repeated observations of a particular actual value give a range of observed values around the actual value (*e.g.* appraised values), then an assumption of correlation zero (uncorrelated noise) may be reasonable.

If we write $\sigma_{X\delta}$ for the covariance of X and δ, we have:

$$E(\hat{\beta}) = \beta \frac{\sigma_X^2 + \sigma_{X\delta}}{\sigma_X^2 + 2\sigma_{X\delta} + \sigma_\delta^2}$$ Equation 1 (General Case)

Equation 1 can be thought of as a credibility weighted average of the true value of the slope parameter, β, and zero.

$$E(\hat{\beta}) = \beta \left(\frac{\sigma_X^2 + \sigma_{X\delta}}{\sigma_X^2 + 2\sigma_{X\delta} + \sigma_\delta^2} \right) + 0 \left(\frac{\sigma_{X\delta} + \sigma_\delta^2}{\sigma_X^2 + 2\sigma_{X\delta} + \sigma_\delta^2} \right)$$ Equation 1A (Credibility Version)

The weight on beta is the sum of the variance[3] of X and its covariance with δ and the weight on zero is the sum of the variance of δ and its covariance with X. Note that when the correlation is negative 1 (such as when the actual values has been rounded) Equation 1 becomes:

$$E(\hat{\beta}) = \beta$$ Equation 2 (Special case: rounded data.)

So, in this case the estimator is unbiased. This equation is in fact the definition of unbiasedness.

When the covariance between the actual value of X and the error in measurement, $\sigma_{X\delta}$, is zero, we can see that the weights are just the respective variances. In this case, we have attenuation:

$$E(\hat{\beta}) = \beta \frac{\sigma_X^2}{\sigma_X^2 + \sigma_\delta^2}$$ Equation 3 (Special case: uncorrelated noise)

When the covariance is equal to negative one times the variance of δ, a special case that we return to briefly at the end, all of the credibility is assigned to the true value β, and there is no attenuation. That these two cases need to be distinguished was first observed in 1950 by Berkson [B].

[3] To be exact, this is n/(n-1) times the sample variance.

The literature on measurement error refers to the biased estimator in Equation 1 as the naïve estimate. In the case of ordinary least squares regression, since we know the exact extent of the bias, we can simply correct for it. (See, for example, [CRSC] Chapter 2. or [Fu])

An alternative method which is easily implemented and much more generally applicable is SIMEX, which we now illustrate with an insurance-based example using hypothetical data.

Example 2:

Our data set will consist of blood levels of a toxin and exposure information obtained from exposure badges worn by 500 workers covered by a workers' compensation policy, which covers occupational disease claims. It is believed that the exposure is linearly related to the blood level in each worker. The data is from an older model of badge which was known to be inaccurate. These are being replaced by newer badges which are much more accurate. For this example, we have assumed that measurement error on the older model badges was known to be one unit. We wish to estimate the slope coefficient.

Going forward we will have be using the more accurate exposure information from the new badges and wish to infer expected blood level. We want the best possible estimate for beta.

The 500 workers' actual exposures and observed exposures are generated in the code below:

```
set.seed(1007)
actual_vec<-runif(500,5,10)
obs_vec<-actual_vec+rnorm(500,0,1) # add individual variation to get the badge reading for each
worker
```

Now we will generate the (conditional) mean blood level for each worker. We selected an intercept of 10, but to make the exercise more exciting we will generate a random value for the true slope parameter (and reveal it at the end to see how we did).

```
set.seed(1008)
```

```
beta<-runif(1,2,4) # pick a uniformly distributed slope named beta in [2,4]
actual_bl<-beta*actual_vec + 10 # compute the actual mean for each individual
obs_bl<- actual_bl+rnorm(500,0,2) # add some random variation to the blood_level

naive_model<-lm(obs_bl~obs_vec)
(naive_slope<-naive_model$coeff[2])
```

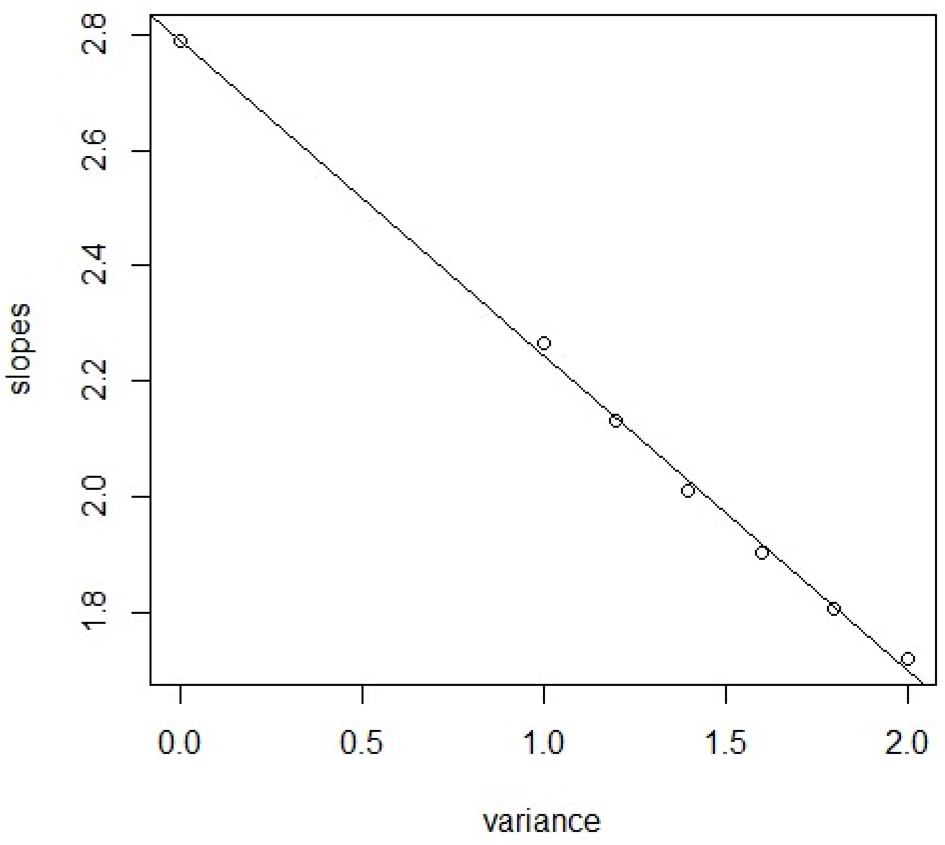
The slope estimate is 2.266. This is the naïve estimate. For this example we are given that the readings on the old badges vary from the mean with a standard deviation of 1 unit. We will add more noise to our observed data (increasing the variance to 1.2), compute the estimated slope and repeat many times (1000). This sample taken from the sampling distribution of $\hat{\beta}$ will by the law of large numbers have a mean very close to the expected value of $\hat{\beta}$. We then repeat that process for variances of 1.4, 1.6, 1.8, and 2.0.

```
set.seed(1009)
# SIMEX
slope<-numeric(1000) # allocate space for storing slopes
average_slope<-numeric(5) # space for averages
for (i in 1:5) { # 1:5 is 1,2,3,4,5
    extra_variance <- i/5 # generates 1/5, 2/5, 3/5, etc.
    for (j in 1:1000) {
            obs_plus_extra_noise <- obs_vec + rnorm(500,0,sqrt(extra_variance))
            slope[j] <- lm(obs_bl~obs_plus_extra_noise)$coef[2]
            }
    average_slope[i] <- mean(slope)
    }
variances<-1+0:5/5
simulated_slopes<-c(naive_slope,average_slope) #add naïve estimate to vector of slopes
```

```
(simex_mod<-lm(simulated_slopes~variances)) #find the slope and intercept of the best fit line
```

The naïve estimate was 2.266. We fit a linear model predicting the slope from the number of units of noise. The original data had one unit of noise … noiseless data would have zero units of noise; zero units of noise corresponds to the y-axis in the last regression above. In other words, the SIMEX estimate (of the slope) is the intercept from the linear model stored in sim_mod above. That is 2.791, significantly larger than the naïve estimate.

```
variance<-c(0,variances) #add ideal variance (0) to list of variances for plotting
slopes<-c(simex_mod$coeff[1],simulated_slopes) #add intercept from last regression for plotting
plot(variance,slopes)
abline(simex_mod$coeff[1],simex_mod$coeff[2]) #plot the regression line
```



The plot appears to show some concavity upwards. It can be shown that this is indeed the case and so SIMEX with linear extrapolation gives conservative estimates. Additionally, more sophisticated extrapolation techniques can obtain sharper and still conservative results. (See for example [CRSC], or the original paper by Cook and Stefanski [CS].) These more elaborate extrapolation techniques are implemented in the R package described briefly below in the section titled "The *simex* package in R". There a version of the above plot with a quadratic extrapolating function is shown.

One of the very nice by-products of SIMEX is this plot. The plot clearly illustrates what is happening, even more so if quadratic or other non-linear extrapolation method is used. This is often valuable in helping to explain how the method works and showing users or regulators that the method produces conservative estimates.

Time to reveal the true value of beta:

```
beta
```

So, the true value was 3.335.

Explaining the results to others.

Senior management or regulators, when presented reviewing a SIMEX analysis, may object to the addition of random noise into the data. It is important to quell such concerns. Here is a suggested approach:

The SIMEX procedure first computes the regular OLS estimate, the naïve estimate. This is done using the observed data with no additional noise.

We know that the naïve estimate will be biased if the errors are uncorrelated with the actual values. We want to correct for that bias and in order to estimate how much bias there is, we introduce additional noise. We then adjust the naïve estimate by the estimated bias adjustment.

It may be helpful to show the results of running the procedure a few times using different seeds in order to show that the choice of noise does not materially change the final estimates.

The *simex* package in R.

While working through the algorithm is a good way to learn and understand the method, there is no need to reinvent the wheel. A well-documented and versatile implementation of the algorithm is easily available in R. [R]

The *simex* package, written by Wolfgang Lederer and Helmut Küchenhoff [LK1], implements some more sophisticated extrapolation techniques, which improve the bias removal significantly. It also includes a variation called *mcsimex* that can handle discrete data with misclassification[4]. The package has the nice feature that the user can simply pass it an object of type *lm* or *glm* and it will perform the procedure on the corresponding model. More details on this package can be found in [LK2].

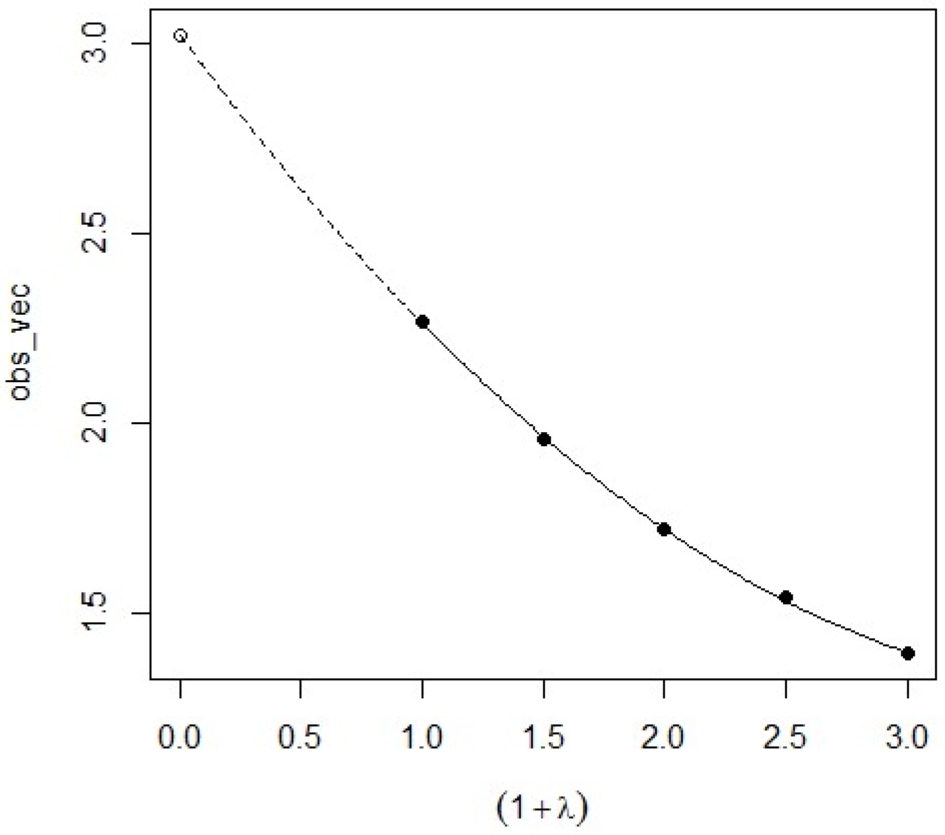
```
set.seed(1010)
library(simex) # this assumes that the simex package has already been loaded, if not, do so
# the call to simex below uses the default extrapolation method (quadratic), default number of
# iterations for each value of lambda (the amount of noise), with values of 0.5, 1.0, 1.5, and 2.0

# the given parameters are: the naïve model, the name of the variable with measurement error, the
amount of error, finally the switch asymptotic=FALSE suppresses asymptotic variance estimation.
simex_model<-simex(naive_model,"obs_vec",1,asymptotic=FALSE)
simex_model
```

The model output shows the estimated coefficient on obs_vec of 3.022 which is a much better estimate of the true value (3.335) than either SIMEX with linear extrapolation (2.791) or the naïve model (2.266) produced.

To obtain the corresponding plot, we need only pass the simex model to the generic plot function, again using defaults for most parameters.

```
plot(simex_model)
```

[4] The MCSIMEX methodology was used in an NIH-funded study of predictors of periodontal disease [SB], which potentially could be of interest to dental insurers.



What about rounded data?

If regressors have been rounded, the resulting estimates from OLS regression are not biased! This may seem surprising since rounded data appears to look much like data with noise uniformly distributed on the interval (-1/2,1/2) added. The difference is subtle. This is the case where the noise and the actual value are correlated. An example may make this clearer. If the actual data was 9.4, we would always round that down to the observed rounded value of 9.0. In other words, knowledge of the true value, tells you what the observed value will be — they are correlated. Contrast this with adding random noise where the observed value could be either bigger or smaller for a fixed actual value. For a further discussion of this point see for example Faraway [F] or the original 1950 paper by Berkson [B].

CONCLUSION

SIMEX is a general method that allows one to estimate the attenuation bias present in a model. This paper illustrates the technique in the setting of ordinary least squares regression. The requirements to use SIMEX are an understanding of the size of the measurement error in the regressors and an understanding of how these are correlated with the actual values. Unless the errors are from rounding or binning, usually it will be reasonable to assume that the correlation is zero and that attenuation bias is occurring.

By adding even more noise to the regressors and observing the additional attenuation, it is possible to estimate the amount of attenuation that is already present in the data.

One important by-product of the analysis is a graph that is easy to explain and understand. That graph allows the method to be quickly explained in a non-technical way.

An implementation of SIMEX (and a related method, MCSIMEX) is available in the *simex* package in R. This publicly available implementation has a more sophisticated extrapolation procedure which removes even more of the attenuation from the slope estimates.

REFERENCES

[1.] [B] Berkson, J. (1950). Are there two regressions? Journal of the American Statistical Association 45, 165-180.

[2.] [CRSC] Carroll, R.J, Ruppert, D., Stefanski, L.A., & Crainiceanu, C.M. (2006). Measurement Error in Nonlinear Models. New York: Chapman & Hall/CRC.

[3.] [CS] Cook, J. and Stefanski, L. (1994). Simulation-extrapolation estimation in parametric measurement error models. Journal of the American Statistical Association 89, 1314-1328.

[4.] [F] Faraway, J.F. (2005). Linear Models with R. New York: Chapman & Hall/CRC.

[5.] [Fu] Fuller, W. (1987). Measurement Error Models. New York: Wiley.

[6.] [LK1] Wolfgang Lederer and Helmut Küchenhoff (2013). simex: SIMEX- and MCSIMEX-Algorithm for measurement error models. R package version 1.5. https://CRAN.R-project.org/package=simex.

[7.] [LK2] Wolfgang Lederer and Helmut Küchenhoff (2006) R News, Vol. 6/4, October 2006.

[8.] [R] R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[9.] [SB] Slate, E.H. and Bandyopadhyay, D. (2009) Stat Med 28(28), 3523-38.

[10.] [WM] Werner, G. and Modlin, C. (2016). Basic Ratemaking. CAS.