BOOME: A Python package for handling misclassified disease and ultrahigh-dimensional error-prone gene expression data

Li-Pang Chen

Department of Statistics, National Chengchi University, Taipei, Taiwan, ROC

* lchen723@nccu.edu.tw

Abstract

In gene expression data analysis framework, ultrahigh dimensionality and measurement error are ubiquitous features. Therefore, it is crucial to correct measurement error effects and make variable selection when fitting a regression model. In this paper, we introduce a python package BOOME, which refers to BOOsting algorithm for Measurement Error in binary responses and ultrahigh-dimensional predictors. We primarily focus on logistic regression and probit models with responses, predictors, or both contaminated with measurement error. The BOOME aims to address measurement error effects, and employ boosting procedure to make variable selection and estimation.

1 Introduction

1.1 Motivation

Analysis of gene expression data is a popular topic and deserves careful research development. A motivating example in this paper is a gene expression microarray data collected by [1] and explored in some references (e.g., [2]). The full dataset can be found in the R package SIS. The data contain binary responses, acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL), and 7128 gene expression levels that were measured using Affymetrix oligonucleotide arrays. In addition, samples with size 72 come from the two classes, with 47 specimens in class ALL and 25 specimens in class AML. The primary objective of this study is to characterize the relationship between leukemia and gene expression values, and see how gene expression values interpret leukemia. To achieve this goal, a commonly used approach is to build a regression model by treating leukemia and gene expression values as binary responses and the predictors, respectively. To model a binary response, logistic regression or probit models are perhaps frequently implemented parametric approaches.

According to this gene expression data, ultrahigh dimensionality ($p \gg n$) is a challenging feature. Since not every gene expression value is informative, using irrelevant predictors in regression models may affect the performance of classification and induce wrong conclusions. Therefore, making variable selection and retaining important ones are needed. While variable selection techniques have been widely explored (e.g., [3–5]), those strategies may fail to handle
the case that the dimension is extremely larger than the sample size. The other concern is measurement error in the response and predictors. As discussed in [6–8], gene expression values may be measured imprecisely due to unadjusted machines. Moreover, as commented by [9], it is also possible to falsely record ALL (or AML) to AML (or ALL), known as misclassification, because the microscopy images of AML bone marrow cells contain many immature granulocytes and monocytes, and ALL bone marrow cell microscopy images contain many immature lymphocytes. It is known that ignoring measurement error effects may cause tremendous biases and induce incorrect decisions, such as the false exclusion of truly informative predictors or false inclusion of irrelevant predictors when making variable selection (e.g., [6, 10, 11]). Therefore, it is crucial to correct measurement error effects. In particular, unlike existing literature that handles either variable selection or measurement error, the main challenge of this dataset is to correct measurement error and select informative predictors under ultrahigh-dimensional data simultaneously, and measurement error may explicitly affect the performance of variable selection. In other words, truly noninformative predictors may be falsely included if measurement error effects are ignored (e.g., [6, 11, 12]). As a result, it is necessary to suitably adjust measurement error effects and then use the corrected version to make variable selection.

1.2 Contributions

To address those concerns, we develop a package **BOOME** that is now available on https://pypi.org/project/BOOME/0.0.2/. The purpose of this package is to correct two measurement error processes in responses and predictors, and employ the boosting procedure to retain important predictors and estimate nonzero coefficients simultaneously.

In standard analysis of regression models, to estimate unknown parameters, one may require to derive likelihood functions, or more generally, unbiased estimating functions. Then the resulting estimator can be obtained by optimizing the constructed estimating functions. In the presence of measurement error, however, naively adopting error-prone predictors to the estimating functions would yield the biased estimators (e.g., [10, 13]). Therefore, to address this challenge, as discussed in measurement error framework, one should derive the corrected estimating functions with measurement error effects eliminated before implementing computational algorithms or estimation methods to derive the estimator, which is the standard step in measurement error analysis (e.g., [6, 10–18]). Following this idea, our strategy is to derive a new estimating function with measurement error effects in responses and predictors corrected, then adopt it to select informative predictors and obtain the corresponding estimators. Specifically, to correct measurement error effects to the binary response, we define the misclassification matrix (e.g., [13], p.131), which is formulated by specificity and sensitivity (e.g., [13], p.70), and will be described in details in Section 2.2, to adjust for measurement error effects in the responses and derive a new corrected response. Regarding the error-prone predictors, we adopt the sufficient statistics of the predictors and the regression calibration to correct measurement error effects to the predictors. Based on such strategies of measurement error corrections, we develop the corrected estimating functions under logistic regression or probit models, respectively. After that, we implement the corrected estimating functions to the boosting algorithm to make variable selection and estimation (e.g., [19]; [20], p.608). Detailed descriptions of measurement error corrections and the boosting algorithm are deferred to Sections 3.1 and 3.2, respectively.

1.3 Comparisons

Variable selection and estimation with correction of measurement error have been discussed, and many methods based on different settings have been developed. For example, [21] consider generalized linear models (GLM) and proposed the generalized matrix uncertainty
selector (GMUS), whose idea is based on a Taylor series expansion of the GLM mean function around the true, but unknown, predictors. [22] considered parametric and semi-parametric regression models with error-prone predictors, and developed a corrected estimating equation to make variable selection. For survival data with incomplete responses, [6, 11, 12] considered several types of survival models and developed penalized estimating function to deal with variable selection. [23] proposed the MEBoost method, which adopts the boosting method to select informative variables under error-prone linear regression models. While many methods have been developed, they primarily focus on measurement error in predictors and rare work has been available to address measurement error effects in responses. In addition, although the boosting method has been applied to error-prone data, the existing method simply focuses on linear regression models, and other types of regression models have not been explored.

In the past developments, several existing packages based on different software have been developed to deal with either measurement error or variable selection. To name a few, for the R software, two packages glmnet [24] and SIS [25] are popular methods to handle variable selection. For the Python software, xverse [26] can be adopted to do feature selection. However, they fail to deal with measurement error effects. On the other hand, the two packages GLSME [27] and mecor [28] in the R software focus on linear models and aim to adjust for measurement error effects in the response and/or predictors, but they cannot deal with variable selection.

Compared with existing packages, there are some differences from the package BOOME. Specifically, our package is able to handle ultrahigh-dimensionality and mismeasured data simultaneously. Unlike most existing frameworks that focus on measurement error in predictors or continuous responses, our approach extends measurement error in binary responses (a.k.a misclassification), and the model structure for misclassification is more complex than that under continuous responses. Our approach can deal with error-prone response and predictors simultaneously. Moreover, boosting iteration may reduce the possibility of falsely excluding important predictors and enhance the accuracy of the estimator. Most importantly, our development is based on the Python language, and, to the best of knowledge, there is no relevant development in Python packages.

1.4 Organization of this paper
The remainder is organized as follows. In Section 2, we introduce two regression models to characterize binary responses. In addition, we introduce two measurement error models to describe error-prone responses and predictors, respectively. In Section 3, we present the BOOME method. Specifically, we first discuss some valid strategies to handle measurement error effects, and then discuss the boosting method for variable selection and estimation. In Section 4, we introduce the Python package BOOME, including some important functions as well as their implementation. In Section 5, we demonstrate the application of the package BOOME and analyze the gene expression data. Moreover, we also demonstrate simulation studies. Finally, a general discussion is presented in Section 6.

2 Regression models
2.1 Regression models with binary responses
Following the motivating example in Section 1.1, let \( n = 72 \) denote the sample size. For \( i = 1, \ldots, n \), let \( Y_i \) be a binary response where \( Y_i = 1 \) represents AML and \( Y_i = 0 \) indicates ALL. Moreover, let \( X_i \) be a \( p \)-dimensional vector of gene expressions with \( p = 7128 \).
With the absence of measurement error effects, our goal is to use the gene expression values \( X_i \) to characterize the disease \( Y_i \) through a \( p \)-dimensional vector of parameters \( \beta \). In the framework of GLM, logistic regression or probit models are commonly used. Specifically, the logistic regression model (LR) is formulated by

\[
P( Y_i = 1 | X_i) = \frac{\exp(X_i^\top \beta)}{1 + \exp(X_i^\top \beta)};
\]

and the probit model (PM) is given by

\[
P( Y_i = 1 | X_i) = \Phi(X_i^\top \beta),
\]

where \( \Phi(\cdot) \) is the cumulative distribution function of the standard normal distribution.

To estimate \( \beta \), a common strategy is to optimize the likelihood function, or equivalently, solve the estimating equation. Specifically, for \( i = 1, \ldots, n \), the estimating function based on (1) is defined as

\[
g_{LR}(\beta) = - \frac{1}{n} \sum_{i=1}^{n} \left\{ X_i Y_i - \frac{X_i \exp(X_i^\top \beta)}{1 + \exp(X_i^\top \beta)} \right\};
\]

and the estimating function based on (2) is given by

\[
g_{PM}(\beta) = - \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{X_i}{\sqrt{2\pi}} \exp \left\{ - \frac{1}{2} (X_i^\top \beta)^2 \right\} \left\{ \frac{Y_i}{\Phi(X_i^\top \beta)} + \frac{Y_i - 1}{1 - \Phi(X_i^\top \beta)} \right\} \right].
\]

Solving \( g_{LR}(\beta) = 0 \) or \( g_{PM}(\beta) = 0 \) yields the estimator of \( \beta \).

### 2.2 Measurement error models

In applications, as discussed in Section 1.1, \( Y_i \) and \( X_i \) might be subject to measurement error due to wrong records by investigators or imprecise measurements by unadjusted machines. Under this scenario, we particularly denote \( Y_i \) and \( X_i \) as unobserved variables, and let \( Y_i^* \) and \( X_i^* \) denote the surrogate measurements of \( Y_i \) and \( X_i \), respectively, and they are recorded in the data.

We now provide an intuition of modeling error-prone data. Let \( f(\cdot|\cdot) \) represent the conditional distribution for variables indicated by the corresponding arguments, and let \( f(\cdot) \) denote marginal or joint distribution of random variables. Following the similar discussion in [13] (Chapter 8), we consider the joint distribution \( f(Y_i, Y_i^*, X_i, X_i^*) \) and factorize it as

\[
f(Y_i, Y_i^*, X_i, X_i^*) = f(Y_i|Y_i^*, X_i) \times f(Y_i^*|X_i, X_i) \times f(X_i^*|X_i) \times f(X_i),
\]

where the second step is obtained by the nondifferential \( X_i^* \). With the marginal distribution \( f(X_i) \) left specified, the factorization (5) says that the inference about \( f(Y_i|X_i) \) is conducted based on examining \( f(Y_i^*|X_i^*, X_i) \) with the predictor measurement error process being characterized by \( f(X_i^*|X_i) \), and \( f(Y_i^*|Y_i, X_i) \) facilitates the response measurement error process.

To analyze measurement error effects when constructing regression models, we first need to characterize the relationship between \( Y_i \) and \( Y_i^* \) as well as \( X_i \) and \( X_i^* \). Specifically, to connect \( Y_i \) and \( Y_i^* \), we consider the conditional probability

\[
\pi_{kl} = P(Y_i^* = k | Y_i = l, X_i)
\]

for \( k, l \in \{0, 1\} \), satisfying \( \pi_{110} + \pi_{00} = 1 \) and \( \pi_{01} + \pi_{11} = 1 \), where \( \pi_{11} \) and \( \pi_{00} \) are called...
sensitivity and specificity, respectively, or known as classification probabilities; \( \pi_{10} \) and \( \pi_{01} \) are known as misclassification probabilities (e.g., [13], p.70). Moreover, to characterize \( \pi_{01} \) and \( \pi_{10} \), logistic regression (1) or probit models (2) with additional parameter \( \gamma \) are suitable choices. By the law of total probability, \( P(Y_i = 1|X_i) \) and \( P(Y_i = 0|X_i) \) can be expressed as

\[
\begin{pmatrix}
P(Y_i = 1|X_i) \\
P(Y_i = 0|X_i)
\end{pmatrix} = \Pi_i \begin{pmatrix}
P(Y = 1|X_i) \\
P(Y = 0|X_i)
\end{pmatrix},
\]

where \( \Pi_i = \begin{pmatrix} \pi_{11} & \pi_{10} \\ \pi_{01} & \pi_{00} \end{pmatrix} \) is called a 2 \( \times \) 2 misclassification matrix (e.g., [13], p.131) that is assumed to be invertible.

Next, to describe the relationship between \( X_i \) and \( X_o \), we employ the classical measurement error model

\[
X_i = X_o + \epsilon_i,
\]

where \( \epsilon_i \) is independently and identically distributed normal distribution \( N(0, \Sigma) \) with \( \Sigma \) being a \( p \times p \) covariance matrix representing the magnitude of measurement error effects in the predictors. We assume that \( \epsilon_i \) is independent of \( X_i \).

3 Method

3.1 Correction of measurement error

In this section, we primarily correct measurement error effects to responses and predictors.

Motivated by (7), by multiplying the inverse matrix of \( \Pi_i \) to both sides of (7), we can obtain that

\[
P(Y_i = 1|X_i) = \frac{P(Y_i^* = 1|X_i) - \pi_{10}}{1 - \pi_{10} - \pi_{01}},
\]

which indicates that the unobserved response \( Y_i = 1 \) can be implicitly characterized by \( Y_i^* = 1 \) with the adjustment in terms of \( \pi_{01} \) and \( \pi_{10} \). It motivates us to consider the "corrected" response, denoted \( Y_i^{**} \), which satisfies

\[
P(Y_i^{**} = 1|X_i) = \frac{P(Y_i^* = 1|X_i) - \pi_{10}}{1 - \pi_{10} - \pi_{01}},
\]

suggesting that

\[
Y_i^{**} = \frac{Y_i^* - \pi_{10}}{1 - \pi_{10} - \pi_{01}}.
\]

In addition, (11) indicates that \( E(Y_i^{**}|Y_i, X_i) = Y_i \), verifying that (11) is a suitable correction to recover \( Y_i \). Moreover, we note that (11) holds regardless of the choice of regression models because it is obtained by the equalities (9) and (10) where the conditional probability can be (1) or (2).

To correct measurement error effects to the predictors, we provide two different strategies for different models. For the logistic regression model in terms of \( Y_i^{**} \) and unobserved \( X_o \), we follow the similar discussion in [18] and aim to replace \( X_i \) by its sufficient statistic:

\[
X_{SS}^{**} = X_i^* + Y_i^{**} \Sigma_i \beta,
\]

which can be regarded as correction of \( X_i^* \). Replacing \( Y_i \) and \( X_i \) in (1) by (11) and (12) gives the
corrected estimating function:

\[
g_{1R}^*(\beta) = -\frac{1}{n} \sum_{i=1}^{n} \left[ (X_i^* + Y_i^* \Sigma_j \beta)Y_i^* - (X_i^* + Y_i^* \Sigma_j \beta) \exp \left\{ \frac{(X_i^* + Y_i^* \Sigma_j \beta)^T \beta}{1 + \exp \left\{ (X_i^* + Y_i^* \Sigma_j \beta)^T \beta \right\} \right\} \right]. \tag{13}
\]

On the other hand, to handle measurement error effects to the probit model, we adopt the regression calibration (e.g., [10], Chapter 4), whose key idea is to replace \( X_i \) by the conditional expectation \( E(X_i|X_i^*) \). By the best linear unbiased prediction, it can be expressed as (e.g., [11, 15])

\[
E(X_i|X_i^*) = \mu_X + (\Sigma_{X^r} - \Sigma_j)^T \Sigma_j^{-1} (X_i^* - \mu_{X^r}),
\]

where \( \mu_X \) and \( \mu_{X^r} \) represent the mean vectors of \( X \) and \( X^r \), respectively, and \( \Sigma_{X^r} \) represents the covariance matrix of \( X \). Since \( \mu_X = \mu_{X^r} \), by the method of moments, we obtain that

\[
X_i^* \sim \mu_{X^r} + (\Sigma_{X^r} - \Sigma_j)^T \Sigma_j^{-1} (X_i^* - \mu_{X^r}), \tag{14}
\]

where \( \mu_{X^r} \) and \( \Sigma_{X^r} \) are empirical estimates of \( \mu_{X^r} \) and \( \Sigma_{X^r} \), respectively. Consequently, replacing \( Y_i \) and \( X_i \) in (2) by (11) and (14) gives the corrected estimating function:

\[
g_{1M}^*(\beta) = -\frac{1}{n} \sum_{i=1}^{n} \left[ \frac{X_i^*}{\sqrt{2\pi}} \exp \left\{ -\frac{1}{2} (X_i^* \beta)^T (X_i^* \beta) \right\} \right] \left\{ \frac{Y_i^*}{\Phi(X_i^* \beta)} + \frac{Y_i^* - 1}{1 - \Phi(X_i^* \beta)} \right\}. \tag{15}
\]

### 3.2 Boosting algorithm

Let \( g^*(\beta) \) denote the unified notation to represent the corrected estimating function (13) or (15). To make variable selection and estimation for \( \beta \), we adopt the boosting algorithm with the correction of measurement error effects. The proposed method is called BOOME, and the procedure is summarized in Algorithm 1.

Specifically, the algorithm starts by an initial value \( \beta^{(0)} \) taken by the \( p \)-dimensional zero vector \( 0_p \). Suppose that we run \( T \) times iterations, and for each iteration step \( t = 1, \ldots, T \), we compute the estimating function \( g^*(\beta) \) evaluated at the \((t - 1)\)th iterated value \( \beta^{(t-1)} \), and denote it as \( \Delta^{(t-1)} \). After that, we define the active set \( J_{t-1} \) that define the indexes satisfying \( |\Delta_j^{(t-1)}| \geq \tau \max |\Delta_j^{(t-1)}| \), where \( \tau \in [0, 1] \) is a constant and \( \Delta_j^{(t-1)} \) is the \( j \)th component in a vector \( \Delta^{(t-1)} \). It implies that the active set \( J_{t-1} \) aims to retain informative predictors by treating \( \Delta_j^{(t-1)} \) as a signal. Finally, for those \( j \in J_{t-1} \), we update the iterated value of the \( j \)th component in \( \beta^{(t-1)} \), say \( \beta_j^{(t-1)} \), by adding an increment \( \eta \cdot \Delta_j^{(t-1)} \) for some positive constant \( \eta \). Repeating those steps \( T \) times yields the final estimator of \( \beta \).

In Algorithm 1, \( \tau, \eta, \) and \( T \) can be user-specific and may affect the iteration result. Similar to the comment in [29], the algorithm satisfying \( T\eta \rightarrow 0 \) as \( T \rightarrow \infty \) and \( \eta \rightarrow 0 \) is approximately equivalent to the LASSO method. Therefore, it suggests taking \( \eta \) as a small value, such as \( \eta = 0.01 \), in applications. On the other hand, while \( T \) is suggested being large, it may cause over-fitting. To provide a suitable \( T \) and stop the iteration earlier, we suggest a criterion: the iteration stops at \( T \) if

\[
\| g^*(\beta^{(T)}) - g^*(\beta^{(T+1)}) \| < \xi
\]

is satisfied for some positive constant \( \xi \). Finally, for the choice of \( \tau \), one may adopt some criteria such as cross-validation (e.g., [19]).

**Algorithm 1:** Boosting Procedure in BOOME

---

Python package: BOOME
Let $\beta^{(0)} = 0_p$ denote an initial value;
for step $t$ with $t = 1, 2, \ldots, T$
do
(a) calculate $\Delta^{(t-1)} = \Delta^* (\beta) |_{\beta = \beta^{(t-1)}}$;
(b) determine $J_{t-1} = \{j : |\Delta_j^{(t-1)}| \geq \tau \max |\Delta^{(t-1)}|\}$;
(c) update $\beta_j^{(t)} \leftarrow \beta_j^{(t-1)} + \eta \Delta_j^{(t-1)}$ for all $j \in J_{t-1}$, and define
$\beta^{(0)} = (\beta_1^{(0)}, \ldots, \beta_p^{(0)})$;
The final estimator is given by $\hat{\beta} \triangleq \beta^{(T)}$.

4 Description and implementation of BOOME

We develop a Python package, called BOOME, to implement the variable selection and estimation with measurement error correction described in Section 3. The package BOOME contains three functions: ME_Generate, LR_Boost, and PM_Boost. The function ME_Generate aims to generate artificial data under specific models listed in Section 2.1 and error-prone predictors. The functions LR_Boost and PM_Boost implement the boosting procedure in Algorithm 1, except for the difference that LR_Boost is based on the logistic regression model, and PM_Boost focuses on the probit model. We now describe the details of these three functions.

4.1 ME_Generate

We use the following command to obtain the artificial data:

```
ME_Generate(n, beta, matrix, X, gamma)
```

where the meaning of each argument is described as follows:

- $n$: The number of observations.
- $\beta$: A $p$-dimensional vector of parameter $\beta$ specified by users.
- $\text{matrix}$: A user-specific covariance matrix implemented to (8).
- $X$: A user-specific $n \times p$ matrix of predictors.
- $\gamma$: A $p$-dimensional vector of parameter $\gamma$ in $\pi_{10}$ and $\pi_{01}$ specified by users.

The function ME_Generate returns a list of components:

- $\text{data}$: A dataset with error-prone predictors and responses. It is an $n \times (p+1)$ data frame, where the column with label $y$ represents the error-prone response, and the column with label $j, j = 1, \ldots, p$, represents the $j$th error-prone predictor $X_j$.
- $pr$: Two misclassification probabilities $\pi_{10}$ and $\pi_{01}$ in (7).

4.2 LR_Boost

To demonstrate Algorithm 1 with the corrected estimating function (13) for the logistic regression model, we adopt the following command:

```
LR_Boost(X, Y, ite, thres, correctX, correctY, pr, lr, matrix)
```

where the meaning of each argument is described as follows:

- $X$: A $n \times p$ matrix of continuous predictors that are precisely measured or subject to measurement error.
• **Y**: A \( n \)-dimensional vector of binary responses that are precisely measured or subject to measurement error.

• **ite**: A number of iteration \( T \) in Algorithm 1.

• **thres**: A threshold value \( \tau \) in Algorithm 1.

• **correct_X**: Determine the correction of measurement error in predictors. Select “1” if correction is needed, and “0” otherwise.

• **correct_Y**: Determine the correction of measurement error in the response. Select “1” if correction is needed, and “0” otherwise.

• **pr**: Two misclassification probabilities \( \pi_{10} \) and \( \pi_{01} \) in (7).

• **lr**: A learning rate \( \eta \) in Algorithm 1.

• **matrix**: A \( p \times p \) covariance matrix \( \Sigma \) in (8).

The function \( \text{LR}_\text{Boost} \) returns a list of components:

• **estimated coefficients**: the \( p \)-dimensional vector of estimators of \( \beta \).

• **predictors**: Indexes of nonzero values in estimated coefficients.

• **number of predictors**: The number of nonzero values in estimated coefficients.

### 4.3 PM_Boost

To make variable selection and estimation for probit model by using Algorithm 1 with the corrected estimating function (15), we implement the following function:

\[
\text{PM}_\text{Boost}(X, Y, \text{ite}, \text{thres}, \text{correct}_X, \text{correct}_Y, \text{pr}, \text{lr}, \text{matrix})
\]

The arguments in \( \text{PM}_\text{Boost} \) as well as the output produced by \( \text{PM}_\text{Boost} \) are the same as those in \( \text{LR}_\text{Boost} \).

### 5 Numerical studies

In this section, we implement the functions in the package BOOME to analyze a real dataset as well as demonstrate simulation studies. Detailed code demonstrations are also available on the pypi website https://pypi.org/project/BOOME/0.0.2/.

#### 5.1 Analysis of gene expression microarray data

In this section, we implement the package BOOME to analyze a gene expression microarray data introduced in Section 1.1. The steps for analysis are summarized in Fig 1. As shown in Step 1 of Fig 1, we recognize that, for \( i = 1, \ldots, n \), \( Y_i \) is the binary random variable with outcomes AML and ALL that may possibly subject to misclassification, and \( X_i \) represents the gene expression values that are contaminated with measurement error. Before analyzing this dataset, we first standardize all predictors, such that the mean and the variance of each predictor become 0 and 1, respectively. Let data GE in Python code represent the gene expression microarray data that we introduced in Section 1.1, where the first column is the binary outcome and the remaining columns are gene expression values. Based on this dataset, the
The following code shows the input of gene expression data and the standardized procedure:

```python
x = data_GE.drop(['y'], axis=1); y = data_GE[['y']]
zscores = preprocessing.StandardScaler()
x_z = pd.DataFrame(zscores.fit_transform(x))
x_z.columns = range(1,719)
data_z1 = pd.concat([x_z,y],axis=1)
```

To examine the impact of measurement error effects, we primarily consider four settings in Step 2 of Fig 1:

- Setting 1: $Y_i^*$ and $X_i^*$ are not corrected.
- Setting 2: $Y_i^*$ is corrected while $X_i^*$ is not.
- Setting 3: $X_i^*$ is corrected while $Y_i^*$ is not.
- Setting 4: $Y_i^*$ and $X_i^*$ are corrected.

Here Setting 1 aims to implement Algorithm 1 to the estimating functions (3) or (4) with $Y_i$ and $X_i$ replaced by error-prone variables $Y_i^*$ and $X_i^*$, respectively. Setting 2 considers the estimating function in Setting 1 with $Y_i^*$ replaced by corrected responses (11); and Setting 3 adopts the estimating function in Setting 1 with $X_i^*$ replaced by corrected predictors (12) or (14). Setting 4 uses Algorithm 1 to estimating functions (13) or (15), where measurement error in both responses and predictors are corrected. As discussed in Section 1.1, both responses and predictors are contaminated with measurement error, then Setting 4 is the proposed method by correcting measurement error effects in responses and predictors. On the other hand, Settings 1-3, known as naive methods, reflect that measurement error in leukemia, gene expressions, or...
both are not corrected. Basically, Settings 1-3 are considered to show the impact of ignorance of measurement error effects and are compared with the proposed method in Setting 4.

In Step 3, we implement the functions in BOOME for four different settings. We first note that the dataset has no additional information, such as repeated measurements or validation data, to estimate parameters \( \Sigma \), in measurement error models as well as two misclassification probabilities \( \pi_{01} \) and \( \pi_{10} \), here we conduct sensitivity analyses, which specify reasonable values for \( \Sigma \) and enable us to examine the impact of different magnitudes of measurement error. In our study, we specify \( \Sigma \) as a diagonal matrix with diagonal entries \( \sigma_i^2 \) being 0.2, 0.5, and 0.7. For the implementation, we take \( \sigma_i^2 = 0.2 \) as an example and use the following command to demonstrate \( \Sigma \), denoted as matrix:

\[
\text{matrix} = 0.2 \times \text{np.identity}(7128)
\]

With \( \sigma_i^2 \) being specified, we further determine misclassification probabilities \( \pi_{10} \) and \( \pi_{01} \). Specifically, since \( \pi_{10} \) and \( \pi_{01} \) defined in (6) rely on \( X_i \), we reproduce \( X_i \) by (8), where \( X_i \) is observed gene expression values and \( \epsilon_i \) is generated from a normal distribution with \( \sigma_i^2 \) given by 0.2, 0.5 or 0.7. After that, we adopt logistic regression or probit models to characterize (6) with the corresponding parameter specified as \( \gamma \triangleq 1_p \), where \( 1_p \) is a \( p \)-dimensional vector with all entries being one. Therefore, values of \( \pi_{10} \) and \( \pi_{01} \) are obtained. To show the demonstration, we summarize the following function \( \text{pi} \) that is used to implement this idea and compute \( \pi_{10} \) and \( \pi_{01} \). The resulting values of \( \pi_{10} \) and \( \pi_{01} \) are denoted as \( \text{pr} \):

```python
def pi(df, cov):
    x = np.array(df[['y']], axis = 1)
    p = df.shape[1] - 1
    n = len(df)
    def mean(p):
        zero = [0]
        return zero * p
    p = df.shape[1] - 1
    m = []
    p = df.shape[1] - 1
    for i in range(n):
        p = df.shape[1] - 1
        e = np.array(np.random.multivariate_normal(mean(p), cov))
        p = df.shape[1] - 1
        x = np.array(df.T[i] + p) - e
        m_list = []
        scalar = math.exp(1 + np.dot(x, np.array([1] * p)))
        matrix = np.array([[1/(1 + scalar), scalar/(1 + scalar)],
                           [scalar/(1 + scalar), 1/(1 + scalar)]])
        m_list.append(matrix[0][i])
        m_list.append(matrix[1][i])
        m.append(m_list)
    m = np.array(m)
    return m
pr = pi(data_z1, matrix)
```
We now implement two functions \texttt{LR\_Boost} and \texttt{PM\_Boost} to analyze the data, where, for logistic regression model, \( T \) is given by 1000, \( \eta \) is set as 0.01, and \( r \) is equal to 0.9; for probit models, \( T \) is given by 2000, \( \eta \) is set as 0.01, and \( r \) is equal to 0.9.

Detailed implementations of the proposed method are described below, keeping in mind that we demonstrate \texttt{correct\_X = 1} and \texttt{correct\_Y = 1} to show the proposed method (Setting 4); different values for arguments \texttt{correct\_X} and \texttt{correct\_Y} reflect different settings mentioned above.

\[
\texttt{LR\_Boost}(X,Y, \texttt{ite} = 1000, \texttt{thres} = 0.9, \texttt{correct\_X = 1}, \texttt{correct\_Y = 1}, \texttt{pr}, \texttt{lr} = 0.01, \texttt{matrix})
\]

\[
\texttt{PM\_Boost}(X,Y, \texttt{ite} = 2000, \texttt{thres} = 0.9, \texttt{correct\_X = 1}, \texttt{correct\_Y = 1}, \texttt{pr}, \texttt{lr} = 0.01, \texttt{matrix})
\]

In Step 4 of Fig 1, we report the estimation results. To save limited space and provide precise information, we summarize the predictors and their estimates that are commonly selected under \( \alpha_i^2 = 0.2, 0.5 \) or 0.7, and numerical results for all settings obtained by (1) and (2) are placed in Tables 1 and 2, respectively. Moreover, to see the impacts of different regression models, we summarize commonly chosen predictors in Table 3.

We first examine Setting 1 where measurement error corrections are not incorporated. Based on \texttt{BOOME}, the logistic regression model retains 45 gene expression values, and the probit model suggests that 53 gene expression values should be included. Next, we explore the case that either the response or the predictors are corrected. Under Setting 2, the logistic regression model retains 59 gene expression values, and the probit model suggests that 74 gene expression values should be included. Under Setting 3, the logistic regression model retains 42 gene expression values, and the probit model suggests that 36 gene expression values should be included. Finally, under Setting 4 where measurement error effects response and the predictors are corrected, we have that the logistic regression model retains 51 gene expression values, and the probit model retains 75 gene expression values.

For the overall comparisons, we first observe that the variable selection result may depend on the correction of measurement error effects in the response and/or the predictors. The number of selected gene expressions under Setting 1 is almost smaller than that under other settings. For two regression models, the probit model retains more predictors than what the logistic regression model does, except for Setting 3. Finally, there are 35, 45, 29, and 8 gene expressions that are commonly selected by two models under Settings 1-4, respectively.

5.2 Demonstration of simulation studies

To show the validity of the BOOME method as well as the implementation of the package, we conduct simulation studies and demonstrate the programming code in this section.

Let \( n = 100 \) denote the sample size, and let \( p = 1000 \) or 5000 denote the dimension of predictors. For \( i = 1, \ldots, n \), we generate the \( p \)-dimensional vector of predictors \( X_i \) from the standard multivariate normal distribution. Let \( \beta_0 = (1,1,1,0_{p-3}^T) \) denote the true value of parameters, where \( 0_q \) represents the \( q \)-dimensional zero vector. Given \( X_i \) and \( \beta_0 \), we generate the binary response \( Y_i \).

Noting that \( \{ (Y_i, X_i): i = 1, \ldots, n \} \) is regarded as \textit{unobserved} data, we now generate error-prone data \( \{ (Y'_i, X'_i): i = 1, \ldots, n \} \). For the generation of error-prone responses \( Y'_i \), we
| #  | Setting 1 |       | Setting 2 |       | Setting 3 |       | Setting 4 |       |
|----|----------|-------|----------|-------|----------|-------|----------|-------|
| 1  | 134      | -0.208| 7        | 0.41  | 134      | -0.04 | 7        | 0.438 |
| 2  | 412      | -0.606| 268      | -0.684| 412      | -0.484| 268      | -0.338|
| 3  | 760      | 0.271 | 731      | -1.161| 635      | -0.186| 731      | -1.456|
| 4  | 804      | -1.271| 746      | -0.11 | 738      | 0.196 | 746      | -0.045|
| 5  | 1144     | -0.09 | 1024     | -0.032| 760      | 0.271 | 1089     | -0.12 |
| 6  | 1207     | -0.108| 1089     | -1.301| 804      | 0.243 | 1148     | 0.584 |
| 7  | 1239     | -0.566| 1148     | 0.475 | 1144     | -0.123| 1307     | 0.446 |
| 8  | 1260     | 0.786 | 1307     | 0.742 | 1207     | -0.115| 1588     | -0.132|
| 9  | 1669     | 0.155 | 1687     | -0.153| 1239     | -0.234| 1687     | -0.085|
| 10 | 1685     | -0.169| 1708     | -0.229| 1260     | 0.912 | 1858     | 1.043 |
| 11 | 1834     | 1.471 | 1724     | 0.049 | 1669     | 0.013 | 1987     | 0.327 |
| 12 | 1882     | 1.337 | 1858     | 1.126 | 1685     | -0.122| 2039     | 1.184 |
| 13 | 1928     | -1.229| 1987     | 0.46  | 1834     | 1.496 | 2059     | -0.581|
| 14 | 1997     | 0.132 | 2039     | 0.903 | 1882     | 1.271 | 2065     | -0.294|
| 15 | 2121     | 0.033 | 2047     | -0.07 | 1928     | -1.221| 2377     | 1.677 |
| 16 | 2288     | 0.414 | 2057     | -0.175| 1997     | 0.155 | 2460     | 0.281 |
| 17 | 2354     | -0.087| 2059     | -0.169| 2288     | 0.858 | 2469     | -0.039|
| 18 | 2441     | -0.032| 2208     | 0.166 | 2354     | -0.085| 2534     | -0.799|
| 19 | 2737     | 0.051 | 2377     | 1.389 | 2737     | 0.054 | 2895     | 0.035 |
| 20 | 2797     | 0.052 | 2460     | 0.55  | 2797     | 0.075 | 2941     | 0.058 |
| 21 | 3183     | 0.376 | 2534     | -0.969| 3183     | 0.228 | 3276     | 0.104 |
| 22 | 3252     | 0.464 | 2682     | 0.169 | 3252     | 0.801 | 3432     | -0.631|
| 23 | 3714     | 0.078 | 2895     | 0.102 | 3714     | 0.022 | 3660     | 0.07  |
| 24 | 3847     | 1.529 | 2941     | 0.733 | 3847     | 1.553 | 3771     | -1.121|
| 25 | 4211     | -0.578| 3276     | 0.112 | 4030     | -0.01 | 3810     | -0.298|
| 26 | 4291     | -0.66 | 3432     | -0.378| 4211     | -0.367| 3997     | -0.069|
| 27 | 4328     | -0.082| 3478     | -0.029| 4291     | -0.88 | 4054     | -0.106|
| 28 | 4399     | -0.092| 3771     | -0.791| 4399     | -0.045| 4099     | -0.438|
| 29 | 4697     | 0.126 | 3810     | -0.131| 4438     | -0.057| 4219     | 1.097 |
| 30 | 4847     | 0.601 | 3997     | -0.2  | 4697     | 0.099 | 4355     | -0.085|
| 31 | 4951     | 0.019 | 4054     | -0.119| 4847     | 0.298 | 4890     | 0.071 |
| 32 | 5039     | 0.083 | 4099     | -0.087| 4951     | 0.019 | 4927     | -0.16 |
| 33 | 5230     | -0.024| 4219     | 1.208 | 5039     | 0.08  | 4957     | 0.041 |
| 34 | 5361     | -0.169| 4263     | -0.103| 5361     | -0.365| 5077     | -0.074|
| 35 | 5376     | 0.034 | 4355     | -0.186| 5466     | -0.643| 5145     | -1.21 |
| 36 | 5466     | -0.73 | 4504     | -0.028| 5833     | 0.054 | 5323     | -0.015|
| 37 | 5657     | -0.034| 4553     | -0.178| 6049     | -0.278| 5372     | -0.031|
| 38 | 6041     | 0.082 | 4927     | -0.16 | 6162     | 0.054 | 5680     | 0.159 |
| 39 | 6049     | -0.282| 4957     | 0.07  | 6308     | 0.019 | 5726     | 0.662 |
| 40 | 6162     | 0.055 | 5077     | -0.235| 6477     | -0.092| 5781     | 0.202 |
| 41 | 6169     | 0.04  | 5145     | -1.223| 6854     | -0.878| 6067     | -0.158|
| 42 | 6308     | 0.034 | 5323     | -0.406| 6886     | 0.391 | 6258     | -0.158|
| 43 | 6477     | -0.12 | 5372     | -0.097| 6383     | 0.029 |
| 44 | 6854     | -0.822| 5680     | 0.159 | 6422     | -0.48 |
| 45 | 6886     | 0.495 | 5726     | 1.119 | 6541     | -0.202|
| 46 | 5781     | 0.443 | 6680     | -0.904|          |       |          |       |

(Continued)
adopt the model (8), where misclassification probabilities $\pi_{10}$ and $\pi_{01}$ are formulated by logistic regression models. On the other hand, to generate error-prone predictors $X_i$, we adopt the model (7) with $\Sigma$ being specified as a diagonal matrix and diagonal entries are commonly specified as $\sigma_j^2 = 0.15$, $0.5$ or $0.75$.

To see the data generation in details, we demonstrate the following code. We first specify the generation of predictors:

```python
X = []
for i in range(100):
    X.append(np.random.normal(0, 1, 100))
X = np.array(X)
```

Next, we specify the sample size and $\beta_0$, and take $p = 1000$ and $\sigma^2 = 0.15$ as an example. Based on those information, we employ the function `ME_Generate` to generate error-prone data, where `data` represents the artificial data from the output of the function `ME_Generate` and `pr` represents two misclassification probabilities.

```python
n = 100
beta = [1] * 3 + [0] * 997
cov = np.identity(1000) * 0.15
gamma = [[1], [0] * 1000, [1], [0] * 1000]
ME = ME_Generate(n, beta, cov, X, gamma)
data = ME[1]
pr = ME[0]
```

Given the generated data, we define the response $y$ and predictors $x$. To implement the BOOME method, we specify iteration number, values of $\tau$ and $\eta$ to be $\text{ite} = 1000$, $\text{thres} = 0.9$, and $\text{lr} = 0.00001$, respectively. We now implement the function `LR_Boost` to examine the logistic regression model with measurement error in responses.
| #  | Setting 1 | Setting 2 | Setting 3 | Setting 4 |
|----|----------|----------|----------|----------|
| 1  | 134      | -0.103   | 7        | -0.442   | 4        | -1.633   |
| 2  | 263      | 0.020    | 7    | 0.081    | 412      | -0.442   | 7        | 1.019    |
| 3  | 412      | -0.101   | 183     | -0.054   | 635      | -0.389   | 83       | 0.137    |
| 4  | 738      | 0.035    | 268     | -0.408   | 738      | 0.377    | 183      | -0.120   |
| 5  | 758      | -0.013   | 617     | -0.040   | 760      | 0.271    | 225      | 0.054    |
| 6  | 760      | 0.433    | 727     | -0.079   | 804      | -1.178   | 731      | -0.047   |
| 7  | 804      | -0.432   | 731     | -0.499   | 1144     | -0.130   | 864      | 0.037    |
| 8  | 1144     | -0.013   | 917     | 0.062    | 1207     | -0.123   | 1108     | 1.407    |
| 9  | 1204     | -0.053   | 1058    | 0.024    | 1239     | -0.145   | 1148     | 1.343    |
| 10 | 1207     | -0.065   | 1089    | -0.048   | 1260     | 0.957    | 1307     | 0.233    |
| 11 | 1239     | -0.072   | 1148    | 0.256    | 1685     | -0.080   | 1357     | -0.062   |
| 12 | 1260     | 0.168    | 1307    | 0.309    | 1834     | 1.297    | 1402     | -0.028   |
| 13 | 1268     | -0.022   | 1326    | -0.053   | 1882     | 1.076    | 1687     | -0.787   |
| 14 | 1381     | -0.029   | 1565    | 0.047    | 1928     | -0.980   | 1716     | -0.038   |
| 15 | 1497     | 0.014    | 1588    | -0.067   | 1997     | 0.069    | 1734     | -0.056   |
| 16 | 1669     | 0.059    | 1687    | -0.093   | 2288     | 1.473    | 1854     | 0.320    |
| 17 | 1685     | -0.066   | 1708    | -0.134   | 2354     | -0.081   | 1858     | 0.031    |
| 18 | 1834     | 0.464    | 1858    | 0.424    | 2737     | 0.045    | 1906     | -0.155   |
| 19 | 1882     | 0.433    | 1944    | 0.078    | 2797     | 0.042    | 1915     | 0.132    |
| 20 | 1928     | -0.254   | 1987    | 0.327    | 2903     | 0.075    | 1987     | 0.103    |
| 21 | 1941     | 0.057    | 2039    | 0.375    | 3183     | 0.055    | 2039     | 0.918    |
| 22 | 2061     | -0.069   | 2057    | -0.098   | 3252     | 1.032    | 2161     | -0.259   |
| 23 | 2288     | 0.460    | 2059    | -0.026   | 3847     | 1.527    | 2240     | -0.276   |
| 24 | 2426     | 0.052    | 2301    | 0.028    | 4211     | -0.280   | 2373     | 0.194    |
| 25 | 2737     | 0.051    | 2377    | 0.696    | 4291     | -0.987   | 2377     | 0.077    |
| 26 | 2797     | 0.090    | 2460    | 0.235    | 4438     | -0.149   | 2443     | -0.018   |
| 27 | 3144     | -0.066   | 2534    | -0.491   | 4697     | 0.125    | 2469     | -0.070   |
| 28 | 3183     | 0.012    | 2682    | 0.070    | 4847     | 0.081    | 2661     | -0.075   |
| 29 | 3252     | 0.465    | 2863    | -0.077   | 5130     | -0.115   | 2764     | -0.030   |
| 30 | 3611     | -0.016   | 2941    | 0.242    | 5140     | -0.079   | 2809     | 0.078    |
| 31 | 3714     | 0.026    | 2984    | -0.024   | 5361     | -0.585   | 2941     | 0.050    |
| 32 | 3847     | 0.544    | 3016    | 0.031    | 5466     | -0.598   | 3016     | 0.274    |
| 33 | 3932     | -0.010   | 3205    | -0.129   | 5833     | 0.215    | 3424     | -0.093   |
| 34 | 4050     | -0.068   | 3276    | 0.140    | 6049     | -0.089   | 3446     | 0.051    |
| 35 | 4211     | -0.138   | 3432    | -0.026   | 6854     | 1.044    | 3478     | -0.145   |
| 36 | 4291     | -0.152   | 3771    | -0.229   | 6886     | 0.100    | 3493     | -0.408   |
| 37 | 4399     | -0.033   | 3810    | -0.026   | 3494     | 0.084    |
| 38 | 4697     | 0.089    | 3814    | 0.067    | 3572     | -0.163   |
| 39 | 4754     | 0.054    | 3997    | -0.121   | 3697     | -0.038   |
| 40 | 4847     | 0.130    | 4054    | -0.177   | 3752     | -0.030   |
| 41 | 4951     | 0.031    | 4219    | 0.477    | 3771     | -0.114   |
| 42 | 4955     | -0.011   | 4355    | -0.074   | 3838     | -0.032   |
| 43 | 5230     | -0.016   | 4536    | 0.136    | 3997     | -0.239   |
| 44 | 5290     | -0.016   | 4600    | -0.078   | 4099     | -0.202   |
| 45 | 5361     | -0.066   | 4927    | -0.255   | 4103     | -0.125   |
| 46 | 5376     | 0.087    | 4957    | 0.031    | 4219     | 0.631    |

(Continued)
and predictors corrected. Detailed implementation and partial output are given below:

```python
x = data.drop(['y'], axis=1)
y = data['y']
ite = 1000
thres = 0.9
lr = 0.00001
LRBoost(x, y, ite, thres, 1, 1, pr, lr, cov)
```

estimated coefficient: [1.02, 1.04, 0.99, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
predictors: [1, 2, 3]
number of predictors: 3
Table 3. Summary of common genes based on the logistic regression model and the probit model.

| #  | Setting 1 |   | Setting 2 |   | Setting 3 |   | Setting 4 |   |
|----|-----------|---|-----------|---|-----------|---|-----------|---|
|    | ID        | LR(1) | PM(2)     |   | ID        | LR(1) | PM(2)     |   |
| 1  | 134       | -0.208 | -0.103    |   | 412       | 0.410  | 0.081     |   |
| 2  | 412       | -0.606 | -0.101    |   | 760       | 0.271  | 0.433     |   |
| 3  | 760       | 0.271  | 0.433     |   | 804       | -1.271 | -0.432    |   |
| 4  | 804       | -1.271 | -0.432    |   | 1144      | -0.090 | -0.013    |   |
| 5  | 1144      | -0.090 | -0.013    |   | 1207      | -0.108 | -0.065    |   |
| 6  | 1207      | -0.108 | -0.065    |   | 1239      | -0.566 | -0.072    |   |
| 7  | 1239      | -0.566 | -0.072    |   | 1260      | 0.786  | 0.168     |   |
| 8  | 1260      | 0.786  | 0.168     |   | 1669      | 0.155  | 0.059     |   |
| 9  | 1669      | 0.155  | 0.059     |   | 1685      | -0.169 | -0.066    |   |
| 10 | 1685      | -0.169 | -0.066    |   | 1834      | 1.471  | 0.464     |   |
| 11 | 1834      | 1.471  | 0.464     |   | 1882      | 1.337  | 0.433     |   |
| 12 | 1882      | 1.337  | 0.433     |   | 1928      | -1.229 | -0.254    |   |
| 13 | 1928      | -1.229 | -0.254    |   | 2288      | 0.414  | 0.460     |   |
| 14 | 2288      | 0.414  | 0.460     |   | 2737      | 0.051  | 0.051     |   |
| 15 | 2737      | 0.051  | 0.051     |   | 2797      | 0.052  | 0.090     |   |
| 16 | 2797      | 0.052  | 0.090     |   | 3183      | 0.376  | 0.012     |   |
| 17 | 3183      | 0.376  | 0.012     |   | 3252      | 0.464  | 0.465     |   |
| 18 | 3252      | 0.464  | 0.465     |   | 3714      | 0.078  | 0.026     |   |
| 19 | 3714      | 0.078  | 0.026     |   | 3847      | 1.529  | 0.544     |   |
| 20 | 3847      | 1.529  | 0.544     |   | 4211      | 0.252  | 0.544     |   |
| 21 | 4211      | 0.252  | 0.544     |   | 4291      | -0.660 | -0.152    |   |
| 22 | 4291      | -0.660 | -0.152    |   | 4399      | -0.092 | -0.033    |   |
| 23 | 4399      | -0.092 | -0.033    |   | 4697      | 0.126  | 0.089     |   |
| 24 | 4697      | 0.126  | 0.089     |   | 4847      | 0.601  | 0.130     |   |
| 25 | 4847      | 0.601  | 0.130     |   | 4951      | 0.019  | 0.031     |   |
| 26 | 4951      | 0.019  | 0.031     |   | 5230      | -0.024 | -0.016    |   |
| 27 | 5230      | -0.024 | -0.016    |   | 5361      | -0.169 | -0.066    |   |
| 28 | 5361      | -0.169 | -0.066    |   | 5376      | 0.034  | 0.087     |   |
| 29 | 5376      | 0.034  | 0.087     |   | 5466      | -0.730 | -0.224    |   |
| 30 | 5466      | -0.730 | -0.224    |   | 6162      | 0.055  | 0.033     |   |
| 31 | 6162      | 0.055  | 0.033     |   | 6169      | 0.040  | 0.065     |   |
| 32 | 6169      | 0.040  | 0.065     |   | 6308      | 0.034  | 0.068     |   |
| 33 | 6308      | 0.034  | 0.068     |   | 6584      | -0.822 | -0.452    |   |
| 34 | 6584      | -0.822 | -0.452    |   | 6886      | 0.495  | 0.118     |   |
| 35 | 6886      | 0.495  | 0.118     |   | 6067      | -0.158 | -0.251    |   |
| 36 | 6067      | -0.158 | -0.251    |   | 6258      | -0.707 | -0.324    |   |
| 37 | 6258      | -0.707 | -0.324    |   | 6422      | -0.516 | -0.273    |   |
| 38 | 6422      | -0.516 | -0.273    |   | 6680      | -1.341 | -0.673    |   |
| 39 | 6680      | -1.341 | -0.673    |   | 6712      | -0.124 | -0.082    |   |
| 40 | 6712      | -0.124 | -0.082    |   | 6839      | 0.996  | 0.498     |   |
| 41 | 6839      | 0.996  | 0.498     |   | 6886      | 0.107  | 0.136     |   |
| 42 | 6886      | 0.107  | 0.136     |   | 6929      | -0.443 | -0.048    |   |
| 43 | 6929      | -0.443 | -0.048    |   | 7001      | 0.141  | 0.116     |   |
| 44 | 7001      | 0.141  | 0.116     |   | 7105      | -0.345 | -0.251    |   |

https://doi.org/10.1371/journal.pone.0276664.t003
For the comparison with the proposed method with correction of measurement error in responses and predictors, we examine naive analysis based on Settings 1-3 in Section 5.1. Detailed implementation and partial outputs are given below. In general, we find from outputs that the first three estimator based on the proposed method is close to the true value 1, and selected predictors are the same as the underlying true setting. On the other hand, without correcting measurement error effects, we observe from the below results that the first three estimators have larger biases and are far from the true value 1. Moreover, additional irrelevant predictors are falsely included.

```python
LRBoost(x, y, ite, thres, 0, 0, pr, lr, cov)
estimated coefficient: [-0.94, -1.13, -0.92, 0.0, 0.0, -0.20, 0.0, -0.16,
0.0, 0.0, -0.11, 0.0, 0.0, 0.14, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0]
predictors: [1, 2, 3, 6, 8, 11, 14, 129, 148, 229, 300, 346, 374, 421, 436,
453, 471, 480, 498, 520, 523, 543, 562, 589, 590, 628, 631, 634, 639, 640,
650, 668, 684, 704, 768, 774, 798, 851, 936, 965, 983, 985]
number of predictors: 42
```

```python
LRBoost(x, y, ite, thres, 1, 0, pr, lr, cov)
estimated coefficient: [-0.90, -1.20, -1.01, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
predictors: [1, 2, 3, 26, 33, 148, 229, 300, 346, 374, 421, 436, 453, 471,
480, 498, 518, 520, 562, 589, 590, 628, 631, 634, 639, 640, 650, 668, 753,
768, 774, 798, 851, 936, 965, 983, 985]
number of predictors: 37
```

```python
LRBoost(x, y, ite, thres, 0, 1, pr, lr, cov)
estimated coefficient: [0.94, 1.17, 0.92, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
predictors: [1, 2, 3, 26, 33, 129, 148, 229, 300, 346, 374, 421, 436, 453,
471, 480, 498, 520, 523, 543, 562, 589, 590, 628, 631, 634, 639, 640, 650,
668, 684, 704, 768, 774, 798, 851, 936, 965, 983, 985]
number of predictors: 40
```

In addition to the logistic regression models, we further examine the probit model based on four settings as described in Section 5.1. Specifically, we implement the function PM_Boost to construct the probit model and specify arguments (correct_X, correct_Y) to be (0,0), (0,1), (1,0), and (1,1) that reflect Settings 1-4 in Section 5.1, respectively. Detailed implementation and partial outputs are available below. Similar to the findings based on logistic regression models, we observe that the estimator with measurement error in responses and predictors corrected outperforms other scenarios because of smaller biases and precise variable selection. As expected, without suitable variable selection, the estimators induce tremendous biases and
some irrelevant predictors are included.

```python
PM_Boost(x, y, ite, thres, 1, 1, pr, lr, cov)
```

estimated coefficient: [1.02, 0.98, 0.97, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
predictors: [1, 2, 3]
number of predictors: 3

```python
PM_Boost(x, y, ite, thres, 0, 0, pr, lr, cov)
```

estimated coefficient: [1.08, 1.06, 1.28, −0.02, −0.03, −0.01, 0.0, −0.03, −0.02, 0.0, 0.10, 0.0, −0.02, 0.02, −0.045, 0.04, 0.0, 0.06, −0.02, 0.05, 0.01, 0.0, −0.01]
predictors: [1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 16, 17, 19, 20, 21, 22, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 41, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 59, 60, 62, 63, 65, 66, 67, 68, 69, 72, 73, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 97, 99, 100]
number of predictors: 82

```python
PM_Boost(x, y, ite, thres, 1, 0, pr, lr, cov)
```

estimated coefficient: [1.09, 1.07, 1.04, −0.03, −0.03, 0.0, 0.0, 0.0, −0.04, −0.02, 0.0, 0.12, 0.0, −0.02, 0.02, −0.06, 0.05, 0.0, 0.07, 0.0, 0.06, 0.0, −0.01, −0.01]
predictors: [1, 2, 3, 4, 5, 9, 10, 12, 14, 15, 16, 17, 19, 21, 23, 24, 25, 26, 27, 30, 31, 32, 33, 35, 37, 38, 39, 40, 41, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 59, 60, 62, 63, 65, 66, 67, 68, 69, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 99, 100]
number of predictors: 77

```python
PM_Boost(x, y, ite, thres, 0, 1, pr, lr, cov)
```

estimated coefficient: [0.91, 0.94, 0.92, 0.0, 0.0, 0.0, −0.02, 0.01, 0.0, 0.03, 0.0, 0.04, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
predictors: [1, 2, 3, 7, 8, 10, 12, 30, 31, 34, 35, 38, 43, 54, 62, 80, 92, 97]
number of predictors: 18
Finally, to precisely access the accuracy of the estimator, we use the $L_1$-norm and the $L_2$-norm, which are respectively defined as

\[ \| \Delta \|_1 = \sum_{i=1}^{p} |\hat{\beta}_i - \beta_{0,i}| \]

and

\[ \| \Delta \|_2 = \sum_{i=1}^{p} (\hat{\beta}_i - \beta_{0,i})^2, \]

where $\Delta = \hat{\beta} - \beta_0$, $\hat{\beta}_i$ and $\beta_{0,i}$ are the $i$th entry of $\hat{\beta}$ and $\beta_0$, respectively. To access the performance of variable selection, we examine specificity (SPE) and sensitivity (SEN), which are respectively defined as

\[ \text{SPE} = \frac{\text{the number of predictors that are correctly included}}{\text{the number of truly informative predictors}} \]

and

\[ \text{SEN} = \frac{\text{the number of predictors that are correctly excluded}}{\text{the number of the truly unimportant predictors}}. \]

Numerical results under all settings described above are reported in Table 4. We can observe that biases in the $L_1$ and $L_2$-norms are increasing when the magnitude of measurement error $\sigma^2$ and dimension $p$ become large. As expected, when measurement error in responses and predictors are corrected (Setting 4), the biases are the smallest and SPE as well as SEN are the largest among all settings, which verify that the proposed method is valid to handle measurement error regardless of specification of regression models. On the other hand, without correcting measurement error effects, we find that the naive methods (Settings 1-3) produce significant biases and low values of SPE and SEN, indicating the worse performance of variable selection. In particular, if measurement error in responses and predictors are not corrected, as shown in Setting 1, we have the worst estimation results. Compared with Settings 2 and 3, it is interesting to see that the biases under Setting 2 are greater than those based on Setting 3, and values of SPE and SEN obtained by Setting 2 are smaller than those based on Setting 3. It implies that ignoring measurement error in predictors would incur severe biases and would be worse than ignoring measurement error effects occurred in responses.

6 Discussion

In this paper, we introduce the Python package BOOME that aims to address ultrahigh-dimensional data subject to measurement error in responses and predictors. Unlike existing packages that deal with either variable selection or measurement error but not both, our package can handle variable selection and correct measurement error effects to both responses and predictors simultaneously. In addition, the computational time is fairly fast and arguments are flexible for public use. In applications, sometimes variables in datasets can be shown to be free of measurement error and can be precisely measured, such as age or gender. The package BOOME is still flexible to handle those scenarios. For example, if researchers believe that predictors in their datasets are free of measurement error, then they can adopt Setting 2 in Section 5.1 by employing corrected responses and precisely measured predictors; if only predictors are shown to have measurement error, then one can adopt Setting 3 in Section 5.1 by implementing corrected predictors and precisely measured responses.
Table 4. Simulation results for two regression models with $n = 100$.

| Model  | $p$ | $\sigma^2$ | Setting | $\|\Delta\|_1$ | $\|\Delta\|_2$ | SPE | SEN |
|--------|-----|------------|---------|-----------------|-----------------|-----|-----|
| LR (1) | 1000 | 0.15 | 1       | 4.633           | 1.807           | 0.900 | 0.213 |
|        |      |       | 2       | 4.581           | 1.812           | 0.900 | 0.326 |
|        |      |       | 3       | 0.358           | 0.184           | 1.000 | 0.945 |
|        |      |       | 4       | 0.228           | 0.136           | 1.000 | 0.979 |
|        | 0.50 |       | 1       | 4.400           | 1.780           | 0.667 | 0.230 |
|        |      |       | 2       | 4.286           | 1.791           | 0.567 | 0.483 |
|        |      |       | 3       | 1.758           | 0.606           | 1.000 | 0.583 |
|        |      |       | 4       | 0.917           | 0.410           | 1.000 | 0.853 |
|        | 0.75 |       | 1       | 4.244           | 1.772           | 0.767 | 0.249 |
|        |      |       | 2       | 4.213           | 1.786           | 0.600 | 0.496 |
|        |      |       | 3       | 2.458           | 0.789           | 1.000 | 0.417 |
|        |      |       | 4       | 1.158           | 0.526           | 1.000 | 0.847 |
|        | 5000 | 0.15 | 1       | 6.181           | 1.821           | 0.533 | 0.245 |
|        |      |       | 2       | 5.794           | 1.818           | 0.467 | 0.400 |
|        |      |       | 3       | 0.412           | 0.197           | 1.000 | 0.957 |
|        |      |       | 4       | 0.256           | 0.147           | 1.000 | 0.981 |
|        | 0.50 |       | 1       | 6.145           | 1.803           | 0.633 | 0.201 |
|        |      |       | 2       | 5.485           | 1.802           | 0.233 | 0.458 |
|        |      |       | 3       | 2.490           | 0.641           | 1.000 | 0.602 |
|        |      |       | 4       | 1.082           | 0.427           | 1.000 | 0.889 |
|        | 0.75 |       | 1       | 5.622           | 1.797           | 0.300 | 0.218 |
|        |      |       | 2       | 5.122           | 1.806           | 0.533 | 0.529 |
|        |      |       | 3       | 3.358           | 0.813           | 1.000 | 0.452 |
|        |      |       | 4       | 1.445           | 0.581           | 1.000 | 0.862 |
| PM (2) | 1000 | 0.15 | 1       | 9.367           | 1.855           | 0.767 | 0.297 |
|        |      |       | 2       | 7.890           | 1.829           | 0.567 | 0.459 |
|        |      |       | 3       | 0.424           | 0.168           | 1.000 | 0.964 |
|        |      |       | 4       | 0.234           | 0.124           | 1.000 | 0.988 |
|        | 0.50 |       | 1       | 9.345           | 1.843           | 0.767 | 0.224 |
|        |      |       | 2       | 7.168           | 1.821           | 0.267 | 0.545 |
|        |      |       | 3       | 3.203           | 0.664           | 1.000 | 0.696 |
|        |      |       | 4       | 1.598           | 0.507           | 1.000 | 0.887 |
|        | 0.75 |       | 1       | 9.027           | 1.861           | 0.900 | 0.247 |
|        |      |       | 2       | 7.105           | 1.860           | 0.567 | 0.555 |
|        |      |       | 3       | 4.592           | 0.822           | 1.000 | 0.548 |
|        |      |       | 4       | 2.301           | 0.654           | 1.000 | 0.837 |
|        | 5000 | 0.15 | 1       | 16.555          | 1.994           | 0.400 | 0.452 |
|        |      |       | 2       | 12.462          | 1.908           | 0.367 | 0.593 |
|        |      |       | 3       | 0.419           | 0.145           | 1.000 | 0.979 |
|        |      |       | 4       | 0.262           | 0.132           | 1.000 | 0.993 |
|        | 0.50 |       | 1       | 17.391          | 2.031           | 0.667 | 0.233 |
|        |      |       | 2       | 13.177          | 1.943           | 0.233 | 0.587 |
|        |      |       | 3       | 1.244           | 0.204           | 1.000 | 0.819 |
|        |      |       | 4       | 0.333           | 0.066           | 1.000 | 0.930 |
|        | 0.75 |       | 1       | 17.903          | 2.043           | 0.633 | 0.419 |
|        |      |       | 2       | 13.289          | 1.948           | 0.500 | 0.574 |
|        |      |       | 3       | 1.445           | 0.194           | 1.000 | 0.738 |
|        |      |       | 4       | 0.707           | 0.098           | 1.000 | 0.789 |

https://doi.org/10.1371/journal.pone.0276664.t004
There are several possible extensions based on the current developments. First, in addition to continuous or binary random variables, categorical or counted data are frequently adopted in the framework of bioinformatics, such as RNA sequence or GWAS data, and they might be subject to mismeasurement error. Therefore, it is important to propose a valid approach to adjust for measurement error effects to those data. In addition, our current approach focuses on parametric logistic regression or probit models. To provide general formulations, it is interesting to extend the BOOME method to nonparametric models or semi-parametric models. In the current development, our attention primarily focuses on variable selection for high-dimensional data subject to measurement error. In supervising learning, examining the performance of classification and prediction is a crucial concern. Provided that additional information, such as validation samples, is available, it is interesting to adopt selected predictors and adjustments of measurement error from the BOOME method to define a general model of measurement heterogeneity and develop several measures (e.g., C-statistic or Brier score) to assess the predictive performance (e.g., [30]). In addition, since responses are subject to measurement error as well, it deserves careful exploration to handle measurement error in responses when doing prediction.

Finally, as commented by a referee, dimension reduction techniques, such as principal component analysis (PCA) or factor analysis, can be valid tools to reduce dimension from ultrahigh-dimensional predictors. However, there are two main issues in the current development. First, the purpose in this study is to detect informative predictors and exclude irrelevant ones, while dimension reduction techniques aim to reduce dimension through linear combinations of high-dimensional predictors. Second, when the predictors are subject to measurement error, the BOOME package is able to address measurement error effects and correctly retain important predictors, while correction of measurement error effects for dimension reduction techniques is not explored, especially when the response is contaminated with measurement error as well. Undoubtedly, it is an interesting perspective to handle ultrahigh-dimensional data and deserves careful exploration in the future research.

Acknowledgments

The author would like to thank his master student, QinYing OuYang, for assistance in developing the package and summarizing the results of data analysis, and thank Lingyu Cai for revising Python code, helpful language editing, grammar revision, and proofreading. The author also thanks two referees for their useful comments to significantly improve the presentation of the initial manuscript.

Author Contributions

Conceptualization: Li-Pang Chen.
Formal analysis: Li-Pang Chen.
Funding acquisition: Li-Pang Chen.
Methodology: Li-Pang Chen.
Software: Li-Pang Chen.
Writing – original draft: Li-Pang Chen.

References

1. Guyon I., Weston J., Barnhill S., Vapnik V. (2002). Gene selection for cancer classification using support vector machines. Machine Learning, 46, 389–422. https://doi.org/10.1023/A:1012487302797
2. He, W., Yi, G. Y., and Chen, L.-P. (2019). Support vector machine with component graphical structure incorporated. Proceedings, Machine Learning and Data Mining in Pattern Recognition, 15th International Conference on Machine Learning and Data Mining, MLDM 2019, vol.II, 557–570.

3. Tibshirani R. (1996). Regression shrinkage and selection via the LASSO. Journal of the American Statistical Association, 101, 1418–1429. https://doi.org/10.1198/0162145960000000735

4. Zou H. (2006). The adaptive Lasso and its oracle properties. Journal of the American Statistical Association, 101, 1418–1429. https://doi.org/10.1198/0162145060000000735

5. Zou H. and Hastie T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67, 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

6. Chen L.-P. and Yi G. Y. (2021a). Analysis of noisy survival data with graphical proportional hazards measurement error models. Biometrics, 77, 956–969. https://doi.org/10.1111/biom.13331

7. Chen L.-P. and Yi G. Y. (2022a). De-noising analysis of noisy data with graphical models. Electronic Journal of Statistics, 16, 3861–3909. https://doi.org/10.1214/22-EJS2028

8. Chen L.-P. and Yi G. Y. (2022b). Sufficient dimension reduction for survival data analysis with error-prone variables. Electronic Journal of Statistics, 16, 2082–2123. https://doi.org/10.1214/22-EJS2028

9. Huang F., Guang P., Li F., Liu X., Zhang W., and Huang W. (2020). AML, ALL, and CML classification and diagnosis based on bone marrow cell morphology combined with convolutional neural network. Medicine, 99:45, 1–8. https://doi.org/10.1097/MD.0000000000003154

10. Carroll R. J., Ruppert D., Stefanski L. A., and Crainiceanu C. M. (2006). Measurement Error in Nonlinear Models. Chapman and Hall.

11. Chen L.-P. and Yi G. Y. (2020). Model selection and model averaging for analysis of truncated and censored data with measurement error. Electronic Journal of Statistics, 14, 4054–4103. https://doi.org/10.1214/20-EJS1762

12. Chen L.-P. (2020). Variable selection and estimation for the additive hazards model subject to left-truncation, right-censoring and measurement error in covariates. Journal of Statistical Computation and Simulation, 90, 3261–3300. https://doi.org/10.1080/00949655.2020.1800705

13. Yi G. Y. (2017). Statistical Analysis with Measurement Error and Misclassification: Strategy, Method and Application. Springer.

14. Carroll R. J., Spiegelman C. H., Gordon Lan K. K., Bailey K. T., and Abbott R. D. (1984). On errors-in-variables for binary regression models. Biometrika, 71, 19–25. https://doi.org/10.1093/biomet/71.1.19

15. Chen L.-P. and Yi G. Y. (2021b). Semiparametric methods for left-truncated and right-censored survival data with covariate measurement error. Annals of the Institute of Statistical Mathematics, 73, 481–517. https://doi.org/10.1007/s10463-020-00755-2

16. Roy S., Banerjee T., and Maiti T. (2005). Measurement error model for misclassified binary responses. Statistics in Medicine, 24, 269–283. https://doi.org/10.1002/sim.1886 PMID: 15546132

17. Schafer D. W. (1993). Analysis for probit regression with measurement errors. Biometrika, 80, 899–904. https://doi.org/10.1093/biomet/80.4.899

18. Stefanski L. A., and Carroll R. J. (1987). Conditional scores and optimal scores for generalized linear measurement error models. Biometrika, 74, 703–716. https://doi.org/10.2307/2336464

19. Brown B., Miller C. J., and Wolfson J. (2017). ThrEEBoost: Thresholded boosting for variable selection and prediction via estimating equations. Journal of Computational and Graphical Statistics, 26, 579–588. https://doi.org/10.1080/10618600.2016.1247005

20. Hastie T., Tibshirani R., and Friedman J. (2008). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.

21. Serensen Ø, Hellton K. H., Frigessi A., and Thoresen M. (2018). Covariate selection in high-dimensional generalized linear models with measurement error. Journal of Computational and Graphical Statistics, 27, 739–749. https://doi.org/10.1080/10618600.2018.1425626

22. Ma Y. and Li R. (2010). Variable selection in measurement error models. Bernoulli, 16, 273–300. https://doi.org/10.3150/09-bej205 PMID: 2020920

23. Brown B., Weaver T., and Wolfson J. (2019). MEBoost: Variable selection in the presence of measurement error. Statistics in Medicine, 38, 2705–2718. https://doi.org/10.1002/sim.8130 PMID: 30856279

24. Friedman, J., Hastie, T., Tibshirani, R., Narasimhan, B., Tay, K., Simon, N., et al. (2022). glmnet: Lasso and Elastic-net regularized generalized linear models. R package version 4.1-4. https://CRAN.R-project.org/package=glmnet

25. Fan J. and Li J. (2008). Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70, 849–911. https://doi.org/10.1111/j.1467-9868.2008.00674.x
26. Krishnan, S. (2019). xverse. Python package version 1.0.5. https://pypi.org/project/xverse/#description
27. Bartoszek, K. (2019). GLSME: Generalized least squares with measurement error. R package version 1.0.5. https://CRAN.R-project.org/package=GLSME
28. Nab L., van Smeden M., Keogh R. H. and Groenwol R.H.H. (2021). Mecor: An R package for measurement error correction in linear regression models with a continuous outcome. Computer Methods and Programs in Biomedicine, 208, 106238. https://doi.org/10.1016/j.cmpb.2021.106238 PMID: 34311414
29. Wolfson J. (2011). EEBOOST: a general method for prediction and variable selection based on estimating equation. Journal of the American Statistical Association, 106, 296–305. https://doi.org/10.1198/jasa.2011.tm10098
30. Luijken K., Groenwold R.H.H., Van Calster B., Steyerberg E.W., and van Smeden M. (2019). Impact of predictor measurement heterogeneity across settings on the performance of prediction models: A measurement error perspective. Statistics in Medicine, 38, 3444–3459. https://doi.org/10.1002/sim.8183 PMID: 31148207