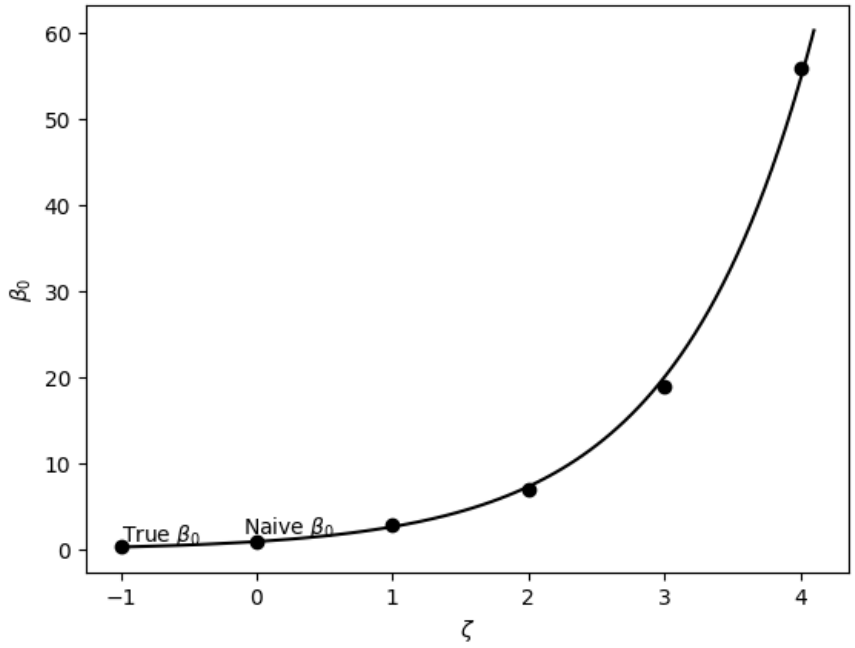
The complexity and performance of the SIMEX algorithm

IJsbrand Meeter

June 30, 2024

Bachelorscriptie Wiskunde en Informatica

Begeleiding: dr. Eni Musta, dr. Victoria Degeler



Instituut voor Informatica
Korteweg-de Vries Instituut voor Wiskunde
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Universiteit van Amsterdam

Abstract

In this thesis the influence of various parameters of SIMEX is investigated. SIMEX is an algorithm that corrects for measurement error in data. The parameters that are varied are the number of data points(N), the amount of iterations per data point (M) and the size of the dataset(n). A time and space complexity is investigated for each of these parameters as well as a performance analysis. The time and space complexity is both done analytically and numerical. The space and time complexity is only investigated analytically for the number of explanatory variables(p). In all cases the results of the numerical and analytical approach of the complexity are identical. The time complexity is linear for all parameters, while the space complexity is linear for all except the number of iterations, for which it is constant. The time complexity of SIMEX is given by $\mathcal{O}(NM(n + C_{n,p} + Np)$ where $C_{n,p}$ denotes the time required for fitting a model, for instance a linear or logistic regression model, which is dependent on n and p. The space complexity of SIMEX is given by $\mathcal{O}(Np+n+C_{n,p})$. The performance experiments indicate that SIMEX reduces the bias of an estimator but increases the variance of these estimators. The experiment conducted to investigate the impact of varying certain parameters on the performance do only show a difference when increasing n.

Titel: The complexity and performance
of the SIMEX algorithm

Auteur: IJsbrand Meeter, ysbrandm@xs4all.nl, 13880624
Begeleiding: dr. Eni Musta, dr. Victoria Degeler
Tweede beoordelaars: dr. Tim van Erven, dr. Chrysa Papagianni

Einddatum: June 30, 2024

Instituut voor Informatica
Universiteit van Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.ivi.uva.nl`

Korteweg-de Vries Instituut voor Wiskunde
Universiteit van Amsterdam
Science Park 904, 1098 XH Amsterdam
`http://www.kdvi.uva.nl`

Contents

1 Introduction

Data plays starts playing an increasingly pivotal role in the modern world. Data is used for training large AI models, perform research, improve daily life and so much more. The correctness of collected data is often taken for granted. Many assume that as long that data is gathered correctly that mathematical models can be used as intended. This actually is often not the case, measurement error can not, even when the data is collected correctly, be removed. Many mathematical models assume perfect data and can provide wrong results when fed imperfect data. So there is a clear need for ways to correct for measurement error.
This thesis focuses on the SIMEX algorithm, one such way to correct for measurement error. This algorithm is first mentioned in [9] in 1994 as a way to correct for measurement error. A number of other atricles have been published. These were a great help in writing this thesis. For instance the work of Grace [2], Caroll [3] and Wallace [4].
At the basis SIMEX works by adding measurement error with increasing variance to the dataset and trying to find a pattern in estimators to the in this way created datasets. By then using this pattern to extrapolate to an estimator that corresponds to a dataset without error. A more detailed description is given later in the thesis. A particular case in which SIMEX can account for measurement errors is in estimating coefficients of a logistic regression function. In SIMEX there are two parameters that need to be chosen. The first is the number of times a new dataset with measurement error with more variance is created and the second parameter is how many times this procedure is repeated to get a reliable average. This thesis tries to give grounds for a correct choice of both of these parameters. At the moment choices for parameters within SIMEX are often not substantiated. These parameters are a trade of between computational cost and accuracy of the algorithm. By taking a look into how these two components relate to each other a more informed choice can be made for these parameters. To investigate this relation both performance and complexity are considered separate.
This thesis is separated in multiple sections. It starts out by painting a broad picture about what measurement error is and what kind of measurement error model exists. In addition a quick refresher on linear regression is given. After which the effect of measurement error on linear regression is investigated to show the direct influence of measurement error on regression models. A second regression model is handled in which measurement error plays a more prominent role. In chapter two SIMEX is introduced as a way to correct for measurement error. Since SIMEX assumes the variance of the measurement error to be known a quick overview on ways to estimate this variance is also discussed. Lastly, in chapter three the complexity and performance is considered for all parameters separately.
This thesis has no significant ethical aspects. The thesis describes a mathematical pro-

cedure to correct for measurement error. If used correctly, it can not be used to commit any ethical wrongdoing. However wrong use of statistics can always be used for deliberate manipulation or pushing a own agenda.
All code used is available on github[8].

2 Measurement error

2.1 Basic concepts

Measurement error, sometimes abbreviated as ME, denotes the difference between the true and measured value of a variable X. For instance X can be the true blood pressure of a patient. Measurements of the blood pressure can differ from the true blood pressure due to inaccuracy of the measuring device. The difference between the measured blood pressure and the true blood pressure can be a random error or a systematic error. Systematic error would for instance occur if the sphygmomanometer (a device that measures blood pressure) is calibrated wrong, so it consistently adds a fixed value to the actual blood pressure. While in the case of random error this added value is not fixed. This thesis will only focus on random error. Since systematic error can be easily accounted for. Let U denote this random error. There exists two different models for additive error, the first of which is

$$\text{(the classical model) } \tilde{X} = X + U \tag{2.1}$$

Where X denotes the true value, $\tilde{X}$ the measured value. U is assumed to have mean zero and variance σ_u^2. In the classical model we assume our measurement error U to be independent of X.

Example 2.1.1 Consider X to be the true blood pressure of the patient and $\tilde{X}$ to be the measured blood pressure of the patient. Due to inaccuracies in the measuring device X differs from $\tilde{X}$ with an additive error U. The amount of random error is independent of the true blood pressure. Thus the measurement error follows the classical model.

The second model this measurement error can follow is,

$$\text{(the Berkson model) } \tilde{X} = X + U, \tag{2.2}$$

where again $\mathbb{E}[U] = 0$, and $Var(U) = \sigma_u^2$ but now U is assumed to be independent of $\tilde{X}$. Which is the difference between these two models.

Example 2.1.2 Consider X to be the true amount of pesticide absorbed by a plant. The exact amount of pesticide absorbed by the plant is unknown. The amount of pesticide used is however measured, denote this as $\tilde{X}$. Since the plant potentially has not absorbed all the pesticide this measured quantity of pesticide differs from the true amount of pesticide in the plant. Let U be this difference. So the situation can be written as $\tilde{X} = X + U$. In this case U is independent of $\tilde{X}$.

Which one of the models is applicable can often only be determined by reasoning. Nevertheless it is important to classify which model the error follows. As will be asserted later.

2.2 The effect of ME in regression functions

In the next section the influence of measurement error is investigated on two different type of regression functions, and if measurement error needs to be accounted for.

2.2.1 linear regression

The earlier introduced measurement error can cause several problems in the correctness of parameter estimators. The following part will investigate the consequences of measurement error on the standard linear regression model.
Consider the simple linear regression model,

$$Y = \beta_0 + \beta_1 X + \epsilon$$

with $\epsilon \sim N(0, \sigma_\epsilon)$, and ϵ being independent of all explanatory variables. When working with n measurements of X, Y, X and ϵ are vectors in $\mathbb{R}^n$. It will be clear from context how many measurements are considered. Expanding this definition to multiple explanatory variables, gives a model of the following form,

$$Y = \beta_0 + \beta_1 X^1 + \beta_2 X^2 + \ldots + \beta_m X^m + \epsilon.$$

This model can be written as,

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} = \beta_0 + \beta_1 \begin{pmatrix} X_1^1 \\ \vdots \\ X_n^1 \end{pmatrix} + \ldots + \beta_m \begin{pmatrix} X_1^m \\ \vdots \\ X_n^m \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}.$$

For simplicity this can be written in matrix notation,

$$Y = \mathbf{X}\beta + \epsilon, \tag{2.3}$$

where,

$$\mathbf{X} = \begin{bmatrix} 1 & X_1^1 & \ldots & X_1^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_n^1 & \ldots & X_n^m \end{bmatrix}, \text{ and } \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_m \end{pmatrix}.$$

This β described the relationship between X^i and Y. For instance a zero coefficient indicates the two to be independent. Based on n i.i.d observations one is interested in estimating these coefficients. When the coefficients are correctly estimated one can predict Y values using only newly observed explanatory variables. Getting an incorrect estimator for β makes the regression model useless. Estimating these coefficients is sometimes referred to as fitting the model. In the next section some basic properties of general estimators are recalled. After which specific estimators of linear regression are introduced.

2.2.2 Basic properties for estimators

Definition 2.2.1 An estimator T_n of a certain parameter $g(\theta)$ is considered unbiased if $\mathbb{E}[T_n] = g(\theta)$. The bias of an estimator is given by $\mathbb{E}[T_n] - g(\theta)$.

Note that unbiased estimators can also be defined as estimators for which the bias equals zero, a semi trivial note but a useful tool to proof unbiased.
Definition 2.2.2 A estimator T_n of a certain parameters $g(\theta)$ is considered consistent if T_n converges in probability to $g(\theta)$, denoted as $T \xrightarrow{\mathbb{P}} g(\theta)$. That is

$$\forall \epsilon > 0, \mathbb{P}[|T_n - g(\theta)| > \epsilon] \xrightarrow{n\to\infty} 0.$$

Bias and consistent both can be seen as a way to determine the correctness of an estimator. Both are not a golden standard. For instance an unbiased estimator can have a high variance resulting in useless estimator.
Returning to the linear regression model and its estimators,
Definition 2.2.3 The least squares estimator, often abbreviated as LSE, of β is the $\hat{\beta}$ that minimizes $\sum_{i=1}^{n}(Y - \mathbf{X}\hat{\beta})^2$.

Proposition 2.2.4 The least squares estimator for β as mentioned above is given by,

$$\hat{\beta}_n = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y},$$

and is an unbiased estimator.

Proof. This formula is the result of a standard minimizing procedure, and a standard result. This proof will focus on the fact that it is an unbiased estimator, to show the contrast when adding ME later. For a detailed proof on why this is formula for β_n is as it is see this book[**?**] To show the estimator being unbiased consider the expected value of $\hat{\beta}_n$, rewriting results in

$$\begin{aligned}
\mathbb{E}[\hat{\beta}_n] &= \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] \\
&= \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\beta + \epsilon)] \\
&= \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\beta + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon] \\
&= \mathbb{E}[\beta] + \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon].
\end{aligned}$$

Since X and ϵ are independent this can be split into two, resulting in,

$$\mathbb{E}[\hat{\beta}_n] = \mathbb{E}[\beta] + \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]\mathbb{E}[\epsilon] = \mathbb{E}[\beta] + \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]0 = \mathbb{E}[\beta].$$

Since β is a constant $\mathbb{E}[\beta] = \beta$, and thus $\mathbb{E}[\hat{\beta}_n] = \beta$ and so $\hat{\beta}_n$ is an unbiased estimator. □

2.2.3 Influence of ME in linear regression

For simplicity consider the case with only one explanatory variable to observe some basic properties about measurement error. Assume the explanatory variable not to be measured exactly but with additive measurement error, i.e $\tilde{X} = X + U$. So instead of X only $\tilde{X}$ is observed. In this section both the standard and Berkson model are used for ME to investigate the different influences they have on estimators. This would result in the following linear regression model,

$$\begin{aligned} Y &= \beta_0 + \beta_1 X + \epsilon \\ &= \beta_0 + \beta_1(\tilde{X} - U) + \epsilon \\ &= \tilde{\mathbf{X}}\beta - \beta_1 U + \epsilon, \end{aligned}$$

with

$$\tilde{\mathbf{X}} = \begin{bmatrix} 1 & \tilde{X}_1 \\ \vdots & \vdots \\ 1 & \tilde{X}_n \end{bmatrix}.$$

The naive way would be to just ignore the measurement error and use the earlier introduced naif estimator, $\hat{\beta}_n = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{Y}$ for β (hence the name). This naive estimator can result in an estimators with bias.

$$\begin{aligned} \mathbb{E}[\hat{\beta}_n - \beta] &= \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T Y - \beta] \\ &= \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T(\tilde{\mathbf{X}}\beta - \beta_1 U + \epsilon) - \beta] \\ &= \mathbb{E}[\beta - (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U + (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon - \beta] \\ &= \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U] - \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon] \end{aligned}$$

Using the same argument as seen in the previous proposition this simplifies down to:

$$\mathbb{E}[\hat{\beta}_n - \beta] = \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U] \tag{2.4}$$

This bias itself still could possibly evaluate to zero, making the estimator not necessarily biased or unbiased. Further below this will be investigated more precisely for the two models for measurement error.

Proposition 2.2.5 In the simple linear regression model

$$Y = \beta_0 + \beta_1 X + \epsilon$$

where X is measured with additive error, following the Berkson model, the naive LSE $\hat{\beta}_n$, is an unbiased estimator.

Proof. As seen in (2.4) the bias is given by:

$$\mathbb{E}[\hat{\beta}_n - \beta] = \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U]$$

The Berkson model assumes U to be independent of $\tilde{X}$ and to have mean zero, and thus:

$$\mathbb{E}[\hat{\beta}_n - \beta] = \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1\mathbf{U}] = \mathbb{E}[\hat{\beta}_n - \beta] = \mathbb{E}[(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1]\mathbb{E}[U] = 0.$$

This concludes that,

$$\mathbb{E}[\hat{\beta}_n - \beta] = 0,$$

and thus that $\hat{\beta}_n$ is a unbiased estimator. □

As mentioned before unbiased of an estimator is not a golden ticket for a "perfect" estimator. Making an estimator unbiased can result in for instance a large variance.
Proposition 2.2.6 In the simple linear regression model where X is measured with additive error, following the classical model, the LSE $\hat{\beta}_n$ is an inconsistent estimator.

Proof. Recall the naive estimator $\hat{\beta}_n$,

$$\begin{aligned}
\hat{\beta}_n &= (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T Y \\
&= (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T(\tilde{\mathbf{X}}\beta - \beta_1 U + \epsilon) \\
&= \beta - (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U + (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon \\
&= \beta - (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U + (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon
\end{aligned}$$

consider both terms individually

$$(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U = (\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\frac{1}{n}\tilde{\mathbf{X}}^T\beta_1 U,$$

and

$$(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon = (\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\frac{1}{n}\tilde{\mathbf{X}}^T\epsilon$$

First calculate the $(\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}$ term explicitly as,

$$\left(\frac{1}{n}\begin{bmatrix}1 & \dots & 1\\ \tilde{X}_1 & \dots & \tilde{X}_n\end{bmatrix}\begin{bmatrix}1 & \tilde{X}_1\\ \vdots & \vdots\\ 1 & \tilde{X}_n\end{bmatrix}\right)^{-1} = \begin{bmatrix}1 & \frac{1}{n}\sum_{i=1}^n \tilde{X}_i\\ \frac{1}{n}\sum_{i=1}^n \tilde{X}_i & \frac{1}{n}\sum_{i=1}^n \tilde{X}_i^2\end{bmatrix}^{-1}$$

The first term now equals:

$$\begin{bmatrix} 1 & \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i \\ \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i & \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i^2 \end{bmatrix}^{-1} \frac{1}{n}\beta_1 \begin{bmatrix} 1 & \tilde{X}_1 \\ \vdots & \vdots \\ 1 & \tilde{X}_n \end{bmatrix}^T \begin{pmatrix} U_1 \\ \vdots \\ U_n \end{pmatrix} =$$

$$\begin{bmatrix} 1 & \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i \\ \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i & \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i^2 \end{bmatrix}^{-1} \frac{1}{n}\beta_1 \begin{bmatrix} 1 & \dots & 1 \\ \tilde{X}_1 & \dots & \tilde{X}_n \end{bmatrix} \begin{pmatrix} U_1 \\ \vdots \\ U_n \end{pmatrix} =$$

$$\begin{bmatrix} 1 & \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i \\ \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i & \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i^2 \end{bmatrix}^{-1} \beta_1 \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n}U_i \\ \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_i U_i \end{pmatrix} =$$

$$\begin{bmatrix} 1 & \overline{\tilde{X}_n} \\ \overline{\tilde{X}_n} & \overline{\tilde{X}_n^2} \end{bmatrix}^{-1} \beta_1 \begin{pmatrix} \overline{U_n} \\ \overline{\tilde{X}U_n} \end{pmatrix}.$$

The law of large numbers states that the the sample mean converges to the true mean in chance. Since taking the inverse is a continuous mapping, thus the continuous mapping theorem states that the limit can be taken before the inverse, thus this converges in chance to

$$\begin{bmatrix} 1 & \mathbb{E}[\tilde{X}] \\ \mathbb{E}[\tilde{X}] & \mathbb{E}[\tilde{X}^2] \end{bmatrix}^{-1} \beta_1 \begin{pmatrix} \mathbb{E}[U] \\ \mathbb{E}[\tilde{X}U] \end{pmatrix} = \begin{bmatrix} 1 & \mathbb{E}[\tilde{X}] \\ \mathbb{E}[\tilde{X}] & \mathbb{E}[\tilde{X}^2] \end{bmatrix}^{-1} \beta_1 \begin{pmatrix} 0 \\ \mathbb{E}[\tilde{X}U] \end{pmatrix},$$

since $\mathbb{E}[U] = 0$ by assumption. Unfortunately $\mathbb{E}[\tilde{X}U] \neq 0$ since U and $\tilde{X}$ are not independent.

$$\frac{1}{Var(\tilde{X})} \begin{bmatrix} \mathbb{E}[\tilde{X}^2] & -\mathbb{E}[\tilde{X}] \\ -\mathbb{E}[\tilde{X}] & 1 \end{bmatrix} \begin{pmatrix} 0 \\ \beta_1\mathbb{E}[\tilde{X}U] \end{pmatrix} = \frac{1}{Var(\tilde{X})} \begin{pmatrix} -\beta_1\mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}U] \\ \beta_1\mathbb{E}[\tilde{X}U] \end{pmatrix}.$$

Using the same arguments,

$$\begin{aligned} (\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\frac{1}{n}\tilde{\mathbf{X}}^T\epsilon &= \begin{bmatrix} 1 & \overline{\tilde{X}} \\ \overline{\tilde{X}} & \overline{X^2} \end{bmatrix}^{-1} \begin{pmatrix} \overline{\epsilon} \\ \overline{\tilde{X}\epsilon} \end{pmatrix} \\ &\xrightarrow{\mathbb{P}} \begin{bmatrix} 1 & \mathbb{E}[\tilde{X}] \\ \mathbb{E}[\tilde{X}] & \mathbb{E}[\tilde{X}^2] \end{bmatrix}^{-1} \begin{pmatrix} \mathbb{E}[\epsilon] \\ \mathbb{E}[\tilde{X}\epsilon] \end{pmatrix} \\ &= \begin{bmatrix} 1 & \mathbb{E}[\tilde{X}] \\ \mathbb{E}[\tilde{X}] & \mathbb{E}[\tilde{X}^2] \end{bmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0, \end{aligned}$$

since $\tilde{X}$ and ϵ are independent and because $\mathbb{E}[\epsilon] = 0$. Thus this concludes that,

$$\hat{\beta}_n \xrightarrow{\mathbb{P}} \beta - \frac{1}{Var(\tilde{X})} \begin{pmatrix} -\beta_1\mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}U] \\ \beta_1\mathbb{E}[\tilde{X}U] \end{pmatrix}$$

Showing that the naive parameters are inconsistent. □

A possible surprising result from theorem 1.2 is the fact that both $\hat{\beta_{0,n}}$ and $\hat{\beta_{1,n}}$ are inconsistent, even though $\hat{\beta_{0,n}}$ has initially nothing to do with the measurement error. To explore this further consider a model with, for simplicity, only two explanatory variables. Denote X^2 as a perfect measured variable and $\tilde{X^1}$ as a variable measured with error. The model then would have the following form:

$$\begin{aligned} Y &= \beta_0 + \beta_1 X^1 + \beta_2 X^2 + \epsilon \\ &= \beta_0 + \beta_1 \tilde{X^1} - \beta_1 U + \beta_2 X^2 + \epsilon \\ &= \tilde{\mathbf{X}}\beta - \beta_1 U + \epsilon \end{aligned}$$

Proposition 2.2.7

In the multiple linear regression model,

$$Y = \beta_0 + \beta_1 X^1 + \beta_2 X^2 + \epsilon$$

where X^1 is measured with additive error following the Berkson model, and X^2 is measured exact, the LSE $\hat{\beta}_n \xrightarrow{\mathbb{P}} \beta - \alpha\beta_1 \begin{pmatrix} -\mathbb{E}[X^1]Var(X^2)\mathbb{E}[X^1U] \\ \mathbb{E}[X^1U]Var(X^2) \\ -\mathbb{E}[X^1U]Cov(X^1, X^2) \end{pmatrix}$,
with $\alpha := \frac{1}{Var(X^2)Var(X^1) - Cov(X^1, X^2)^2}$.

Proof. For the sake of clarity let X^1 and X^2 be denoted by respectively X and Z. Similarly, let $\tilde{X}$ denote $\tilde{X^1}$.
Consider the just described model, for which the naive parameters are as previously seen,

$$\begin{aligned} \hat{\beta_n} &= (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T Y \\ &= (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T(\tilde{\mathbf{X}}\beta - \beta_1 U + \epsilon) \\ &= \beta - (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U + (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon \\ &= \beta - (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U + (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon. \end{aligned}$$

as seen before $(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\epsilon \xrightarrow{\mathbb{P}} 0$. Rewriting the other term as seen before into $(\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\frac{1}{n}\tilde{\mathbf{X}}^T\beta_1\mathbf{U}$,

and researching the convergence of this term separated,

$$\left(\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\right)^{-1} = \left(\frac{1}{n}\begin{bmatrix}1 & 1 & \dots & 1\\ \tilde{X}_1 & \tilde{X}_2 & \dots & \tilde{X}_n\\ Z_1 & Z_2 & \dots & Z_n\end{bmatrix}\begin{bmatrix}1 & \tilde{X}_1 & Z_1\\ 1 & \tilde{X}_2 & Z_2\\ \vdots & & \vdots\\ 1 & \tilde{X}_n & Z_n\end{bmatrix}\right)^{-1}$$
$$= \frac{1}{n}\begin{bmatrix}n & \sum_1^n \tilde{X}_i & \sum_1^n Z_i\\ \sum_1^n \tilde{X}_i & \sum_1^n \tilde{X}_i^2 & \sum_1^n \tilde{X}_i Z_i\\ \sum_1^n Z_i & \sum_1^n \tilde{X}_i Z_i & \sum_1^n Z_i^2\end{bmatrix}^{-1}$$
$$\xrightarrow{\mathbb{P}} \begin{bmatrix}1 & \mathbb{E}[\tilde{X}] & \mathbb{E}[Z]\\ \mathbb{E}[\tilde{X}] & \mathbb{E}[\tilde{X}^2] & \mathbb{E}[\tilde{X}Z]\\ \mathbb{E}[Z] & \mathbb{E}[\tilde{X}Z] & \mathbb{E}[Z^2]\end{bmatrix}^{-1}$$

Where the last limit is due to the fact that taking the inverse is a continuous operator, thus taking the limit as a result of the law of large numbers can be applied first, because of the continuous mapping theorem. This inverse can be calculated using the method of co factors.

$$\begin{bmatrix}1 & \mathbb{E}[\tilde{X}] & \mathbb{E}[Z]\\ \mathbb{E}[\tilde{X}] & \mathbb{E}[\tilde{X}^2] & \mathbb{E}[\tilde{X}Z]\\ \mathbb{E}[Z] & \mathbb{E}[\tilde{X}Z] & \mathbb{E}[Z^2]\end{bmatrix}^{-1}$$

$\xrightarrow{\text{Calculate co factors}}$

$$\begin{bmatrix}\mathbb{E}[\tilde{X}^2]\mathbb{E}[Z^2] - \mathbb{E}[\tilde{X}Z]^2 & \mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2\\ \mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & \mathbb{E}[Z^2] - \mathbb{E}[Z]\mathbb{E}[Z] & \mathbb{E}[\tilde{X}Z] - \mathbb{E}[\tilde{X}]\mathbb{E}[Z]\\ \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2 & \mathbb{E}[\tilde{X}Z] - \mathbb{E}[\tilde{X}]\mathbb{E}[Z] & \mathbb{E}[\tilde{X}^2] - \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}]\end{bmatrix}$$

$\xrightarrow{\text{Simplifying}}$

$$\begin{bmatrix}\mathbb{E}[\tilde{X}^2]\mathbb{E}[Z^2] - \mathbb{E}[\tilde{X}Z]^2 & -\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] + \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2\\ -\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] + \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & Var(Z) & -\mathbb{E}[\tilde{X}Z] + \mathbb{E}[\tilde{X}]\mathbb{E}[Z]\\ \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2 & -\mathbb{E}[\tilde{X}Z] + \mathbb{E}[\tilde{X}]\mathbb{E}[Z] & Var(\tilde{X})\end{bmatrix}$$

$\xrightarrow{\text{Sign changes}}$

$$\begin{bmatrix} \mathbb{E}[\tilde{X}^2]\mathbb{E}[Z^2] - \mathbb{E}[\tilde{X}Z]^2 & -\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] + \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2 \\ -\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] + \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & Var(Z) & -\mathbb{E}[\tilde{X}Z] + \mathbb{E}[\tilde{X}]\mathbb{E}[Z] \\ \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2 & -\mathbb{E}[\tilde{X}Z] + \mathbb{E}[\tilde{X}]\mathbb{E}[Z] & Var(\tilde{X}) \end{bmatrix}$$

$\xrightarrow{\text{divide by the Determinant}}$

$$\alpha \begin{bmatrix} \mathbb{E}[\tilde{X}^2]\mathbb{E}[Z^2] - \mathbb{E}[\tilde{X}Z]^2 & -\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] + \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2 \\ -\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] + \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z] & Var(Z) & -\mathbb{E}[\tilde{X}Z] + \mathbb{E}[\tilde{X}]\mathbb{E}[Z] \\ \mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}]^2 & -\mathbb{E}[\tilde{X}Z] + \mathbb{E}[\tilde{X}]\mathbb{E}[Z] & Var(\tilde{X}) \end{bmatrix},$$

where $\alpha := \frac{1}{det}$, thus,

$$\left(\frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\right)^{-1} \xrightarrow{\mathbb{P}} \alpha \begin{bmatrix} Var(\tilde{X}Z) & -\mathbb{E}[\tilde{X}]Var(Z) & \mathbb{E}[Z]Var(\tilde{X}) \\ -\mathbb{E}[\tilde{X}]Var(Z) & Var(Z) & -Cov(\tilde{X}, Z) \\ \mathbb{E}[Z]Var(\tilde{X}) & -Cov(\tilde{X}, Z) & Var(\tilde{X}) \end{bmatrix}.$$

The second term converges in the following way,

$$\frac{1}{n}\beta_1 \begin{bmatrix} 1 & \tilde{X}_1 & Z_1 \\ \vdots & \vdots & \vdots \\ 1 & \tilde{X}_n & Z_n \end{bmatrix}^T \begin{pmatrix} U_1 \\ \vdots \\ U_n \end{pmatrix} = \beta_1 \frac{1}{n} \begin{pmatrix} \sum_{i=1}^n U_i \\ \sum_{i=1}^n \tilde{X}_i U_i \\ \sum_{i=1}^n Z_i U_i \end{pmatrix} \xrightarrow{\mathbb{P}} \beta_1 \begin{pmatrix} 0 \\ \mathbb{E}[\tilde{X}U] \\ \mathbb{E}[ZU] \end{pmatrix}$$

Since both limits are known, the product of these sequences is the product of its limits, resulting in,

$$\begin{aligned} (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\beta_1 U &\xrightarrow{\mathbb{P}} \beta_1 \alpha \begin{bmatrix} Var(\tilde{X}Z) & -\mathbb{E}[\tilde{X}]Var(Z) & \mathbb{E}[Z]Var(\tilde{X}) \\ -\mathbb{E}[\tilde{X}]Var(Z) & Var(Z) & -Cov(\tilde{X}, Z) \\ \mathbb{E}[Z]Var(\tilde{X}) & -Cov(\tilde{X}, Z) & Var(\tilde{X}) \end{bmatrix} \begin{pmatrix} 0 \\ \mathbb{E}[\tilde{X}U] \\ \mathbb{E}[ZU] \end{pmatrix} \\ &= \alpha\beta_1 \begin{pmatrix} -\mathbb{E}[\tilde{X}]Var(Z)\mathbb{E}[\tilde{X}U] + \mathbb{E}[ZU]\mathbb{E}[Z]Var(\tilde{X}) \\ \mathbb{E}[\tilde{X}U]Var(Z) - \mathbb{E}[ZU]Cov(\tilde{X}, Z) \\ -\mathbb{E}[\tilde{X}U]Cov(\tilde{X}, Z) + \mathbb{E}[ZU]Var(\tilde{X}) \end{pmatrix}. \end{aligned}$$

Since Z and U are independent, and $\mathbb{E}[U] = 0$ this reduces to,

$$\alpha\beta_1 \begin{pmatrix} -\mathbb{E}[\tilde{X}]Var(Z)\mathbb{E}[\tilde{X}U] \\ \mathbb{E}[\tilde{X}U]Var(Z) \\ -\mathbb{E}[\tilde{X}U]Cov(\tilde{X}, Z) \end{pmatrix}.$$

Calculating α explicitly results in,

$$\begin{aligned}
\alpha &:= \frac{1}{\mathbb{E}[\tilde{X}^2]\mathbb{E}[Z^2] - \mathbb{E}[\tilde{X}Z]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[\tilde{X}](\mathbb{E}[\tilde{X}]\mathbb{E}[Z^2] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}Z]) + \mathbb{E}[Z](\mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z] - \mathbb{E}[Z]\mathbb{E}[\tilde{X}^2])} \\
&= \frac{1}{\mathbb{E}[\tilde{X}^2]\mathbb{E}[Z^2] - \mathbb{E}[\tilde{X}Z]^2 - \mathbb{E}[\tilde{X}]^2\mathbb{E}[Z^2] - \mathbb{E}[Z]^2\mathbb{E}[\tilde{X}^2] + 2\mathbb{E}[Z]\mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z]} \\
&= \frac{1}{(\mathbb{E}[\tilde{X}^2] - \mathbb{E}[\tilde{X}]^2)(\mathbb{E}[Z^2] - \mathbb{E}[Z]^2) - \mathbb{E}[\tilde{X}]^2\mathbb{E}[Z]^2 - \mathbb{E}[\tilde{X}Z]^2 + 2\mathbb{E}[Z]\mathbb{E}[\tilde{X}]\mathbb{E}[\tilde{X}Z]} \\
&= \frac{1}{Var(\tilde{X})Var(Z) - (\mathbb{E}[\tilde{X}Z] - \mathbb{E}[\tilde{X}]\mathbb{E}[Z])^2} \\
&= \frac{1}{Var(Z)Var(\tilde{X}) - Cov(\tilde{X}, Z)^2}.
\end{aligned}$$

Finally, combining the results from above and substituting X and Z back with respectively X^1 and X^2 result is,

$$\hat{\beta_n} \xrightarrow{\mathbb{P}} \beta - \alpha\beta_1 \begin{pmatrix} -\mathbb{E}[\tilde{X^1}]Var(X^2)\mathbb{E}[\tilde{X^1}U] \\ \mathbb{E}[\tilde{X^1}U]Var(X^2) \\ -\mathbb{E}[\tilde{X^1}U]Cov(\tilde{X^1}, X^2) \end{pmatrix}$$

□

Corollary 2.2.8 If X^1 and X^2 are independent the naif estimator for β_2 converges to the true parameter.

Proof. If X^1 and X^2 are independent then so are $\tilde{X^1}$ and X^2 so the covariance of the two is zero, resulting in $\hat{\beta_2} = \beta_2$. □

This corollary points out the dependence of the bias of β_2 on the covariance, meaning that the amount of dependence plays a major role in the amount of bias of β_2.
It seems that β_0 and β_1 both rely on X^2. But when taking α into account this dependency is lost. So there is not a condition regarding X^2 for which β_0 and β_1 are unbiased.
As seen in the previous section the influence of ME on parameters in linear regression can calculated. This makes explicit ways to correct for the ME possible. This is not the case for all regression models. In the next section another regression model is briefly explored, which needs a more complex procedure to correct for ME.

2.2.4 Logistic Model

Definition 2.2.9 A general regression function is a function $f_\beta(x)$ depending on a certain parameter β such that,

$$f_\beta(x) = \mathbb{E}[Y|X = x].$$

In simpel linear regression this is function is $f_\beta(x) = \beta_0 + \beta_1 X$ Sometimes one is besides the expected value of Y also interested in the underlying distribution. For simplicity consider a Y variable that can only take binary variables. For instance a simple yes or no question, like "is a person running?" can be written as binary data. Y would be a binary dataset where $Y_i = 1$ if person i is running and $Y_i = 0$ if not. X describes for instance the hearth rate of the person, where X_i is the heart rate of person i. One would be interest in the chance that certain person is running given their heart rate. Or in other words the conditional probability,

$$\mathbb{P}(Y = 1|X = x) = 1 - \mathbb{P}(Y = 0|X = x).$$

A model often used for these kind of problems is the logistic regression model,

$$\varphi_\beta(x) := \frac{e^{\beta x}}{e^{\beta x} + 1} = \mathbb{P}(Y = 1|X = x), \tag{2.5}$$

MLE estimators can be used for β.
Consider X to be not measured exactly but with some measurement error, i.e,

$$\tilde{X} = X + U$$

with $\mathbb{E}[U] = 0$. A general regression model then changes in the following way,

$$f_\beta(\tilde{X}) = \mathbb{E}[Y|\tilde{X}] = \mathbb{E}[\mathbb{E}[Y|X, \tilde{X}]|\tilde{X}]$$

due to the Law of total expectation. When assuming a logistic regression model the inner expectation can be written as,

$$\mathbb{E}[Y|X, \tilde{X}] = \mathbb{E}[Y|X] = \varphi_\beta(x) = \frac{e^{\beta X}}{1 + e^{\beta X}}.$$

It holds since if X is known $\tilde{X}$ holds no further information about Y. So Y is independent of $\tilde{X}$ if X is known. One is interested in the regression function,

$$f_\beta(\tilde{X}) = \mathbb{E}\left[\frac{e^{\beta X}}{1 + e^{\beta X}}|\tilde{X}\right]$$

This is not necessarily again a logistic regression model

$$f_\beta(\tilde{X}) \neq \frac{e^{\hat{\beta}\tilde{X}}}{1 + e^{\hat{\beta}\tilde{X}}}.$$

Working with a different regression function causes different results from measurement error. While in a linear regression function only the estimators of the parameters become biased when measurement error is added, the logistic regression function becomes a misspecified model for the data with ME. It is possible that the model still approximates the true model, but even then the estimators would still be inconsistent[11].

2.2.5 Unnecessary Error correction

A correction for measurement error is not always necessary. There are possible cases in which the measurement error is not causing any problems. Measurement error only causes problems when one is interested in estimating the true parameters, since these can not simply be estimated from data with error without adding bias. However the relation between $\tilde{X}$ and Y can be estimated correctly. So the predictive nature of regression models are still viable. When fitting $\tilde{X}$ and Y the relation between X and Y stays unknown. This however is not a problem when wanting to predict Y for a new data points $\tilde{x_i}$. Nevertheless if a data point $\tilde{x_i}'$ has a different type of ME than $\tilde{X}$ then Y can't be predicted accurately with the model fitted to $\tilde{X}$.

3 Correcting for ME

Correcting for measurement error can be done in several ways, this part will focus on one correction method in particular that suites additive measurement error well. In addition it is easy to implement and works for all kind of the regression models.

3.1 Simulation-extrapolation

SIMEX is short for Simulation Extrapolation, which got its name from the two separate steps of the method. It is a simulation-based method for reducing bias. By simulating data with an increasing measurement error and calculating the bias, a trend in the bias versus the variance in the measurement error can be established which can be used to extrapolate back to a case without measurement error. The underlying idea of SIMEX is the fact that the bias of a estimator as a result of measurement error can be approximated via simulations. For simplicity consider the simple linear regression where X is measured with additive measurement error denoted as U following the classical model. Assume $\mathbb{E}[U] = 0$, and $\text{Var}(U) = \sigma_u^2$. The SIMEX procedure works as follows,

1. Create computer generated measurement error with variance $\zeta_m \sigma_u^2$ independent of the other variables. This ME is often drawn from a normal distribution, but can be drawn from a other appropriate distribution, depending on the underlying distribution of the original ME. By then adding this created ME to $\tilde{X}$ a new dataset with measurement error with variance $(1 + \zeta_m)\sigma_u^2$ is created.

2. Calculate the naive estimator $\hat{\beta}_{\zeta,n}$ with respect to the newly created dataset.

3. Repeat the previous two steps a large number of times and for different values of ζ_m and plot the different values of ζ_m against the calculated estimators. After which the average over the generated β is taken.

4. Fit a model on these created points, which are of the form $(\zeta_1, \beta_{\zeta_1,n}, \vdots, \zeta_N, \beta_{\zeta_N,n})$, and extrapolate to the case without measurement error ($\zeta = -1$).

SIMEX can be visualized in the following manner,

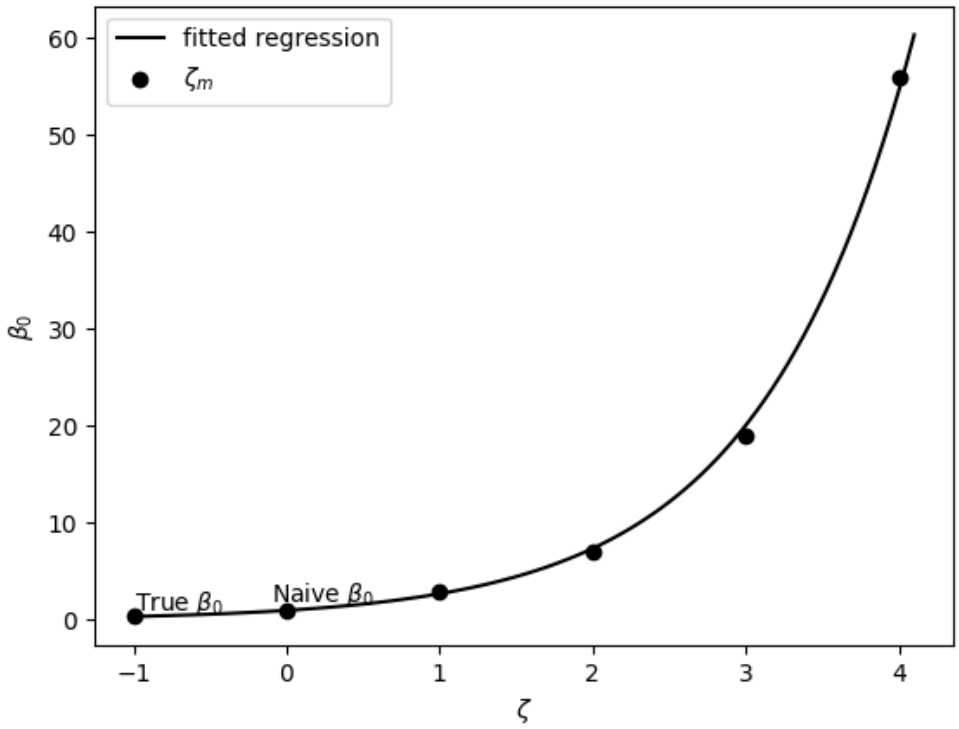


Figure 3.1: β_0 plotted for different number choices of ζ_m, where ζ_m corresponds to a ME variance of $(1+\zeta_m)\sigma_U$ where σ_U denotes the original variance of the ME.

For an analysis over why SIMEX works consider the additive measurement error model of the following form,

$$\tilde{X} = X + U,$$

with $\mathbb{E}[U] = 0$ and $\text{Var}(U) = \sigma_u$. This can be written as,

$$\tilde{X} = X + \sigma_u U,$$

with $\mathbb{E}[U] = 0$ and $\text{Var}(U) = 1$. Exploring the mean and standard deviation of $\tilde{X}$ gives more information about the variable,

$$\begin{aligned}\mathbb{E}[\tilde{X}] &= \mathbb{E}[X] + \mathbb{E}[\sigma_u U] = \mathbb{E}[X] \\ \text{Var}(\tilde{X}) &= \text{Var}(X + \sigma_u U) = \text{Var}(X) + \text{Var}(\sigma_u U) = \text{Var}(X) + \sigma_u^2.\end{aligned}$$

Creating new data during the iteration yields in a variables with the following properties,

$$\tilde{X}_{\zeta_m} = \tilde{X} + \sqrt{\zeta_m}\sigma_u U_{\zeta_m}$$

with again $\mathbb{E}[U] = 0$ and $\text{Var}(U) = 1$. To explore this variable further we see that:

$$\mathbb{E}[\tilde{X}_{\zeta_m}] = \mathbb{E}[\tilde{X}] + \mathbb{E}[\sqrt{\zeta_m}\sigma_u U] = \mathbb{E}[X]$$
$$\text{Var}(\tilde{X}_{\zeta_m}) = \text{Var}(\tilde{X}) + \text{Var}(\sqrt{\zeta_m}\sigma_u U) = \text{Var}(\tilde{X}) + \zeta_m\sigma_u^2 = \text{Var}(X) + (1+\zeta_m)\sigma_u^2.$$

As seen before $\zeta_m = -1$ results in $\text{Var}(\tilde{X}_{\zeta_m}) = \text{Var}(X)$, which results in $\text{Var}(U) = 0$. The attentive reader probably wonders why would one go through the SIMEX trouble, and not calculate the true parameter directly with $\zeta_m = -1$. This is due to the how new data for ζ_m is created in SIMEX. Since the data needs to be the dataset with only more additive error. SIMEX creates this new data by creating a random data set with variance $\sqrt{\zeta_m}\sigma_U$ and an expectation of zero. Then adding this data to the already existing dataset creating more error. This explain why $\zeta_m = -1$ can not be used directly, since there would need to be a possibility to create a dataset with a negative variance, and this is not possible.
When adding more ME and estimating the parameters to this newly created data a lot of variance gets added. Since all these processes are random processes. To counteract this added variance this step is repeated a large number of times, the amount of repetitions needs to be chosen when performing SIMEX.
The second parameter that needs to be chosen is the number of data points that is created. As expected a large number of data points results in a more optimal fit of a regression function to these data points, but again adds more variance in the process. The trend in this data points is not known in advance. When the simulating part of SIMEX is finished an appropriate regression function needs to be chosen. Often a polynomial function or a rational function is appropriate.
In a next chapter the effect of the number of data points and the number of repetitions on the performance of SIMEX is further investigated.

3.2 Measurement error variance

The assumption that SIMEX is based on is the fact that the variance of the error is known. This is not always known, there are several ways to estimate the variance of the measure error, in the following section two will be explained.

3.2.1 Using gold standard

When working with measurement error there is always exists a true value X. In the case that $\tilde{X}$ and X are both known it is really simple to calculate the variance of the measurement error. Consider $(\tilde{X}_1, \cdots, \tilde{X}_n)^T$ and $(X_1, \cdots, X_n)^T$ for which the following relation holds,

$$\tilde{X}_i = X_i + U_i$$

A dataset of only the measurement error can be created, namely $(U_1, \cdots, U_m)^T$ with,

$$U_i = \tilde{X}_i - X_i,$$

from which the variance can be calculated. A larger sample size of U leads to a more accurate calculation of its variance. In the perfect world one would have a X_i for every measured $\tilde{X}_i$. In this case there is no need for any measurement error correction, since one could use the true data X, instead of trying to work with $\tilde{X}$. Measuring a true X_i for every $\tilde{X}_i$ is often not desirable, since it is often time consuming, but it would lead to a perfect estimated regression model. In practice the true X is only known for a fraction of the dataset. When using only this small part of the dataset to fit a regression model would lead to an under fitted regression model, to still get a well fitted regression model the entire dataset $\tilde{X}$ can be used to fit the model.

Example 3.2.1 Consider a group of patients for which the blood pressure needs to be measured for a research. It is possible to measure their blood pressure at home with an easy to use device but which is known to make error in its measurement. There is also the possibility of a patients to come to the hospital where there blood pressure can be measured precise. The researcher decide to let half of the patients come to the hospital to measure their blood pressure with the accurate device and the inaccurate device as well. While the other half only measures there blood pressure at home. Using the above mentioned method the variance of the error of the at home device can be measured, after which measurement error correction can be applied, making their research more powerful since they have more measurements while keeping the costs low, because only the half of the patients needed to come to the hospital.

3.2.2 Replicated measurements of the same variable

The next method makes use of multiple measurements of the same variable. Consider $(\tilde{X}_1^{(1)}, \cdots, \tilde{X}_n^{(1)})^T$ and $(\tilde{X}_1^{(2)}, \cdots, \tilde{X}_n^{(2)})^T$ which are two measurements of $(X_1, \cdots, X_n)^T$ with error. Where $(X_1, \cdots, X_n)$ is unknown. A new dataset $(U_1, \cdots, U_n)^T$ can be created by,

$$U_i := \tilde{X}_n^{(1)} - \tilde{X}_n^{(2)} = (X_i + U_i^{(1)}) - (X_i + U_i^{(2)}) = U_i^{(1)} - U_i^{(2)}.$$

This newly created dataset has variance $2\sigma_U^2$, since

$$\text{Var}(U_i) = \text{Var}(U_i^{(1)} - U_i^{(2)}) = \text{Var}(U_i^{(1)}) + \text{Var}(U_i^{(2)}) = 2\sigma_U^2.$$

From this dataset the variance can be calculated in the standard way, from which the wanted variance of the measurement error can easily be calculated.

Example 3.2.2 Consider again the same setting as in example 1.1. Instead of the researchers opting to make use of the device in the hospital, they let all the patients measure their blood pressure twice at home. Resulting in two slightly different measurements of their blood pressure. By subtracting the two measurement of each patient, and calculating the variance of these subtraction over all the patients the researchers can calculate the measurement error, by then correcting for measurement error when using the data, it can be used without ever needing to visit the hospital.

3.2.3 Other methods

There exists a method to estimate the measurement error variance without the need of extra data but it uses strong assumptions about the model.[10] So is not applicable to all data. If the variance of the measurement error can not be estimated it is also possible to do a sensitivity analysis and perform SIMEX for a different range of measurement error variances to see how the results of the estimators depend on the differences in variance.

4 Complexity vs performance

When working with an algorithm one always needs to weigh up the costs of performing the algorithm against the possible gain of the algorithm. When using SIMEX there are several input choices that need to be made, which all effect the performance of the algorithm and the amount of time and memory it uses. This consideration will be further investigated in the following section.

4.1 SIMEX in code

In section 3.1 SIMEX is mathematically described. The following inputs are needed for SIMEX. First of all SIMEX needs the variance of the measurement error, denoted as σ_U, it needs a dataset of the response variable and a dataset for each explanatory variable (with measurement error), the sizes of these dataset is noted with n, and the number of explanatory variables is denoted with p. The user needs to make some choices. They need to choose the number of data points on which the final extrapolation will be performed, denoted as N, and the number of simulations for each of these data points, denoted as M. The following pseudo code describes SIMEX when X_i and Y have a logistic relation,

```
SIMEX(Explanatory variable with ME := X,
      Explanatory variable without ME:=[X_2,...,X_p],
      Response variable without ME := Y,
      variance of ME :=σ_U,
      number of datapoints := N,
      number of simulations := M):

     parameters = []*(p+1)
     ζ = []*N #list to store the data points
     for i in range(N):
        β = []
        for _ in range(M):
            ε = [ε_i | ε_i ~ N(0,(1+ζ)σ_U)]
            X' = X + ε
            β_0, β_1,..., β_{p+1} = fit_logistic_function(Y, X'+[X_2,...,X_p])
            β[j] += β_j
        ζ[i] = β/M
    for i in range(p+1):
        a_0, a_1,...,a_d = LSE([0, 1,...,length(ζ)], ζ)
        parameters[i]= a_0 + a_1 * (-1) + a_2 * (-1)^2 + ... + a_d * (-1)^d
    return parameters
```

Understanding the complexity of SIMEX is not possible without understanding the pseudo code.
The new data with more measurement error is created from the already existing data in line 13 and 14. For each data point a random variable gets added, which is drawn from a normal distribution with mean zero and variance $(1 + \zeta)\sigma_U$.
When the new data is created the parameters are estimated again in line 15. According to the relation between the explanatory variables and the response variable the estimation is done in different ways. For linear regression a least squares estimator is used, while for logistic regression a maximum likelihood estimator is used. The maximum likelihood estimator for logistic regression is out of scope for this bachelor thesis.
In line 19 a model is fitted to the created data points, several models are used, but they mostly can be fitted using a least square estimator. In the case of logistic regression d is chosen to be 2, i.e a 2 degree polynomial is fitted, but this can change for different regression forms of the original X and Y.
When fitting a polynomial model the least square estimator, as seen before, can be obtained using the following matrix operation.

$$(X^T X)^{-1} X y$$

where X is a matrix of dimension $(d + 1) \times N$. Where d is the degree of the polynomial that is getting fitted.
Both the last extrapolation of line 20 and taking an average over the beta's in line 17 are trivial operations. Taking an average is a matter of just dividing by M, and extrapolation is done by substituting -1 in the fitted model.

4.2 Complexity preliminaries

When working with algorithms it is interesting to see how the amount of resources it needs varies when changing a certain input. This is called the complexity of an algorithm. When working with complexity a difference is made between space and time complexity. In computer science the complexity of an algorithm is given in big $\mathcal{O}$ notation. Big $\mathcal{O}$ describes an upper bound for the amount of time or spaces needed as a function on the size of the input.

Definition 4.2.1 A function f is said to be big $\mathcal{O}$ of g if there exists a α and x_0 such that

$$|f(x)| \leq \alpha g(x) \text{ for all } x \geq x_0,$$

notated as

$$f(x) = \mathcal{O}(g(x)) \text{ as } \lim_{x \to \infty} f(x).$$

Big $\mathcal{O}$ notation only describes the asymptotic behaviour of function, i.e if it behaves linear or quadratic, and not the exact size of the function.

Example 4.2.2 Consider the following simple function for calculating the factorial of a given number.

```
def factorial(n):
    res = 0
    for i in range(n):
        res *= i
    return res

```

This function executes the line,

```
    res *= i,
```

n times. Multiplying two numbers takes the computer some seconds, lets say α second. So the entire *for* loop takes $n \cdot \alpha$ seconds. A function of the amount of time would be $f(n) = \alpha n$, so its time complexity would be $\mathcal{O}(n)$.
The only time new memory is allocated in the function is at the start when *res* is defined. Denote the number of time this takes as α. This is never repeated, thus a function for the amount of space used would be $f(n) = \alpha$, so the space complexity would be $\mathcal{O}(1)$.

4.2.1 Complexity of SIMEX

In the following section the memory complexity is investigated in the case of data following a logistic regression model. For these experiments the test data is created in the following way. First a dataset X is created of size n drawn from a standard normal distribution. For each $x_i \in X$ a $p_i \in [0, 1]$ is created using the logistic function 2.5, creating a data set of chances. Then Y is filled with the result of a Bernoulli experiment with chance p_i, i.e $y_i \sim Bern(p_i)$.. Unless otherwise specified the following parameters are used, $n = 100,\ N = 5,\ M = 300,\ p = 2$ and $\sigma_u = 0.3$. Each experiment is ran ten times to minimize variation in the results due to randomness.

4.2.2 Number of simulations

In this section the influence of the number of iterations, denoted as M, on the time and space complexity is investigated. As well as on the performance.
Using the aforementioned pseudo code it is clear that the number of simulations only has effect in the number of times the inner *for* loop gets executed. The code in the *for* loop creates new data, but since the previous data is not necessary anymore, it can be overwritten resulting in no new memory needing to be allocated. Meaning that in the entire inner *for* loop no new memory is allocated. Which results in a constant space complexity, $\mathcal{O}(1)$.
When looking at the time complexity of this same *for* loop the executed block of code all does not depend on the number of simulations, since the number of simulations only effects the amount of times the *for* loop is executed. Note the time needed for the block of code α. The *for* loop would take αM time units, which results in a linear time complexity, i.e $\mathcal{O}$(M).

An experiment is ran. SIMEX is run with the default values, as previously mentioned. The number of simulations per ζ is varied from 50 up to 1000, with 30 data points equally distributed in this range. To make sure that a previously executed program does not influence the results by for instance changing the state of the cache memory a clear list of three data points are used which are not taken into account in the results.
When running this experiment the first time a weird result appeared. The memory consumption was linear, but as earlier described it should have been constant. To solve fit the logistic function as in line 14, the lbfgs method (Limited-memory Broyden-Fletcher-Goldfarb-Shanno method) is used without penalty. A penalized version is used when working with a large number of covariates, since this is not the setting that the experiments works with a no penalty version is used. Unfortunately this version has an acknowledge memory bug. [7]Creating an bug less own version of an solver is out of scope for this thesis, so a different kind of solver, which does need a penalty, is used for all memory and time experiments that will follow, The penalty is manually chosen to be very small, in the order of 1^{-10}, to simulate a none penalty option. The results of the experiment are summarized in the following plots,

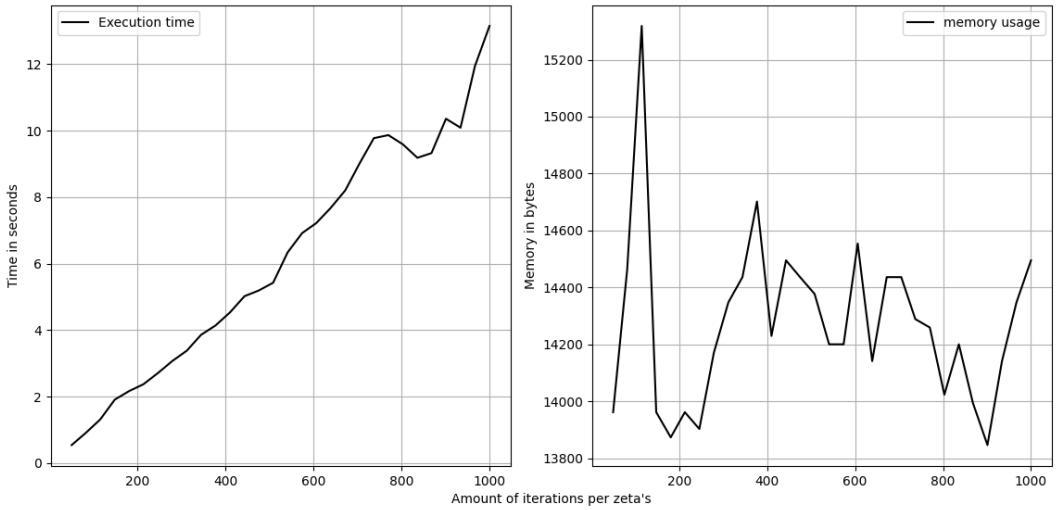


Figure 4.1: Plot of the execution time and memory usage with respect to the number of iterations per zeta. With $n = 100$, $N = 5$, $p = 2, \sigma_u = 0.3$, and 10 repetitions per data points.

As seen in the plot the execution time behaves linear as expected. The memory usage, although a bit jumpy, behaves constant as expected. These jumps are caused by the lower number of repetitions per experiment.
A similar experiment is performed to research the performance of SIMEX with respect to the number of iterations per data points. The results are summarized in the tables below.

	Bias	Variance
β_0	0.2367	0.02066
β_1	0.1793	0.02857

Table 4.1: Bias and variance of the naif estimator for β_0. True regression coefficients of $\beta_0 = 2$ and $\beta_1 = 3$. ME with standard deviation of 0.3. The experiment is ran 200 times.

	Bias(β_0)	Variance(β_0)	Bias(β_1)	Variance(β_1)
M = 50	0.1396	0.02987	0.1497	0.03482
M =100	0.1364	0.02927	0.1499	0.03448
M = 200	0.1416	0.03078	0.1586	0.03805
M = 300	0.1408	0.03034	0.1566	0.03654
M = 500	0.1458	0.03198	0.1630	0.04036
M = 800	0.1343	0.02701	0.1400	0.03056
M = 1000	0.1379	0.02945	0.1530	0.03739

Table 4.2: Bias and variance of the SIMEX estimator for β_0 and β_1 for different choices of M. The rest of parameters are kept constant as $N = 5, n = 1000, p = 1$ with 200 repetitions for each choice of M. The used regression coefficient are $\beta_0 = 2$ and $\beta_1 = 3$. ME has standard deviation of 0.3.

These tables show clearly that the estimators for β are less biased, but that SIMEX creates extra variance in the estimator. The small differences in bias en variance for the different values of M are probably due to the small number of repetitions of the experiment and not caused by the differences in M.

4.2.3 Number of data points.

The number of data points that are created have effect on two parts of the algorithm. The outer *for* loop and the fitting of the model to the new created data points. This outer *for* loop executes two parts of the code, the inner *for* loop and taking an average. The inner *for* loop is because of the same reasons as mentioned before constant in memory use, and so is taking an average. Every time the loop is executed a new data points is created which need to be memorized. Denote the amount of memory needed for memorization of one data point as α. In total there is αN space needed, since N data points are created. So the memory complexity of this *for* loop with respect to the number of data points is $\mathcal{O}(N)$. Now consider the second part that is dependent on the number of data points. As seen previously a least square estimator is used. This can be done by a matrix multiplication, the space complexity would thus be equal to the largest matrix that needs to be stored. In the logistic model a $2-$degree polynomial is fitted,

meaning that the sizes of the matrices in linear regression have the following sizes:

$$\begin{aligned}(X^TX)^{-1}Xy &\to ((d+1)\times N)(N\times(d+1)))^{-1}((d+1)\times N)(N\times 1)\\ &\to ((d+1)\times(d+1))^{-1}((d+1)\times 1)\\ &\to ((d+1)\times(d+1))((d+1)\times 1)\\ &\to ((d+1)\times 1)\end{aligned}$$

It is clear that the largest matrix needed to be stored is a $3 \times N$ matrix. Thus only a linear amount of entries. So the fitting is also linear with respect to the number of data points. Both parts have linear space complexity, thus the entire algorithm is linear in space complexity with respect to N, i.e $\mathcal{O}(N)$.
The time complexity of the outer *for* loop is again linear to the number of data points. Creating a new data point takes a certain amount of time, which is repeated N times. As seen before the least square estimator calculation is just a sequence of matrix operations. It is easy to check that a matrix multiplication of an $(M \times N) \times (N \times M)$ can be performed in NM^2 operations. So one matrix multiplication would be $\mathcal{O}(NM^2)$. When following the aforementioned matrix size changes the largest matrix multiplication that is performed is a $((d+1)\times N)(N\times(d+1))$ matrix operation. Which would be $\mathcal{O}(9N) \subset \mathcal{O}(N)$. Since again both terms are linear this concludes that SIMEX has time complexity of $\mathcal{O}(N)$.
An experiment is performed to test its complexity. All parameters are kept constant expect the number of data points. The number of data points varies from 3 to 20. Since a polynomial fit of degree two is used there need to be at least three data points for a correct fit. As in the previous experiment, for each number of data points SIMEX is repeated 10 times to limit the variation due to randomness, and again three extra number of data points are considered to minimize the influence of earlier executed programs. The results are summarized in the following plots,

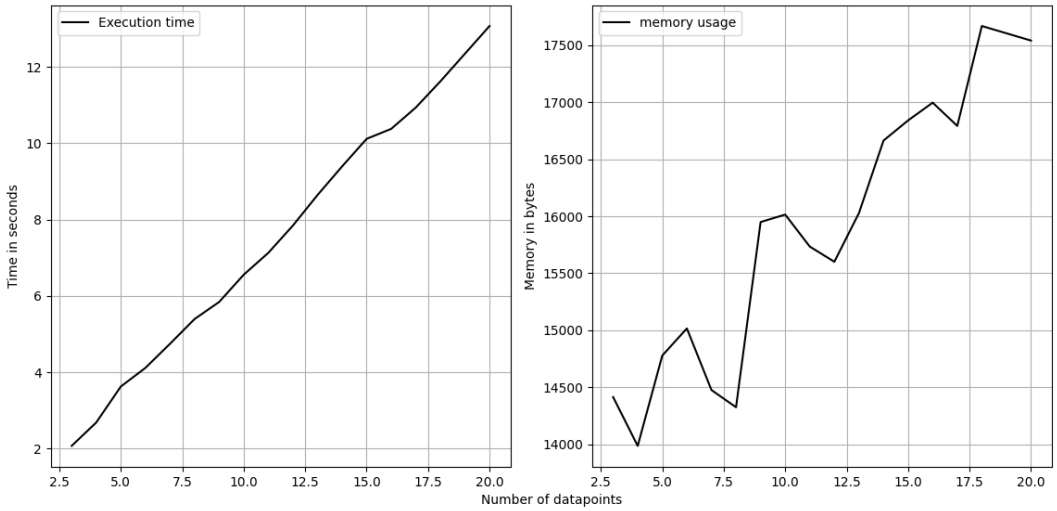


Figure 4.2: Plot of the execution time and memory usage with respect to the number of data points. With $n = 100,\ M = 300,\ p = 2, \sigma_u = 0.3$, and 10 repetitions per data points.

Just as expected both the memory and the execution time behave linear.
An experiment is done to investigate the performance dependence on the number of data points. The results are summarized in the following table,

	Bias	Variance
β_0	0.2554	0.02466
β_1	0.1834	0.02976

Table 4.3: Bias and variance of the naif estimator for β_0. True regression coefficients of $\beta_0 = 2$ and $\beta_1 = 3$. ME with standard deviation of 0.3. The experiment is ran 200 times.

	Bias(β_0)	Variance(β_0)	Bias(β_1)	Variance(β_1)
N = 4	0.1497	0.03463	0.1490	0.03392
N = 6	0.1474	0.03334	0.1545	0.03916
N = 8	0.1376	0.02951	0.1389	0.03035
N = 10	0.1621	0.03672	0.1681	0.04369

Table 4.4: Bias and variance of the SIMEX estimator for β_0 and β_1 for different choices of N. The rest of parameters are kept constant as $M = 300, n = 1000, p = 1$ with 200 repetitions for each choice of N. The used regression coefficient are $\beta_0 = 2$ and $\beta_1 = 3$. ME has standard deviation of 0.3.

The first result that stands out is the difference between 4.1 and 4.3. There estimators

are independent of the choice of N and M and should thus be the same. The difference in the bias and variance are attributable to the fact that for both experiments different data sets, although with the same parameters, are created. Due to the low number of repetitions these different data sets do not create the same tables. Again the small differences in the bias an variance for β are due to the small number of repetitions, and probably not attributable to the difference of N. SIMEX reduces bias for all N, but also increases the variance in the estimator.

4.2.4 Number of explanatory variables

The number of explanatory variables makes a difference in the logistic function and thus also in the way it needs to be fitted. The logistic regression function with p explanatory variables has the following form,

$$\frac{1}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i}}.$$

Which means that SIMEX needs to estimate $p + 1$ parameters, i.e β_i. In each iteration these parameters need to be estimated. As noted before, this is done with an maximum likelihood estimator, its complexity is out of the scope for this thesis. It has a complexity depending on the size of the dataset and the number of explanatory variables. Denote its time complexity as C_p.
The extrapolating part of SIMEX needs to be done for each parameter separately. The time and space complexity of performing this extrapolation is independent of the the parameter for which it extrapolates, i.e extrapolation for β_i takes the same amount of time and space as extrapolation for β_j. Clearly then the extrapolation part of SIMEX has a time complexity linear to the number of parameters. Meaning that entire SIMEX has time complexity of $\mathcal{O}(p + C_p)$.
The space complexity for SIMEX is the same for the simulating part, with C_p denoting the time complexity of fitting to the new data. Since when extrapolating for each parameters the memory used to compute the previous parameter can be reused, causing no more memory to be allocated. The calculated parameter itself also needs to be stored in some way, causing the extrapolation part to still have a linear space complexity. So SIMEX space complexity with respect to the number of explanatory variables is $\mathcal{O}(p + C_p)$. Again an experiment is performed to test these complexities. All parameters are kept constant expect the number of explanatory variables. The number of explanatory variables varies from 2 to 8. As in the previous experiment, for each number of data points SIMEX is repeated 10 times, and again three extra number of data points are calculated but not used. The results are summarized in the following plots,

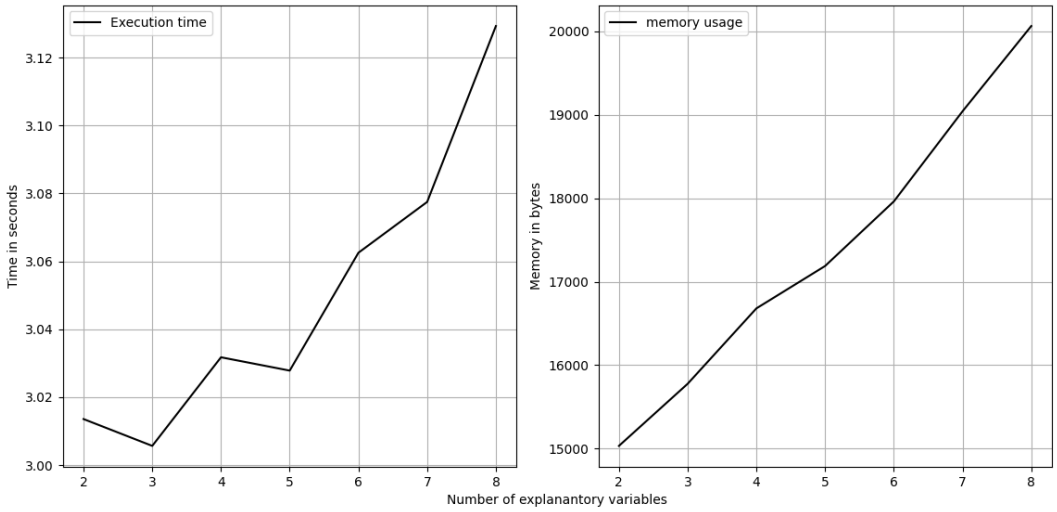


Figure 4.3: Plot of the execution time and memory usage with respect to the number of explanatory variables. With $n = 100,\ M = 300,\ N = 5, \sigma_u = 0.3$, and 10 repetitions per data point.

As argued before the complexity of SIMEX with respect to p is given as $\mathcal{O}(p + C_p)$. This C_p denotes the complexity of fitting the model. In the case of the experiments a logistic fit is used. This has a known time complexity of $\mathcal{O}(np)$ [6], and space complexity of $\mathcal{O}(p)$.[6] Meaning that the space and time complexity both are linear as supported by the plots.

4.2.5 Size of initial dataset

The size of the initial dataset only has a direct influence on the size of the newly created dataset during each iteration. Since these datasets have the same size. The extrapolation part of SIMEX does not depend on n at all, since it does not have an influence on the number of parameters or data points. Consider again the time it takes to fit the logistic regression as an unknown function depending on n, denoted as C_n. The time complexity of SIMEX is $\mathcal{O}(n + C_n)$. For the space complexity the same argument holds. Resulting in a space complexity of also $\mathcal{O}(n + C_n)$. The space complexity for SIMEX is the same for the simulating part, with C_p denoting the time complexity of fitting to the new data. Since when extrapolating for each parameters the memory used to compute the previous parameter can be reused, causing no more memory to be allocated. This would let one believe that it has a constant memory complexity, but off course the calculated parameter itself needs to be stored, causing the extrapolating part to still have a linear space complexity. So SIMEX space complexity with respect to the number of explanatory variables is $\mathcal{O}(n + C_n)$. Again an experiment is performed to test these complexities. All parameters are kept constant expect the size of the dataset. The size of the dataset varies

from 30 to 2000. As in the previous experiment, for each size SIMEX is repeated 10 times, and again three extra sizes are calculated but not used. The results are summarized in the following plots,

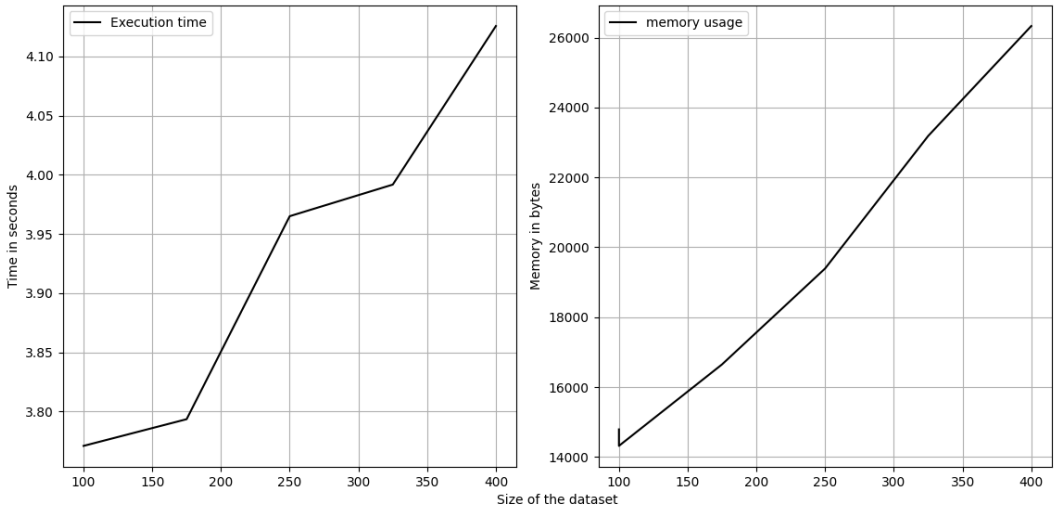


Figure 4.4: Plot of the execution time and memory usage of SIMEX with respect to the size of the dataset with ME. With $n = 100,\ M = 300,\ N = 5, \sigma_u = 0.3$, and 10 repetitions per data point.

As mentioned in the previous section in the logistic case has a known time complexity of $\mathcal{O}(np)$, and space complexity of $\mathcal{O}(p)$. Making the space and time complexity of SIMEX with respect to the size of the dataset respectively $\mathcal{O}(n+p) \subseteq \mathcal{O}(n),\ \mathcal{O}(n+np) \subseteq \mathcal{O}(n)$. The plots reflect this, since they both show a linear relation.
A experiment is performed to see the performance of SIMEX with respect to the number of data points. In similar experiments earlier performed the naif estimators were not dependent on the varied parameter. In this case the naif estimator is depended on the size of the dataset. The results of the experiment are summarized in the tables below,

	Bias(β_0)	Variance(β_0)	Bias(β_1)	Variance(β_1)
n = 100	1.197	1.088	0.5968	1.793
n = 500	1.227	0.04329	0.2255	0.05926
n = 1000	1.227	0.02680	0.1815	0.03256

Table 4.5: Bias and variance of the naif estimator for β_0 and β_1 for different choices of n. The rest of parameters are kept constant as $M = 300, N = 5, p = 1$ with 200 repetitions for each choice of n. The used regression coefficient are $\beta_0 = 2$ and $\beta_1 = 3$. ME has standard deviation of 0.3.

	Bias(β_0)	Variance(β_0)	Bias(β_1)	Variance(β_1)
n = 100	0.6218	2.133	0.6974	3.241
n = 500	0.1985	0.06239	0.2218	0.07661
n = 1000	0.1622	0.03954	0.1613	0.04230

Table 4.6: Bias and variance of the SIMEX estimator for β_0 and β_1 for different choices of n. The rest of parameters are kept constant as $M = 300, N = 5, p = 1$ with 200 repetitions for each choice of n. The used regression coefficient are $\beta_0 = 2$ and $\beta_1 = 3$. ME has standard deviation of 0.3.

Again the first thing to observe is that SIMEX performs significantly better than the naif estimator, but as expected the variance increases. There is a significant difference in the bias of both β_0 and β_1 when going from a dataset of size 100 to 500. This performance difference is way smaller when jumping from 500 to 1000. The second difference is even so small that it is not clear if it is due to the n or to the small number of repetitions.

4.2.6 Complexity of SIMEX

Adding the findings from the previous sections together should yield the total complexity of SIMEX. Starting with the time complexity of SIMEX. Creating the new data is as seen before $\mathcal{O}(n)$. Estimating the parameters to this new data in line 15 has different complexities when the data follows different regression models. The complexity of fitting an arbitrary model is dependent on n and p. Denote its complexity as $\mathcal{O}(C_{n,p})$. These lines need to be executed NM times. Taking the average for each parameter has complexity $\mathcal{O}(p)$, and needs to be executed N times. Resulting in the simulating part of SIMEX to have time complexity $\mathcal{O}(NM(n + C_{n,p}) + Np)$. Lines 19, and 20 used for fitting the model and extrapolating to the true parameter have time complexity $\mathcal{O}(N)$ and $\mathcal{O}(1)$ respectively, and need to be repeated for each parameter. Thus the extrapolating part of SIMEX has time complexity $\mathcal{O}(pN + p)$. The time complexity of SIMEX in total would be $\mathcal{O}(NM(n + C_{n,p}) + 2Np + p) \subset \mathcal{O}(NM(n + C_{n,p}) + Np)$.
The space complexity can be determined in a similar way. Creating the new dataset and estimating the parameters have space complexity $\mathcal{O}(n)$ and $\mathcal{O}(C_{n,p})$. This space needs to be allocated only once, because when repeating these lines the allocated space can be reused. There also needs to be kept track of each parameter for each data point, which allocates Np space units. Resulting in the simulating part to have space com-

plexity $\mathcal{O}(Np + n + C_{n,p})$. The extrapolation part of SIMEX needs the model to be fitted to the data points, taking which as seen before has a linear time complexity to N, and extrapolating for each parameter is linear to p. Since all parameters need to be stored. Causing the simulating part of SIMEX to have a complexity of $\mathcal{O}(N + p)$. So SIMEX in total has a space complexity of $\mathcal{O}(Np+n+C_{n,p}+N+p) \subset \mathcal{O}(Np+n+C_{n,p})$.

5 Conclusion

In this chapter results from this thesis will be discussed,together with possible future experiments will be reviewed. Furthermore the implications of the findings and the significance of the experiments will be discussed.
In the second chapter basic measurement error is introduced. It discussed the influence on both linear regression and logistic regression. The chapter shed light on the differences between the influence of these two models. In the linear regression model the influence of ME on estimators is explicit calculated. It turned out that the logistic regression model becomes a misclassified model when ME is added. The fact that the logistic model is less useful when ME error is added gave an incentive to explore a way to correct for ME. In chapter three such a method is described. As seen SIMEX assumes the variance of the ME to be known, so some methods to estimate the ME variance are discussed as well. SIMEX can in theory produces unbiased estimator for data with ME for several different models. Chapter four researches, with the help of experiments, the performance of SIMEX when different parameters get changed. In the same chapter the time and space complexity are identified, both analytically and numerical. The chapter aims to help make a deliberate choice between performance and cost of the algorithm.
These experiment seem to hint toward the fact that the choice of N and M do not significantly matter. Which would indicate to choose them both as small as possible to keep the computational cost as low as possible. However as will be discussed in the next section these conclusions are to strong for the experiment.

Discussion

The created plots validate the analytical results for the time and space complexity clearly. They are mostly linear as expected. However as mentioned before these plots are spiky. This is probably due to the low number of repetitions of the experiment. These spikes should flatten out with more repetitions, but make it hard to determine the exact trend. Furthermore the memory bug in the solver makes the results not as reliable as desired. It still is possible for the correct implemented solver to omit unexpected behaviour. With regard to the performance analysis the same argument of a lower number of repetitions holds. It is most clearly visible in the difference in bias and variance of the naive estimator, even thought it is independent of N and M. The relatively low number of repetitions causes the trends in the bias and variance instead of the difference in N, or M. Furthermore are all experiments only performed on a single setting. Only normal distributed data is considered with a fixed variance of the ME. It is still possible for a different setting of data and ME to omit a clearer trend in bias and variance of the estimator.

Future experiments

As aforementioned only one particular setting is considered. This choice of setting will not have an influence on the time and space complexity of SIMEX, so these experiment do not have to be repeated for a different setting. The experiments for the bias and variance of the estimators however can be repeated for a number of different settings. Both experiments still can be repeated for a larger number of repetitions to account for the randomness in these experiments. This however is computational quite heavy.
Another possible future experiment to account for this computational heaviness is to look into the opportunity of distributing parallel processes in SIMEX. Since all iterations per zeta are processes are independent of all other iterations these could possibly be done in parallel. This would possibly decrease the computational cost drastically.

Acknowledgements

A honourable mention needs to be made to dr. Eni Musta and dr. Victoria Degeler. I would wish to cite the countless meeting and filled scrap paper which gave me more insight in the subject then all the articles combined.

Bibliography

[1] Bijma, F.,Jonker, M. A., & van der Vaart, A. W., (2011). An Introduction to Mathematical Statistics. Amsterdam University press.

[2] Grace, Y. Yi, (2017). Statistical Analysis with Measurement Error or Misclassification Strategy, Method and Application. Springer.

[3] Carroll, R. J., MEASUREMENT ERROR IN EPIDEMIOLOGIC STUDIES.

[4] Wallace, M., (2020).Analysis in an imperfect world. significancemagazine.com.

[5] Openbaar ministerie, Marges en Meetcorrectie, [https://www.om.nl/onderwerpen/verkeer/handhaving/snelheid-en-te-hard-rijden/marges-en-meetcorrecties].

[6] Kumar, P., Computational Complexity of ML Models. analytics-vidhya. [https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770]

[7] Vighnesh, D. (2020, July 6). MAINT Uses in safe sparse dot check that numba is installed (Fixes #17125) [Issue #17125]. GitHub. [https://github.com/scikit-learn/scikit-learn/issues/17125].

[8] Meeter, IJ. (2024). SIMEX complexity and performance (Version 1.0) [Source code]. GitHub. https://github.com/ysbrandm11/SIMEX-complexity-and-performance

[9] Cook, J. R. and Stefanski, L. A. (1994). Simulation-extrapolation estimation in parametric measurement error models. Journal of the American Statistical Association.

[10] Kekec, E., & Van Keilegom, I. (2022). Estimation of the variance matrix in bivariate classical measurement error models. Electronic Journal of Statistics.

[11] Rosner, B., Willett, W. C., & Spiegelman, D. (Year). Correction of logistic regression relative risk estimates and confidence intervals for systematic within-person measurement error. *Journal Name, Volume*(Issue), Page numbers.

Popular summary

Working with data in theory and in practice differs a lot. While in theory everything works exactly as expected, the real world can be harsh. Measurement devices can collect data with a less accuracy than expected. A clear example of this phenomenon is the fact that the Dutch police handles a three percent error marge on fines for driving above the speed limit [5]. Measurement devices are almost never one hundred percent correct. This measurement error needs to be accounted for. Collected data often has a connection, for instance there is probably a connection between the urge of being on time and how much they are speeding. When people have a higher urge to be on time they will probably be driving a bit faster. When you are interested in the strength of this connection it is important to take this measurement error into account.
This connection can be represented by a number. Where 1 would represent a weak connection and for instance 100 would represent a really strong connection. Ignoring the measurement error naively can form a distorted image of the true connection. There exist estimators that estimate the number that represent connection. There are different ways of accounting for this measurement error. In this thesis we are interested in one particular way of accounting for this measurement error, the SIMEX (SIMulation-EXtrapolation) method. This method works in two steps, as the name already suggests. The first step is simulating more data with an increasing ME. Take the earlier example about the speeding tickets. By using speed cameras that work worse and worse we get a worse estimator for the connection between the urge of being on time and how fast people are driving. This seems like the exact opposite of what we want. However we hope to see a trend in these connections. For instance if we notice that every time we make the speed camera twice as bad, this estimator becomes twice as large we can form an idea of what the true connection should be. If we know that our initial camera is twice as bad as a perfect camera, we directly know that the estimated connection is also twice as bad. Now we can simply divide the estimated connection by two to get the true connection. This is called extrapolating.
To make this method possible we need to know how much worse the speed camera we use is than a perfect camera. A way to do this is to use the perfect camera a couple of times on the same cars as the inaccurate cameras and see the difference. Since this perfect camera is possibly really expensive we can not replace the inaccurate camera everywhere, but we can however measure the difference for all these cameras. SIMEX is a method to correct for measurement error in the estimation of such a connection. In this thesis we take a look at how well this algorithm works in practice, and how much time and memory it needs.