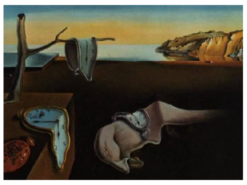


Journal of Statistical Software

December 2025, Volume 115, Issue 8. *doi: 10.18637/jss.v115.i08*

SMLE: An R Package for Joint Feature Screening in Ultrahigh-Dimensional GLMs

Qianxiang Zang
University of Ottawa

Chen Xu
Pengcheng Laboratory &
Xi'an Jiaotong University

Kelly Burkett
University of Ottawa

Abstract

Sparsity-restricted maximum likelihood estimation (SMLE) has received considerable attention for feature screening in ultrahigh-dimensional regression. SMLE is a computationally convenient method that naturally incorporates the joint effects among features in the screening process. We develop a publicly available R package **SMLE**, which provides a user-friendly environment to carry out the SMLE method in generalized linear models. In particular, the package includes functions to conduct SMLE-screening and the related post-screening selection with popular selection criteria such as AIC and (extended) BIC. The package gives users the flexibility in controlling a series of screening parameters and accommodates both numerical and categorical feature input. The usage of **SMLE** is illustrated on extensive numerical examples, where the promising performance of the package is well observed.

Keywords: EBIC, generalized linear models, iterative hard-thresholding, joint feature screening, ultrahigh-dimensional data.

1. Introduction

In modern scientific research, it is common to encounter ultrahigh-dimensional datasets with a huge number of features. For example, geneticists often need to measure thousands to hundreds of thousands of genes in the hope of discovering those that influence an observable trait; an Internet firewall may scan millions of keywords on data packets in order to determine their security risk. While ultrahigh-dimensional data bring rich resources to explore many unknown areas, they pose simultaneous challenges of computational cost, statistical accuracy, and algorithmic stability for classic statistical methods (Fan, Samworth, and Wu 2009).

When the number of features is huge, it is often reasonable to assume that only a handful of them are relevant to the analysis. In a regression setting, this amounts to assuming that most predictors in an ultrahigh-dimensional model have no effect on the response (i.e., the regression coefficient is zero). With this sparsity assumption, one natural strategy is to screen most irrelevant features out before a more elaborate analysis is conducted. This pre-processing procedure is referred to as feature screening. With dimensionality reduced from high to low, analytical difficulties can be reduced drastically.

In recent years, much research has been done on feature screening. Fan and Lv (2008) proposed to screen features based on their marginal Pearson correlations with the response; they referred to this procedure as sure independence screening (SIS) and justified its theoretical effectiveness for linear models. Fan and Song (2010) extended SIS to generalized linear models (GLMs). In the same spirit, Zhu, Li, Li, and Zhu (2011) proposed a sure independent ranking and screening (SIRS) based on the conditional distribution of the response given each feature. Li, Zhong, and Zhu (2012) developed a model-free sure independence screening based on the distance correlation (DC-SIS). Wu and Yin (2015) proposed a distribution function screening by testing the independence between the response and each feature. Zhou, Zhu, Xu, and Li (2020) proposed a robust screening method for features containing extreme values. Li, Li, Xia, and Xu (2020) proposed a distributed screening framework for the divide-and-conquer setup. The list above is certainly far from complete; readers may refer to Liu, Zhong, and Li (2015) for a selective overview on feature screening.

In the literature, most screening methods are developed based on the marginal effects of features on the response. Despite the convenience of implementation, these methods are often found to be unreliable in practice, as the joint effects among features are ignored. Features with significant joint effects but showing weak marginal effects are likely to be wrongly screened out. To tackle this issue, Fan and Lv (2008) suggested applying SIS iteratively (ISIS) with a smaller number of features retained in each round. Wang (2009) suggested using classic forward regression for screening purposes. These strategies help to incorporate some feature joint effects in the screening process. However, they are usually at a high computational cost, which can be unfavorable in many applications.

Starting from a different angle, Xu and Chen (2014) proposed a joint feature screening method via the sparsity restricted maximum likelihood estimator (SMLE). With a L_0 penalty specifying the number of features allowed in the model, the method attempts to roughly estimate a handful of the most significant coefficients from the full model while setting all other coefficients to zero. Since the estimation is carried out on the full model, the resulting sparse estimator readily serves as a feature screener, which naturally takes the joint effects among features into account. The SMLE method can be efficiently implemented by an iterative hard-thresholding algorithm (IHT), which does not involve complex numerical operations such as matrix inversion. Each IHT iteration increases the value of the sparsity-constrained joint likelihood and thereby provides an improved sparse solution, which eventually leads to a reliable screening result. Xu and Chen (2014) further justified the sure screening property of SMLE in ultrahigh-dimensional GLMs and proved the convergence of the IHT algorithm. SMLE has been demonstrated to be an effective tool for feature screening; the method has attracted considerable attention in the literature (Yang, Yu, Li, and Buu 2016; Yang, Hou, Wang, and Sun 2018; Qu, Hao, and Sun 2022).

In this paper, we provide a publicly available R package **SMLE**, which gives a user-friendly environment to carry out SMLE in ultrahigh-dimensional GLMs including linear, logistic, and

Poisson models. The package makes use of the `crossprod()` function to handle ultrahigh-dimensional matrix products and includes a well-tuned main function to efficiently conduct SMLE-screening based on the IHT algorithm. With the package, we are able to repeat the same numerical experiments in Xu and Chen (2014) at a significantly reduced time cost in comparison with the original code provided by the authors. In the package, we extend SMLE by permitting both numerical and categorical features in the screening, where the categorical features can be automatically identified and encoded by a user-selected method. Moreover, combined with popular selection criteria such as AIC or (extended) BIC, we propose a SMLE-based selection method, which helps to further identify the relevant features after screening. This post-screening selection method can be conveniently conducted within the main screening function or can be used independently on a user-supplied dataset. We illustrate the usage of **SMLE** via extensive numerical examples. The promising performance of the package is observed in comparison with **SIS** (Saldana and Feng 2018) and **VariableScreening** (Li, Huang, and Dziak 2022), which are the standard R packages for feature screening.

The rest of the paper is organized as follows. In Section 2, we give a brief overview of SMLE and the IHT algorithm. In Section 3, we discuss implementation details of the **SMLE** package. In Section 4, we illustrate the usage of the package using extensive numerical examples and compare its performance with several existing R packages. We conclude the paper in Section 5 with a few remarks.

2. The SMLE method

2.1. Notation and problem setup

Suppose the data $\{(y_i, \boldsymbol{x}_i), i = 1, \ldots, n\}$ are collected independently from $(Y, \boldsymbol{x})$, where Y is a response variable and $\boldsymbol{x} = (x_1, \ldots, x_p)$ is a p-dimensional covariate (feature) vector. We postulate a GLM between Y and $\boldsymbol{x}$ as follows. Conditioning on $\boldsymbol{x}$, the distribution of Y is assumed to belong to an exponential family taking the form

$$f(y; \theta) = \exp(\theta y - b(\theta) + c(y)),$$

where θ is the natural or canonical parameter and $b(.)$, $c(.)$ are two known functions. Under the canonical link, $\boldsymbol{x}$ influences Y in the form of a linear combination

$$\theta = \boldsymbol{x}\boldsymbol{\beta},$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$ is a p-dimensional regression coefficient. Popular GLMs with canonical links include the normal linear model, the logistic model, and the log-linear Poisson model (Nelder and Wedderburn 1972).

Under the GLM framework, the effect of each feature x_j on the response Y is characterized by the size of the corresponding regression coefficient β_j. In applications, when the number of features p is large, it is often believed that only a small number of the features in $\boldsymbol{x}$ contribute to the variation in Y, which leads to an idealistic assumption that $\boldsymbol{\beta}$ contains many zero elements. With this sparsity assumption, only features with non-zero coefficients are considered to be relevant in explaining the variation of Y. The goal of feature screening is to identify and remove most of the irrelevant features, so that a more elaborate analysis can be conducted only on the features most likely to be related to Y.

2.2. The SMLE-screening and IHT algorithm

The idea of SMLE is simple. When p is overly large and most model coefficients are assumed to be zero, it is probably not wise to estimate the entire $\boldsymbol{\beta}$ from scratch. Instead, it is reasonable to consider just estimating some of the coefficients while setting the others to zero from the beginning. This leads to a sparsity-restricted estimation, which readily serves for feature screening.

Specifically, under the GLM given in Section 2.1, the log-likelihood function of $\boldsymbol{\beta}$ is given by

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n}[y_i \cdot \boldsymbol{x}_i\boldsymbol{\beta} - b(\boldsymbol{x}_i\boldsymbol{\beta})].$$

With a user-specified sparsity $k < p$, the SMLE estimator is defined by

$$\hat{\boldsymbol{\beta}}_k = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, l(\boldsymbol{\beta}) \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 \leq k, \tag{1}$$

where $\|.\|_0$ is the vector L_0 norm indicating the number of non-zero elements in that vector. Clearly, $\hat{\boldsymbol{\beta}}_k$ is designed to set all but the most significant k coefficients to be zero; this amounts to identifying k important features supported most by the joint likelihood. When p is large and k is chosen to be much smaller than p, $\hat{\boldsymbol{\beta}}_k$ can be viewed as a feature screener, which naturally takes the joint effects among features into account. The idea of SMLE has similarities with the use of L_0-regularized techniques in image processing, where sparsity-constrained least-squares methods are frequently used to construct sparse representations for high-resolution images (Donoho 2006; Blumensath and Davies 2009).

While SMLE is conceptually simple, carrying out problem (1) can be numerically challenging, as it is a high-dimensional combinatorial optimization. However, since our goal is feature screening, finding the global solution to (1) is not a major concern. In fact, it would suffice if we can obtain a good local solution, which helps to retain all relevant features.

In this spirit, Xu and Chen (2014) proposed an iterative hard-thresholding algorithm (IHT) to approximately solve (1). The idea is as follows. With a $\boldsymbol{\gamma}$ close to $\boldsymbol{\beta}$, one can approximate $l(\boldsymbol{\beta})$ by a surrogate function

$$h(\boldsymbol{\beta}, \boldsymbol{\gamma}) = l(\boldsymbol{\gamma}) + (\boldsymbol{\beta} - \boldsymbol{\gamma})^\top l'(\boldsymbol{\gamma}) - (u/2)\|\boldsymbol{\beta} - \boldsymbol{\gamma}\|_2^2, \tag{2}$$

where $l'(\boldsymbol{\gamma}) = \partial l(\boldsymbol{\gamma})/\partial\boldsymbol{\gamma}$, $\|.\|_2$ indicates the L_2 (Euclidean) norm, and $u > 0$ is a scale parameter. The first two terms in (2) match the Taylor's expansion of $l(\boldsymbol{\beta})$ at $\boldsymbol{\beta} = \boldsymbol{\gamma}$, and the third term is introduced as a regularization term to enhance the convexity.

The reason for introducing $h(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is that it is separable in the components of $\boldsymbol{\beta}$ and thus serves as a surrogate of $l(\boldsymbol{\beta})$ to conveniently carry out the sparsity-restricted maximization over $\boldsymbol{\beta}$. Specifically, with an initial value $\boldsymbol{\beta}^{(0)}$, we can seek a local solution of (1) via the following iterative procedure.

$$\boldsymbol{\beta}^{(t+1)} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, h(\boldsymbol{\beta}, \boldsymbol{\beta}^{(t)}) \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 < k. \tag{3}$$

Let $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ and $\boldsymbol{X} = (\boldsymbol{x}_i^\top, \ldots, \boldsymbol{x}_n^\top)^\top$. Because of the additivity of $\boldsymbol{\beta}$ in h, the optimization in (3) takes a unified form

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\boldsymbol{\beta} - u^{-1}[u\boldsymbol{\beta}^{(t)} + \boldsymbol{X}^\top\boldsymbol{y} - \boldsymbol{X}^\top b'(\boldsymbol{X}\boldsymbol{\beta}^{(t)})]\right\|_2^2 \quad \text{subject to } \|\boldsymbol{\beta}\|_0 \leq k, \tag{4}$$

Algorithm 1 SMLE-screening via IHT

```
Require: Data $(\boldsymbol{y}, \boldsymbol{X})$, screening size $k$, initial $\boldsymbol{\beta}^{(0)}$, and decrease rate $\tau \in (0,1)$
  set $t = 1$ and $\boldsymbol{\beta}^{(t)} = H_k[\boldsymbol{\beta}^{(0)}]$
  initialize $u^{-1}$ based on $\boldsymbol{X}$ and $\boldsymbol{\beta}^{(0)}$
  repeat until stopping criterion is satisfied {
    set $\nu = u^{-1}$, $r = 0$
    repeat until $l(\tilde{\boldsymbol{\beta}}) \geq l(\boldsymbol{\beta}^{(t)})$ {
      compute (5) and (6): $\tilde{\boldsymbol{\beta}} = H_k[\boldsymbol{\beta}^{(t)} + \nu \boldsymbol{X}^\top[\boldsymbol{y} - b'(\boldsymbol{X}\boldsymbol{\beta}^{(t)})]]$
      $\nu = \tau\nu$
      $r = r + 1$
    }
    $\boldsymbol{\beta}^{(t+1)} = \tilde{\boldsymbol{\beta}}$
    u-search$^{(t)} = r$
    $t = t + 1$
  }
  Output: $\boldsymbol{\beta}^{(t)}$, the number of iterations $t$, the number of $u$-search tries in each iteration,
  and an index set of retained features $\hat{s} = \{1 \leq j \leq p : \beta_j^{(t)} \neq 0\}$
```

where b' is the derivative of the $b(.)$ function depending on the choice of GLM. Obviously, if (4) does not come with the sparsity constraint, its solution should be

$$\tilde{\boldsymbol{\beta}}^{(t)} = \boldsymbol{\beta}^{(t)} + u^{-1}\boldsymbol{X}^\top[\boldsymbol{y} - b'(\boldsymbol{X}\boldsymbol{\beta}^{(t)})], \tag{5}$$

which corresponds to a zero loss in the objective function. Thus, the constrained minimum of (4) is achieved by choosing the k largest (in absolute value) components of $\tilde{\boldsymbol{\beta}}^{(t)}$. Consequently, $\boldsymbol{\beta}^{(t+1)}$ in (3) has an explicit expression

$$\boldsymbol{\beta}^{(t+1)} = H_k[\tilde{\boldsymbol{\beta}}^{(t)}], \tag{6}$$

where $H_k[\boldsymbol{\beta}]$ is the hard-thresholding operator setting all but the k largest components in $|\boldsymbol{\beta}|$ to zero.

In IHT, u^{-1} serves as a step size, which controls the distance moved from $\boldsymbol{\beta}^{(t)}$ to $\boldsymbol{\beta}^{(t+1)}$. While a larger u^{-1} often helps to boost the iterations, the procedure may fail to converge when u^{-1} is overly large. Thus, to balance algorithm convergence and iteration efficiency, one may choose to adaptively tune u^{-1} at each step (called u-search). Xu and Chen (2014) proved that, when u^{-1} is small enough, the IHT procedure leads to a non-decreasing likelihood. In our package, we first initialize u^{-1} with a large value and then adaptively decrease its size by a factor of $\tau \in (0,1)$ until $l(\boldsymbol{\beta}^{(t+1)}) \geq l(\boldsymbol{\beta}^{(t)})$ is satisfied. This seems to be an effective way for achieving sufficiently fast convergence.

We summarize the SMLE-screening procedure via IHT in Algorithm 1. As can be seen from the algorithm summary, the procedure involves only simple numerical operations and adaptively tuning u^{-1} often requires only a few tries to succeed. At each iteration, the joint information carried in $\boldsymbol{X}$ is naturally accounted for as a basis for the next update. These merits make SMLE-screening attractive in ultrahigh-dimensional data analysis, where computational hurdles and complex data structures are often faced.

Algorithm 2 Post-screening selection with SMLE

```
Require: Data $(\boldsymbol{y}, \boldsymbol{X}_s)$, selection criterion, sparsity lower and upper bounds $k_{min}, k_{max}$.
  for $k$ from $k_{min}$ to $k_{max}$ {
    Obtain a sub-model $s_k$ by running Algorithm 1 with sparsity $k$ on $(\boldsymbol{y}, \boldsymbol{X}_s)$
    Evaluate $s_k$ by computing a score $C_k$ based on the input selection criterion
  }
  Output sub-model $s^* \in \{s_{k_{min}}, \ldots, s_{k_{max}}\}$ with the smallest evaluation score $C_k$
```

Xu and Chen (2014) showed that, with appropriate u^{-1} and $\boldsymbol{\beta}^{(0)}$, the IHT updates lead to a local maximum of problem (1), which provides an index set of k important features, $\hat{s}$, corresponding to the non-zero entries of $\boldsymbol{\beta}^{(t)}$. Based on $\hat{s}$, we then obtain a refined feature set from $\boldsymbol{X}$ for subsequent in-depth model fitting. Under some regularity conditions, $\hat{s}$ contains all the relevant features with probability tending to one even when $p \gg n$, and thus is consistent for feature screening (sure screening).

The IHT updates can be viewed as a member of the Majorize-Minimization algorithms. Its practical performance is affected by a series of implementation decisions such as the choice of initial value, stopping criterion, screening size k, and u-search. We address those algorithm details in Section 3.2.

2.3. Post-screening selection with SMLE

In Xu and Chen (2014), SMLE is mainly proposed for feature screening, the goal of which is to remove most irrelevant features before a more elaborate analysis. In practice, it is very likely that the feature set retained after screening still contains some irrelevant features. In principle, users can apply any well-developed selection method on the retained feature set to further identify relevant features.

In particular, one may further use the idea of SMLE to conduct post-screening selection. Specifically, assume that SMLE-screening was done and q features were retained; we obtain a refined $n \times q$ feature matrix $\boldsymbol{X}_s$. When q is moderate, we can conveniently obtain a series of sub-models by running Algorithm 1 on $(\boldsymbol{y}, \boldsymbol{X}_s)$ with sparsity k varying from $k_{min} \geq 1$ to $k_{max} \leq q$. A final sub-model can then be selected based an information criterion such as AIC (Akaike 1974), BIC (Schwarz 1978), and EBIC (Chen and Chen 2008). We summarize this post-screening selection method in Algorithm 2, which inherits all the numerical merits from Algorithm 1. In particular, the joint information in $\boldsymbol{X}_s$ is naturally accounted for in the selection process.

Technically, Algorithm 2 can be used directly on $\boldsymbol{X}$ without the need for feature screening. Its performance is actually quite impressive in our simulation studies. Nevertheless, when p is very large, we do recommend using Algorithm 2 only after Algorithm 1 for improved accuracy and stability.

3. Implementation details

The R package **SMLE** (Zang, Xu, and Burkett 2026) provides a set of functions for ultrahigh-dimensional feature screening under generalized linear models. **SMLE** can be installed from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=SMLE` using the following commands:

Function name	Description
`Gen_Data()`	Simulate ultra-high dimensional GLM data
`SMLE()`	Joint feature screening using the SMLE method
`smle_select()`	Post-screening feature selection
`predict()`	Fitted or predicted values under the final model
`plot()`	Plot method to evaluate SMLE screening/selection for objects of class '`smle`' or '`selection`'

Table 1: Main **SMLE** functions and their brief descriptions.

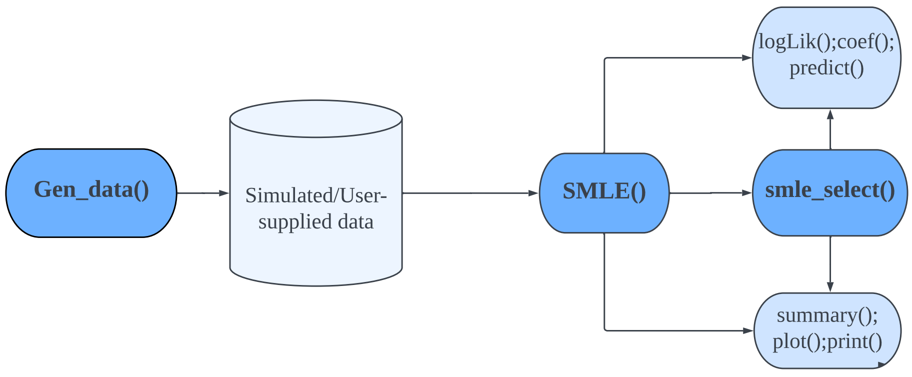

Gen_data()
Simulated/User-supplied data
SMLE()
smle_select()
logLik();coef(); predict()
summary(); plot();print()

Figure 1: Flowchart for calling functions in the **SMLE** package.

```
install.packages("SMLE")
library("SMLE")
```

SMLE requires the R packages **glmnet** (Friedman, Hastie, and Tibshirani 2010) for parameter initialization, **mvnfast** (Fasiolo 2023) and **matrixcalc** (Novomestky 2022) for simulating correlated data.

In this section, we first briefly describe the anticipated work flow for a data analysis using **SMLE**. We then provide a brief overview of the main functions of **SMLE** and illustrate the use and output of the functions through simple function calls. Detailed numerical examples are provided in Section 4.

3.1. SMLE work flow

The main **SMLE** functions are listed in Table 1. The work flow for a typical data analysis with **SMLE** is illustrated in Figure 1 and is summarized as follows:

- For simulation studies or for testing purposes, the function `Gen_Data()` can be used to simulate data assuming the response variable follows a GLM that depends on a small subset of possibly-correlated features. `Gen_Data()` returns an object of class '`sdata`' that contains the matrix of features, the response variable, and additional information about the model used to simulate the data.
- The function `SMLE()` is the main function based on Algorithm 1 for screening out fea-

tures that are unlikely to be related to the response variable. Users can pass a '`sdata`' object, a data frame, or a matrix containing the data to the function. `SMLE()` returns a '`smle`' object that contains the top k features and additional information about the screening process.

- The function `smle_select()` is used to conduct accurate post-screening feature selection based on Algorithm 2. Users can choose to implement this function with a selection criterion such as AIC, BIC or EBIC. Users can pass either a '`smle`' object or a data frame (matrix) to the function. `smle_select()` returns an object of class '`selection`' that provides information about the selected features and the selection process. Users may also choose to run `smle_select()` as a built-in option within the main function `SMLE()` by setting the argument `selection = TRUE`.
- The `plot()` function for a '`smle`' or '`selection`' object can be used to visualize the screening or selection results.
- The `predict()` function returns the fitted or predicted response values based on the features retained in a '`smle`' or '`selection`' object.
- The package also includes several functions with usage similar to existing R functions for summarizing objects and fitting regression models (e.g., `summary()`, `coef()`, `logLik()`).

3.2. Main functions and arguments

Simulating ultrahigh-dimensional GLM data

The motivation for the function `Gen_Data()` is to provide a freely available tool for simulating ultrahigh-dimensional GLM datasets with complex correlation structures between features. Some of the correlation structures available in this function were used in Xu and Chen (2014) to compare feature screening approaches. Both numerical and categorical features are permitted, and the number of features can be larger than the sample size. Users can choose the sample size n and the total number of features p in the dataset.

Numerical features are sampled from a normal distribution with one of four commonly-used correlation structures. The strength of correlation is controlled by a parameter ρ, which can optionally be set by the user using the argument `correlation`. The four different correlation structures are:

Independence (ID) All features are independently sampled from a standard normal distribution.

Moving average (MA) Features are jointly normal with covariance (correlation) ρ between adjacent features, $\rho/2$ between features two indices apart, and 0 otherwise. For example, the covariance matrix for four features would be:

$$\begin{bmatrix} 1 & \rho & \rho/2 & 0 \\ \rho & 1 & \rho & \rho/2 \\ \rho/2 & \rho & 1 & \rho \\ 0 & \rho/2 & \rho & 1 \end{bmatrix}$$

Compound symmetry (CS) Features are jointly normal with covariance $\rho/2$ if both features are causally related (relevant) to the response variable, and with covariance ρ otherwise. For example, with four features and assuming that features one and four are relevant, the covariance matrix used to simulate the features would be

$$\begin{bmatrix} 1 & \rho & \rho & \rho/2 \\ \rho & 1 & \rho & \rho \\ \rho & \rho & 1 & \rho \\ \rho/2 & \rho & \rho & 1 \end{bmatrix}$$

Auto-regressive (AR) Features are jointly normal with covariance $\text{cov}(x_j, x_h) = \rho^{|j-h|}$ for the jth and hth features with $j, h \in \{1, \ldots, p\}$.

Categorical features are generated by first binning a numerical feature that was simulated as described above. The number of groups (levels) for a categorical feature is specified with `level_ctgidx`. After binning, the feature is converted from class '`numeric`' to '`factor`' and each bin is assigned a character from 'A' to 'Z'. Users are able to specify the number of categorical features in the dataset with the argument `num_ctgidx`, and their positions in the dataset with the argument `pos_ctgidx`.

The response variable is simulated by assuming a GLM and that only a subset of the features are influential on the response. Normal, binary, and Poisson response variables are all available; the model is chosen with the argument `family`, as with the R function `glm()`. The user can choose the number of influential features with the argument `num_truecoef`, and optionally, their positions in the feature matrix and their model effects with the arguments `pos_truecoef` and `effect_truecoef`, respectively. If the positions of the influential features are not provided, they would be chosen randomly.

`Gen_Data()` returns an object of class '`sdata`' containing the response vector $\boldsymbol{y}$, the $n \times p$ feature matrix $\boldsymbol{X}$, and the coefficients for the features affecting the response.

The following code shows how to simulate a dataset with $n = 200$ observations and $p = 1000$ features, the first three of which are categorical, with three, four, and five levels, respectively. The response variable is generated based on a normal linear model with five influential features chosen by the function default.

```
R> set.seed(1)
R> Data_ctg <- Gen_Data(n = 200, p = 1000, family = "gaussian",
+    pos_ctgidx = c(1, 2, 3), level_ctgidx = c(3, 4, 5))
R> head(Data_ctg$X)[, 1:5]
```

```
  C1 C2 C3          X4         X5
1  A  B  C -1.29171904  0.1370216
2  A  B  D  0.90967045 -1.2996452
3  B  B  A -1.10775568  1.1514089
4  B  C  C -0.38412387  1.5134475
5  B  C  D  0.08273483  0.8021379
6  B  A  D -0.48388247 -0.4305369
```

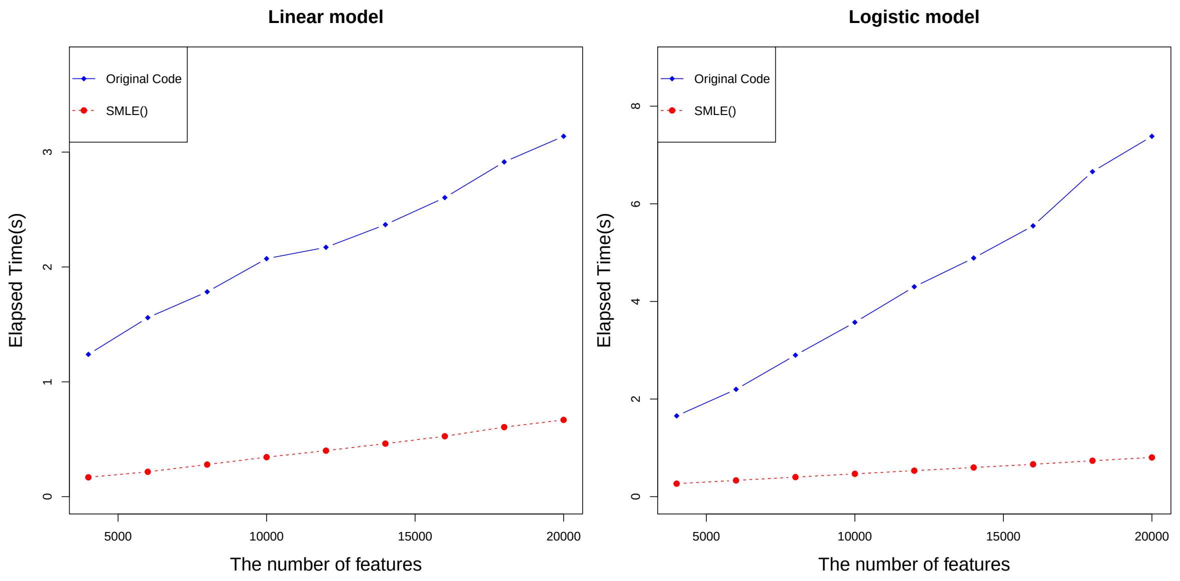


Figure 2: Running time comparison between `SMLE()` and the original IHT implementation in Xu and Chen (2014).

Joint feature screening

The main goal is to identify a manageable set of $k < p$ features that are most related to the response variable. To that end, `SMLE()` is used to screen out features unlikely to be influential (i.e., irrelevant features); it serves as a pre-processing step before an in-depth analysis. Users can pass information about the input data $(\boldsymbol{y}, \boldsymbol{X})$ to `SMLE()` via a '`sdata`' object, a data frame, or data matrices. When data are not input from a '`sdata`' object, the user should further specify the type of GLM with the argument `family`. The function conducts effective feature screening based on Algorithm 1, which naturally incorporates the joint effects among features. In `SMLE()`, we make use of the R function `crossprod()` to handle the ultrahigh dimensional matrix product involved in Algorithm 1 without doing matrix transpose; this leads to improved computational efficiency in comparison with the original implementation in Xu and Chen (2014). In Figure 2, we show the averaged elapsed time (in seconds) of `SMLE()` and the original IHT code on a series of datasets generated by `Gen_Data(correlation = "ID")` with $n = 200$ and p varying from 4000 to 20000. The running time is based on 100 repetitions.

In Table 2, we list the main arguments of `SMLE()` for specialized users to control the screening process. In particular, the argument `k` controls the number of important features to be retained from $\boldsymbol{X}$ after screening. The choice of `k` should reflect a user's belief or prior knowledge on the total number of relevant features for the input data. Intuitively, a larger `k` increases the chance of retaining all relevant features, while a smaller `k` brings more interpretive value and computational convenience for the subsequent in-depth analysis. One practical strategy is to set `k` to be three times larger than the anticipated number of relevant features. In `SMLE()`, the default value of `k` is the largest integer not exceeding $0.5\log(n)n^{1/3}$, which is recommended in Xu and Chen (2014) from a theoretical perspective.

The argument `coef_initial` allows users to specify an initial value $\beta^{(0)}$ for SMLE-screening. Since Algorithm 1 may lead to a local maximum, an informative $\beta^{(0)}$ helps to improve screening accuracy. This argument can be helpful in applications, where prior knowledge of the model coefficients are available. When the number of non-zero components of $\beta^{(0)}$ is larger than p, `SMLE()()` will truncate it to k, ensuring the non-decreasing property of the IHT algorithm. The default value of `coef_initial` is the Lasso (Tibshirani 1996) estimate with the largest sparsity not exceeding $n-1$, which is implemented by the function `glmnet(pmax = n-1)()` in the R package **glmnet**. This data-driven choice is recommended in Xu and Chen (2014) and is also supported by our numerical studies.

When a non-informative initialization is preferred, one may simply set $\beta^{(0)} = 0$, which runs Algorithm 1 with a null model. In this case, `SMLE()()` starts from scratch and iteratively detects important features via $\beta^{(t)}$ during the IHT process (see Figure 4). Since a non-decreasing likelihood is obtained by IHT updates, technically $\beta^{(t)}$ from Algorithm 1 can always be viewed as an improvement over $\beta^{(0)}$ in terms of the joint likelihood. In particular, when SMLE is used on linear models with $\beta^{(0)} = 0$, the screening result based on $\beta^{(1)}$ coincides with the SIS-screening based on marginal effects; $\beta^{(2)}$ further improves SIS by incorporating joint information in X. Readers may refer to Section 4.3 for a numerical comparison between the informative and non-informative initialization strategies.

In `SMLE()`, the initial value of u^{-1} is determined as follows. When $\boldsymbol{\beta}^{(0)} = 0$, we simply set $u^{-1} = U/\sqrt{p}$ with $U > 0$ being a magnification parameter. When $\boldsymbol{\beta}^{(0)} \neq 0$, we generate a sub-matrix $\boldsymbol{X}_0$ using the columns of $\boldsymbol{X}$ indicated by $\{j : \beta_j^{(0)} \neq 0\}$ and set

$$u^{-1} = \frac{U}{\sqrt{p}\|\boldsymbol{X}_0\|_\infty^2},$$

where $\|.\|_\infty$ is the matrix infinity norm denoting the maximum absolute row sum of a matrix. The use of $\sqrt{p}\|\boldsymbol{X}_0\|_\infty^2$ serves as a computationally convenient replacement for computing the largest eigenvalue of $\boldsymbol{X}^T\boldsymbol{X}$, which is used in Xu and Chen (2014) as a theoretical guidance for choosing u. Users can specify the magnification parameter U and the decrease rate $\tau \in (0,1)$ in u-search with the arguments `U` and `U_rate`, respectively. The default values are `U = 1` and `U_rate = 0.5`.

`SMLE()` terminates the IHT iterations when $\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^{(t-1)}\|_2$ is below the tolerance level specified in the argument `tol`. Since our goal here is feature screening, a large number of iterations t may not always be necessary. We observe that, in many of our numerical examples, `SMLE()` can successfully identify all the relevant features within a few iterations by estimating the corresponding coefficients to be non-zero. Therefore, when an accurate coefficient estimate is not needed, users may choose to set the argument `fast = TRUE`, which allows early stopping of Algorithm 1. Specifically, with `fast = TRUE`, `SMLE()` terminates the IHT iterations when one of the following rules is satisfied:

- $\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^{(t-1)}\|_2 < \mathtt{k}^{1/2} \times \mathtt{tol}$,
- $l(\boldsymbol{\beta}^{(t)}) - l(\boldsymbol{\beta}^{(t-1)}) < 0.01\ [\ l(\boldsymbol{\beta}^{(2)}) - l(\boldsymbol{\beta}^{(1)})\ ]$,
- The non-zero entries in $\{\boldsymbol{\beta}^{(t)}\}$ remain unchanged for 10 consecutive iterations.

When the input data for `SMLE()` contains categorical features, users should set the argument `categorical = TRUE`, which enables the function to automatically detect the locations and

Arguments	Description	Default value
`k`	Total number of features to be retained after screening.	$\frac{1}{2}\log(n)n^{1/3}$
`keyset`	A vector to indicate a set of key features that are forced to remain in the model.	`NULL`
`coef_initial`	Initial coefficient value $\boldsymbol{\beta}^{(0)}$ for IHT.	`glmnet(pmax = n - 1)`
`categorical`	Logical flag for whether the input feature matrix includes categorical features.	`NULL`
`group`	Logical flag for whether to treat the dummy variables corresponding to each categorical feature as a group.	`TRUE`
`tol`	A tolerance level for $\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^{(t-1)}\|_2$ to stop the iteration.	10^{-2}
`fast`	Logical flag to enable early stop for IHT.	FALSE
`U_rate`	Decrease rate in u-search.	0.5
`U`	Initial magnification level in u-search.	1

Table 2: Main arguments for the `SMLE()` function.

levels of the `factor` columns in the feature matrix. By default, a L-level categorical feature would be encoded by $L-1$ dichotomous dummy variables, which are to be retained or screened out together if `group = TRUE`.

With the `keyset` argument, users can choose to manually keep a subset of features in the SMLE-screening process; this can be useful when some features are known to be influential or confounding. The following code demonstrates the usage of `SMLE()` with `keyset` on the previously generated dataset `Data_ctg`. Here we ensure that the first (`C1`), the fourth (`X4`), and the fifth (`X5`), features are always kept during the IHT updates and are therefore surely retained after screening. In this example, since we choose to retain `k = 15` features in total, the retained set would include the 3 features specified in `keyset` plus 12 features to be suggested by Algorithm 1. When `keyset` contains categorical features, it is required to have `group = TRUE`.

```
R> fit  <-  SMLE(Y = Data_ctg$Y, X = Data_ctg$X, k = 15,
+    family = "gaussian", keyset = c(1, 4, 5),
+    categorical = TRUE, group = TRUE)
```

`SMLE()` returns an object of class '`smle`', which contains information regarding the input data, model assumption, IHT updates, and the screening results.

Post-screening selection

As mentioned in Section 2.3, the features retained after screening are still likely to contain some that are not related to the response. The function `smle_select()` is designed to implement Algorithm 2 to further identify the relevant features.

`smle_select()` can be applied to a '`sdata`' object, a '`smle`' object, or a user-supplied dataset $(\boldsymbol{y}, \boldsymbol{X}_s)$. As discussed before, when p is very large, a screening step is usually needed before

Arguments	Description	Default value
`k_min` (`k_max`)	The lower (upper) bound for candidate model sparsity.	`k_min` = 1, `k_max` = number of input features
`sub_model`	A index vector indicating which features are to be selected. Not applicable if a ‘`smle`’ object is the input.	`NULL`
`criterion`	Selection criterion to be used. One of `"ebic"`,`"bic"`,`"aic"`.	`"ebic"`
`gamma_ebic`	The EBIC parameter in $[0, 1]$.	0.5
`gamma_seq`	The sequence of values for `gamma_ebic` when `vote = TRUE`.	`(0,0.2,0.4,0.6,0.8,1)`
`vote`	The logical flag for whether to perform the voting procedure, when `criterion = "ebic"`.	`FALSE`
`vote_threshold`	A relative voting threshold in percentage. A feature is considered to be important when it receives votes passing the threshold.	0.6
`para`	Logical flag to use parallel computing to do selection.	`FALSE`

Table 3: Main arguments for the `smle_select()` function.

`smle_select()` can be efficiently applied. In the package, we provide an option to automatically run `smle_select()` after screening within the main function `SMLE()`.

We list the main arguments of `smle_select()` in Table 3. Users can set the range or specify a subset of features to be further selected. With the argument `criterion`, users can choose to run `smle_select()` using their preferred selection criterion.

When the criterion EBIC is used, users may further specify its tuning parameter using the argument `gamma_ebic`. Alternatively, one may set `vote = TRUE`, which would repeat the EBIC-based selection with a sequence of tuning parameters provided in `gamma_seq`. With different tuning parameters, different sets of features would be selected; a feature is considered to be important when it is selected more than `vote_threshold` of times in the entire procedure. Users may change this threshold to enlarge or reduce the size of the selected model by using the `vote_update()` function, which is a simple method for a ‘`selection`’ object that avoids the need to re-run `smle_select()`.

The following code demonstrates the use of `smle_select()` with EBIC voting on the ‘`smle`’ object `fit` generated before. The output `fit_s` is an object of class ‘`selection`’, which contains the information regarding the selection procedure and result.

```
R> fit_s <- smle_select(fit, criterion = "ebic",
+    gamma_seq = seq(0, 1, 0.2), vote = TRUE)
```

In `smle_select()`, the selection is done by evaluating a sequence of sub-models with sizes varying from `k_min` to `k_max`. The arguments `k_min` and `k_max` allow the users to control the searching range based on their prior knowledge or expectation about the model size.

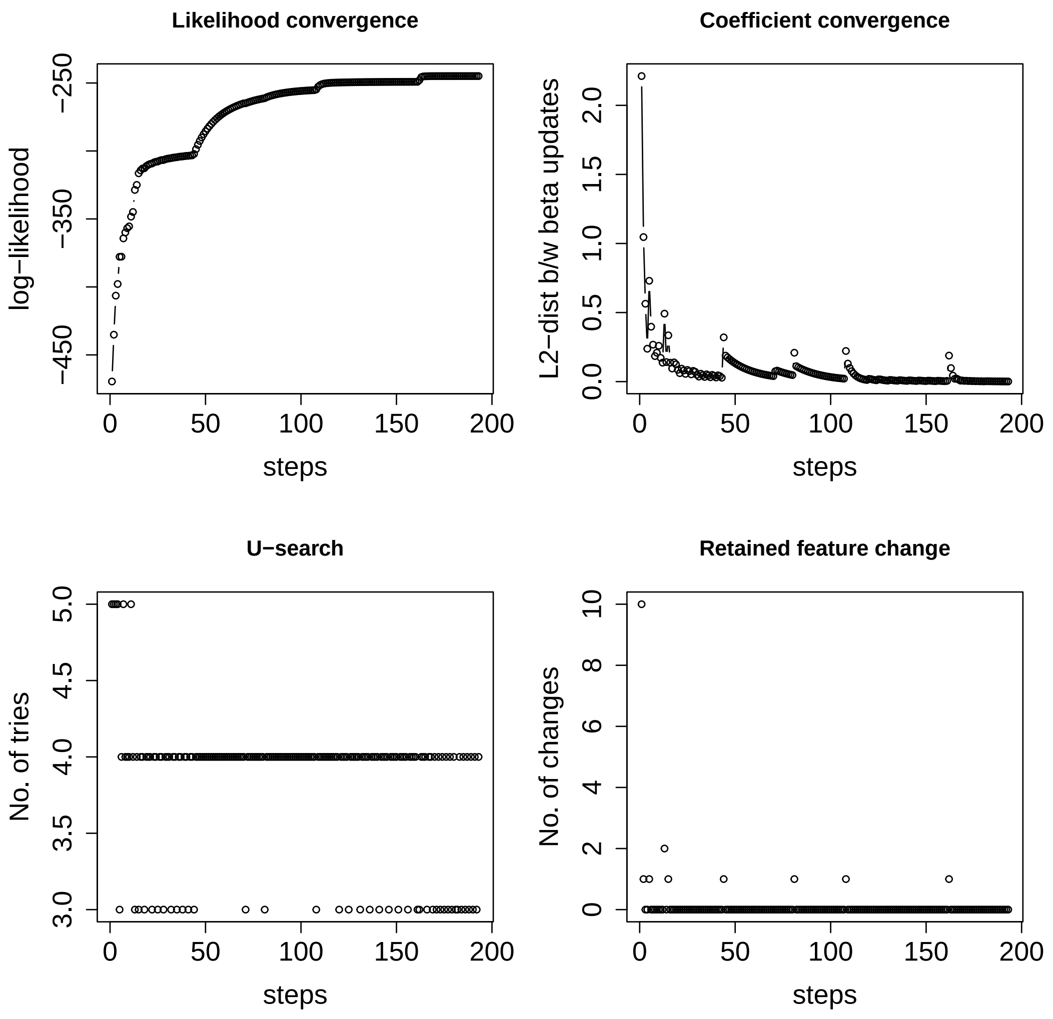


Figure 3: Example plot for '`smle`' class – IHT Convergence with $\boldsymbol{\beta}^{(0)} = 0$.

When the package is run in a multi-core computing environment, users may opt to use parallel computing to boost this selection procedure. Specifically, by setting `para = TRUE`, the creation and evaluation of the sub-models would be distributed to multiple computing cores for parallel processing. The implementation of the parallel option uses the **parallel** R package (R Core Team 2025) and depends on the user's operating system: for Unix and Mac users `mclapply()` is used and for Windows `parLapply()` is used. Note that `smle_select()` is mainly designed for post-screening selection, where the number of candidate features is moderate and the associated computational cost is often acceptable. Considering the communication cost between cores, parallel processing may not necessarily lead to a significant speed-up for the regular usage of `smle_select()`.

Plotting

Plot functions have been included for both the '`smle`' and '`selection`' classes. The `plot()` function for class '`smle`' returns two plot windows. By default, the first plot window contains

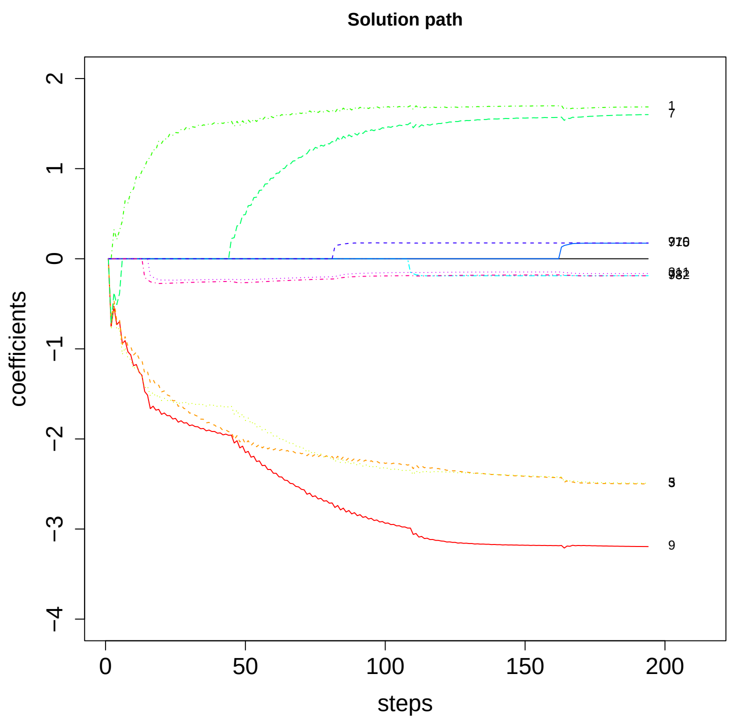


Figure 4: Example plot for '`smle`' class – IHT solution path with $\beta^{(0)} = 0$.

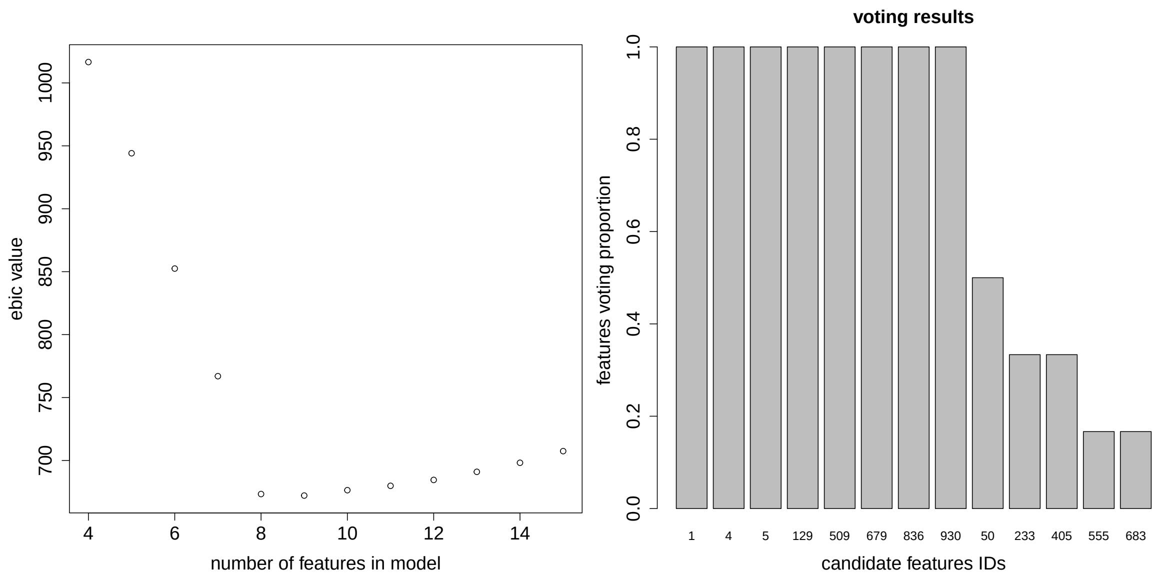
four subplots each showing a measure of IHT convergence on the vertical axis and iteration index on the horizontal axis; see Figure 3 as an example. The convergence measures are:

- log-likelihood (top left),
- Euclidean distance between the current and previous coefficient estimates (top right),
- the number of u-search tries in Algorithm 1 (bottom left),
- the number of membership changes in the retained feature set between the current and the previous IHT updates (bottom right).

The second plot window shows the solution path (estimated coefficient by IHT iteration) for selected features; see Figure 4 as an example. This plot provides a direct insight on how a coefficient changes over the iterations. By default, the solution path is given for all the relevant features suggested in the input '`smle`' object. Users may choose to plot a solution path for a customized group of features. This can be done by plotting only the top `num_path` features or plotting only the features specified in the argument `which_path`.

Users can optionally change which figure appears in the second plot window. For example, passing the argument `outplot = 1` will cause the log-likelihood to appear in the second plot window; the solution path will instead be plotted in the first plot window. Any additional arguments passed to `plot()` will be used when creating the plot in the second plot window.

The `plot()` function for an object of '`selection`' class returns a plot showing the selection criterion scores evaluated for the candidate sub-models with varying sizes (number of



(a) Selection criterion (b) Voting results

Figure 5: Example plot for '`selection`' class.

features); see Figure 5(a) as an example. A selection curve typically has a "V" shape: the curve decreases at the beginning as sub-model size increases before it increases again when there is little benefit in including additional features. For example, Figure 5(a) suggests a sub-model with six features, which are to be treated as influential ones. When criterion EBIC is used with `vote = TRUE`, `plot()` additionally returns a voting plot showing the relative inclusion frequency of each feature in the sequence of sub-models generated by Algorithm 2; see Figure 5(b).

Prediction

The `predict()` function in **SMLE** is based on the generic `predict()` function in **stats**; it returns predicted response values based on a model containing only the features retained in a '`smle`' object or selected in a '`selection`' object. The predictions are made by fitting the model using the `glm()` function.

As with the `predict()` function for class '`glm`' or '`lm`', users can optionally specify a data frame with new values for the features in the model using the `newdata` argument. If `newdata` is not provided, predictions will be given for the original training data; this leads to the fitted response values.

When using `predict()`, the type of prediction required must be specified using the `type` argument. The default is `type = link` which returns predictions on the scale of the linear predictor; `type = response` returns predictions on the scale of the response variable. In particular, with `type = response`, `predict()` returns the predicted mean responses for a linear or Poisson model and the predicted success probabilities for a logistic model.

Additional functions for regression-based objects

SMLE has some additional functions that can be used to extract information about the model fit. Specifically, the function `logLik()` first refits the screened (or selected) model using `glm()` and then returns the log-likelihood of the fitted model. The function `coef()` can be conveniently used to extract regression coefficients from a fitted '`smle`' or '`selection`' object. In addition, `summary()` is extended from the base function to take a '`smle`' or '`selection`' object and displays summary information about the fitted model object.

4. Examples

We demonstrate the usage and effectiveness of **SMLE** using a series of simulation studies. Numerical experiments were conducted on a Mac laptop with M1 chip and 8 GB memory and were performed using R 4.2.2 (R Core Team 2025).

4.1. Demo code for SMLE-screening

We first show how to use **SMLE** to conduct feature screening and post-screening selection via a simulated example. To this end, we use `Gen_Data()` to generate a synthetic dataset with $n = 400$ observations and $p = 1000$ features. We generate the feature matrix $\boldsymbol{X}$ from a multivariate normal distribution with an auto-regressive structure, where the adjacent features have a high correlation of $\rho = 0.9$. The response variable Y is generated based on the following logistic model with success rate π and linear predictor:

$$\text{logit}(\pi) = 2x_1 + 3x_3 - 3x_5 + 3x_7 - 4x_9.$$

In this setup, the feature matrix contains only five features that are causally-related to the response, as indicated in the model.

This dataset is generated by the following code; the resulting simulated data are stored in `Data_eg`, which is an object of class '`sdata`'. Users can use the `print()` function to show properties of the simulated data.

```
R> set.seed(1)
R> Data_eg <- Gen_Data(n = 400, p = 1000, family = "binomial",
+    correlation = "AR", rho = 0.9, pos_truecoef = c(1, 3, 5, 7, 9),
+    effect_truecoef = c(2, 3, -3, 3, -4))
R> print(Data_eg)

Call:
 Gen_Data(n = 400, p = 1000, pos_truecoef = c(1, 3, 5, 7, 9),
    effect_truecoef = c(2, 3, -3, 3, -4), correlation = "AR",
    rho = 0.9, family = "binomial")

An object of class sdata

Simulated Dataset Properties:
 Length of response: 400
 Dim of features: 400 x 1000
 Correlation: auto regressive
 Rho: 0.9
 Index of Causal Features: 1, 3, 5, 7, 9
 Model Type: binomial
```

We then run `SMLE()` to conduct SMLE-screening on the data example `Data_eg`, assuming that the identities of the causal features were unknown; the goal is to obtain a refined feature set by removing most irrelevant features from $\boldsymbol{X}$. The following code shows the simplest function call to `SMLE()`, where we aim to retain only $k = 10$ important features out of $p = 1000$.

```
R> fit1 <- SMLE(Y = Data_eg$Y, X = Data_eg$X, k = 10, family = "binomial",
+    coef_initial = rep(0, 1000))
R> summary(fit1)

Call:
 SMLE(X = Data_eg$X, Y = Data_eg$Y, k = 10, family = "binomial",
 coef_initial = rep(0, 1000))

An object of class summary.smle

Summary:

  Length of response: 400
  Dim of features: 400 x 1000
  Model type: binomial
  Model size: 10
  Feature name: 1, 3, 5, 7, 9, 10, 404, 470, 535, 661
  Feature index: 1, 3, 5, 7, 9, 10, 404, 470, 535, 661
  Intercept: 0.2793397
  Coefficients estimated by IHT: 1.894, 3.534, -2.648, 3.517, -4.617, -0.615,
  -0.429, -0.549, 0.382, 0.578
  Number of IHT iteration steps: 252
```

The function returns a '`smle`' object in the variable called `fit1`. The `summary()` function confirms that a refined set of 10 features is retained after 252 IHT iterations. We can see that all 5 causal features used to generate the response are retained in the refined set. This indicates that screening is successful; the dimensionality of the feature space is reduced from $p = 1000$ down to $k = 10$ without losing any important information.

In the code that follows, we fit a marginal regression between the response and the second feature, x_2, in `Data_eg`. From the true model, we know that x_2 is not causally-related to the response. Yet, we can see that the marginal effect of x_2 appears to be pretty high; thus, this irrelevant feature is likely to be retained in the model if the screening is done based on marginal effects only. In this example, `SMLE()` accurately removes x_2, as its screening naturally incorporates the joint effects among features.

```
R> with(Data_eg, coef(summary(glm(Y ~ X[,2], family = "binomial"))))
```

```
              Estimate Std. Error   z value     Pr(>|z|)
(Intercept) 0.02440072  0.1125979 0.2167067 8.284369e-01
X[, 2]      1.10465766  0.1337063 8.2618250 1.434619e-16
```

Note that the refined set returned in `fit1` still contains some irrelevant features; this is to be expected, as the goal of feature screening is merely to remove most irrelevant features before conducting an in-depth analysis. As discussed in Section 2.3, one may conduct an elaborate selection on the refined set to further identify the causal features. In particular, the `smle_select()` function in **SMLE** can be readily used for the purpose of post-screening selection. The following code shows how this function can be used on `fit1` with the selection criterion EBIC. As can be seen below, `smle_select()` returns a '`selection`' object `fit1_s`, which exactly identifies the five features in the true data generating model.

```
R> fit1_s <- smle_select(fit1, criterion = "ebic")
R> summary(fit1_s)

Call:
  smle_select(object = fit1, criterion = "ebic")

An object of class summary.selection

Summary:

  Length of response: 400
  Dim of features: 400 x 1000
  Model type: binomial
  Selected model size: 5
  Selected features: 1, 3, 5, 7, 9
  Selection criterion: ebic
  Gamma for ebic: 0.5
```

As mentioned in Section 3.2, users can visualize the above screening and selection results by using the `plot()` function. Users can also make use of the arguments such as `keyset` or `k_min` to refine the above results.

4.2. Screening performance

Next, we assess the performance of **SMLE** in terms of screening accuracy and efficiency. To this end, we use `Gen_data()` to generate 500 independent datasets with $p = 2000$ and $n = 100, 250, 600$ from linear, Poisson, and logistic models, respectively. For all these datasets, a compound symmetry structure with $\rho = 0.3$ is used, where the first four features (i.e., x_1, x_2, x_3, x_4) are used to generate the response variable. An equal coefficient is assigned to each of the four causal features; the value of the coefficient is set to 2.5, 0.7, and 1.1 for the three aforementioned models, respectively.

We use `SMLE()` in our package to conduct feature screening on these simulated datasets to retain only $k = 20, 10, 30$ important features for linear, Poisson, and logistic models, respectively. We measure the screening accuracy by sure screening rate (SSR) and positive

Model	Functions	PRR	SSR	Time
Linear	`SIS(iter = FALSE)`	0.21	0.00	0.02
	`SIS(iter = TRUE)`	0.73	0.60	1.36
	`screenIID()`	0.18	0.00	0.41
	`glmnet()`	0.37	0.04	<0.01
	`abess()`	0.65	0.46	0.01
	`SMLE()`	1.00	1.00	0.01
Poisson	`SIS(iter = FALSE)`	0.08	0.00	0.05
	`SIS(iter = TRUE)`	0.51	0.38	3.26
	`screenIID()`	0.50	0.00	2.21
	`glmnet()`	0.09	0.00	0.01
	`abess()`	0.63	0.51	0.03
	`SMLE()`	0.93	0.85	0.81
Logistic	`SIS(iter = FALSE)`	0.53	0.04	0.15
	`SIS(iter = TRUE)`	0.65	0.37	9.09
	`screenIID()`	0.50	0.00	14.69
	`glmnet()`	0.78	0.43	0.02
	`abess()`	0.91	0.81	0.14
	`SMLE()`	0.94	0.82	0.10

Table 4: Comparison of screening performance of `SIS()`, `glmnet()`, `screenIID()`, `abess()` and `SMLE()`.

retaining rate (PRR). SSR is reported as the proportion of times that all causal features are retained after screening; PRR is calculated by the averaged proportion of causal features that are retained after screening. An averaged elapsed time (in seconds) for `SMLE()` in each model setup is reported as a measure of screening efficiency. For comparison, the performances of (I)SIS, DC-SIS, and Lasso are also reported under the same setup. Moreover, we repeat the experiments using the method of Abess, which was proposed in Zhu, Wen, Zhu, Zhang, and Wang (2020) for the high-dimensional best subset selection. Following the recommendations from the corresponding packages, we implement SIS by `SIS(iter = FALSE)` and ISIS by `SIS(iter = TRUE)` in package **SIS** (Saldana and Feng 2018), DC-SIS by `screenIID()` in package **VariableScreening** (Li *et al.* 2022), Lasso by `glmnet()` in package **glmnet** (Friedman *et al.* 2010), and Abess by `abess()` in package **abess** (Zhu *et al.* 2022).

We summarize the simulation results in Table 4, where all decimal numbers are rounded to 2 digits. It can be seen that `SMLE()` has the highest accuracy for all three model setups in terms of both SSR and PRR; the other methods (packages) seem to be heavily affected by the correlation among the features. By the nature of Algorithm 1, the high accuracy of `SMLE()` comes with a computational cost, but this is moderate in most cases. Considering the improved accuracy of `SMLE()`, this small computational investment seems to be quite worthwhile.

4.3. Impacts of initialization

As discussed in Section 3.2, a good choice of $\beta^{(0)}$ may be beneficial for SMLE-screening. To

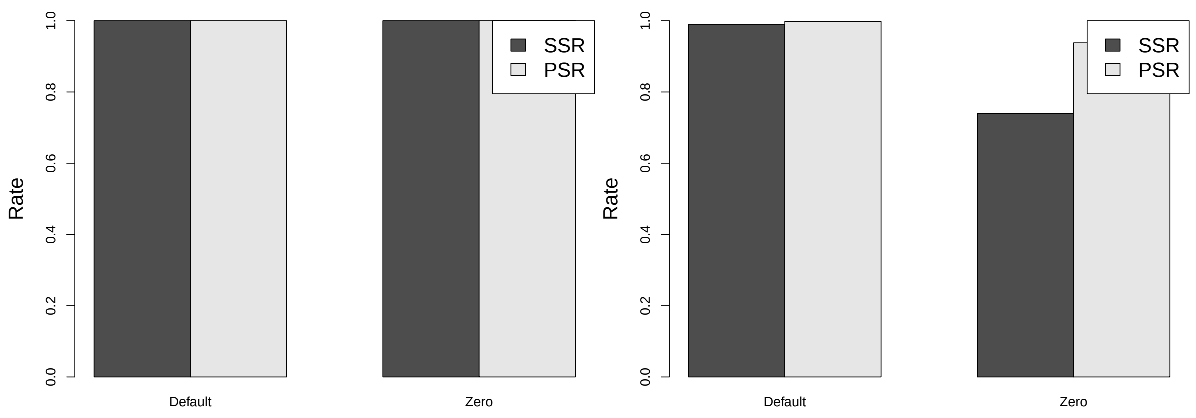


Figure 6: Screening accuracy of `SMLE()` when initialized with default values or a zero-vector. Left: Data simulated to have independent features. Right: Data simulated with autocorrelated features with $\rho = 0.9$.

provide some insights, we ran `SMLE()` on simulated datasets with `coef_initial` being the default value or a zero vector. The datasets in this example are generated by the following code.

```
R> Data_S1 <- Gen_Data(n = 300, p = 1000, family = "binomial",
+    correlation = "ID", pos_truecoef = c(1, 3, 5, 7, 9),
+    effect_truecoef = c(2, 3, -3, 3, -4))
R> Data_S2 <- Gen_Data(n = 400, p = 1000, family = "binomial",
+    correlation = "AR", rho = 0.9, pos_truecoef = c(1, 3, 5, 7, 9),
+    effect_truecoef = c(2, 3, -3, 3, -4))
```

Based on `Data_S1` and `Data_S2`, we calculated SSR and PSR of SMLE-screening with both default and zero start for 100 repetitions; the results are shown in Figure 6. It is seen that both initialization strategies lead to high screening accuracy on `Data_S1`, where features are independent with each other. The benefits of using an informative $\beta^{(0)}$ is observed on `Data_S2`, where strong correlations present among the features. These findings seem to support the recommended value of `coef_initial` in the **SMLE** package; users may also make their own choice of `coef_initial` based on the prior knowledge about model parameters.

4.4. Application to high-dimensional genetic data

To further demonstrate **SMLE**, we applied it to a simulated high-dimensional genetic dataset to detect associations between single-nucleotide polymorphisms (SNPs) and a response variable. To get a realistic genetic dataset, the genotypes were sampled from genotypic distributions derived from the 1000 Genomes project, Phase 1 (The 1000 Genomes Project Consortium 2015) using the R package **sim1000G** (Dimitromanolakis, Xu, Krol, and Briollais 2019). Since the SNP distributions are derived from individuals from existing human populations, this dataset naturally has a complex correlation structure. In particular, it is believed that strong non-linear correlations are present between SNPs in close proximity from the same chromosome.

The dataset consists of $p = 10,031$ SNPs from chromosomes 14 through 22 on $n = 800$ indi-

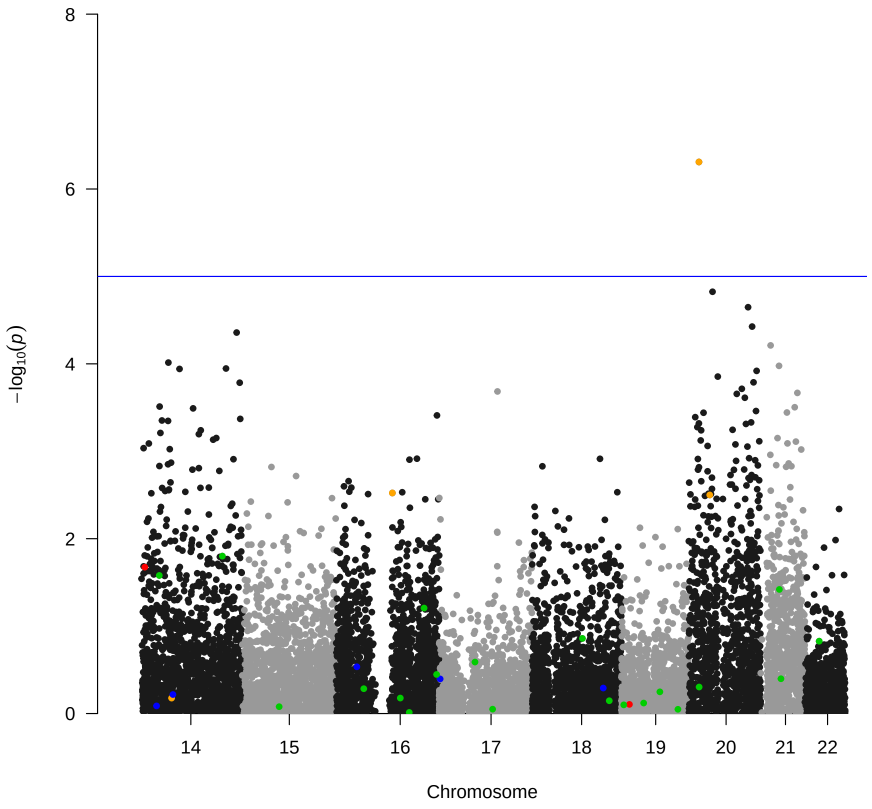


Figure 7: Manhattan plot of $-\log_{10}(p$ value) for the single-SNP association test by package **qqman** (Turner 2018). The blue line represents suggestive significance. The colored points correspond to SNPs simulated to be causally related to the response in the order of importance: red, orange, blue, green.

viduals. The genotypes were coded as 0, 1, or 2 by counting the number of minor alleles (the allele that is less common in the sample). The continuous response variable was simulated from a normal distribution with mean that depends additively on the causal SNPs and a standard deviation of 8. The linear predictor for the response assumed a polygenic model, which specified 31 SNPs with small to large effects. The goal here is to locate regions containing the 31 causal SNPs; the effects of those SNPs were determined as follows.

- Twenty SNPs were randomly selected to have small effects, which were drawn from a $N(0, 0.1)$ distribution.
- Five SNPs were randomly selected to have moderate effects, which were drawn from a $N(0, 0.5)$ distribution.
- Four SNPs (labelled: rs2967291, rs1534941, rs112915930, rs6132052) were selected to have large magnitude effects. Their effects were fixed at 2, -2, 1 and -1, respectively.
- Two SNPs were selected to have an effect on the response only through an interaction; no marginal effect was simulated. The two SNPs are labelled rs12608528 and rs11157211 and their interaction coefficient was set to be 4.

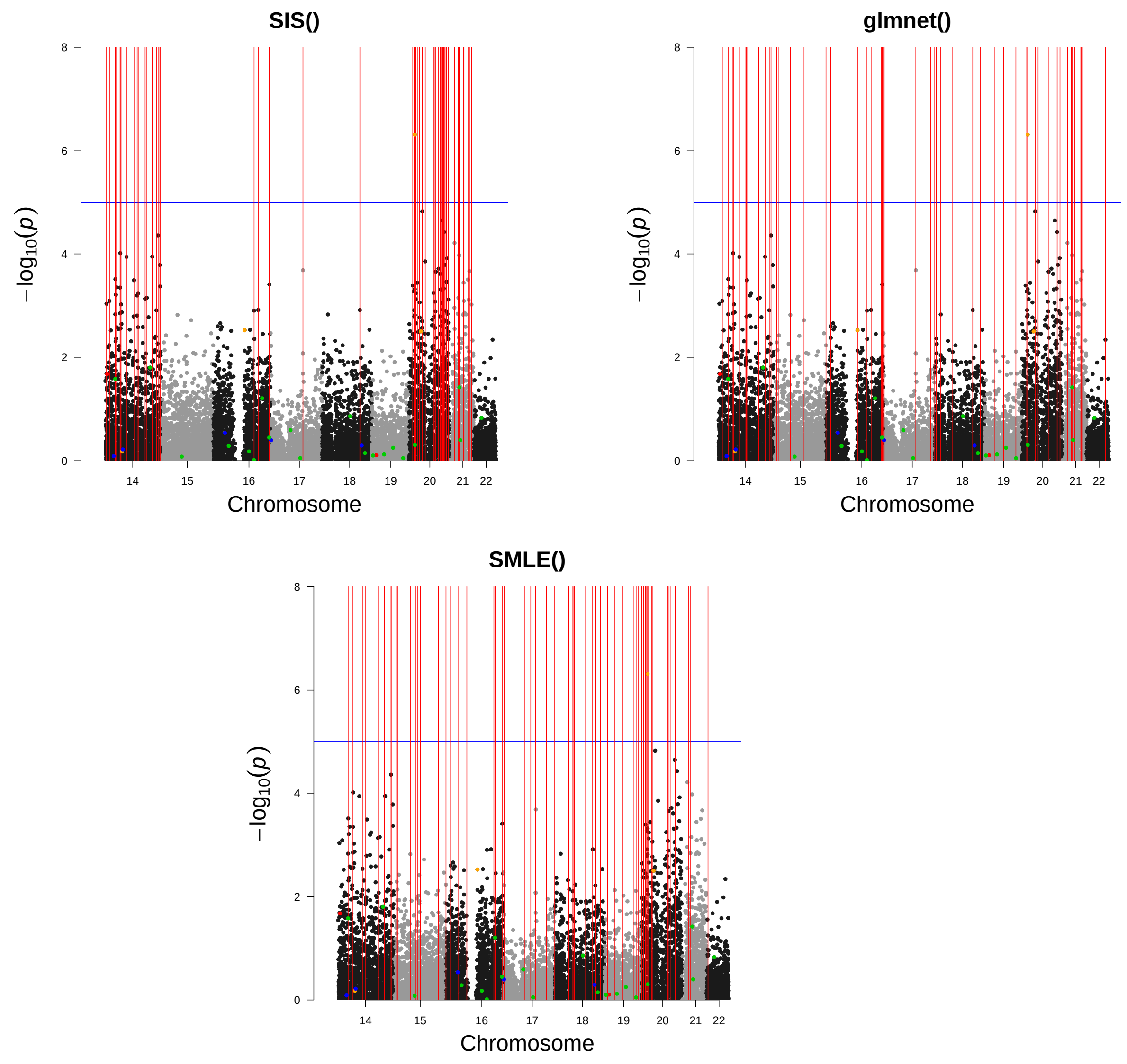


Figure 8: Important genetic regions flagged with red lines by `SIS()` (top left), `glmnet()`(top right) and `SMLE()`(bottom).

- The intercept was arbitrarily chosen to be 40 in order to avoid negative response values.

This simulated genetic dataset is included in **SMLE** and can be loaded with the following command.

```
R> data("synSNP")
```

We saved the response as a vector `Y_SNP` and the features as a matrix `X_SNP`.

```
R> Y_SNP <- synSNP[, 1]
R> X_SNP <- as.matrix(synSNP[, -1])
```

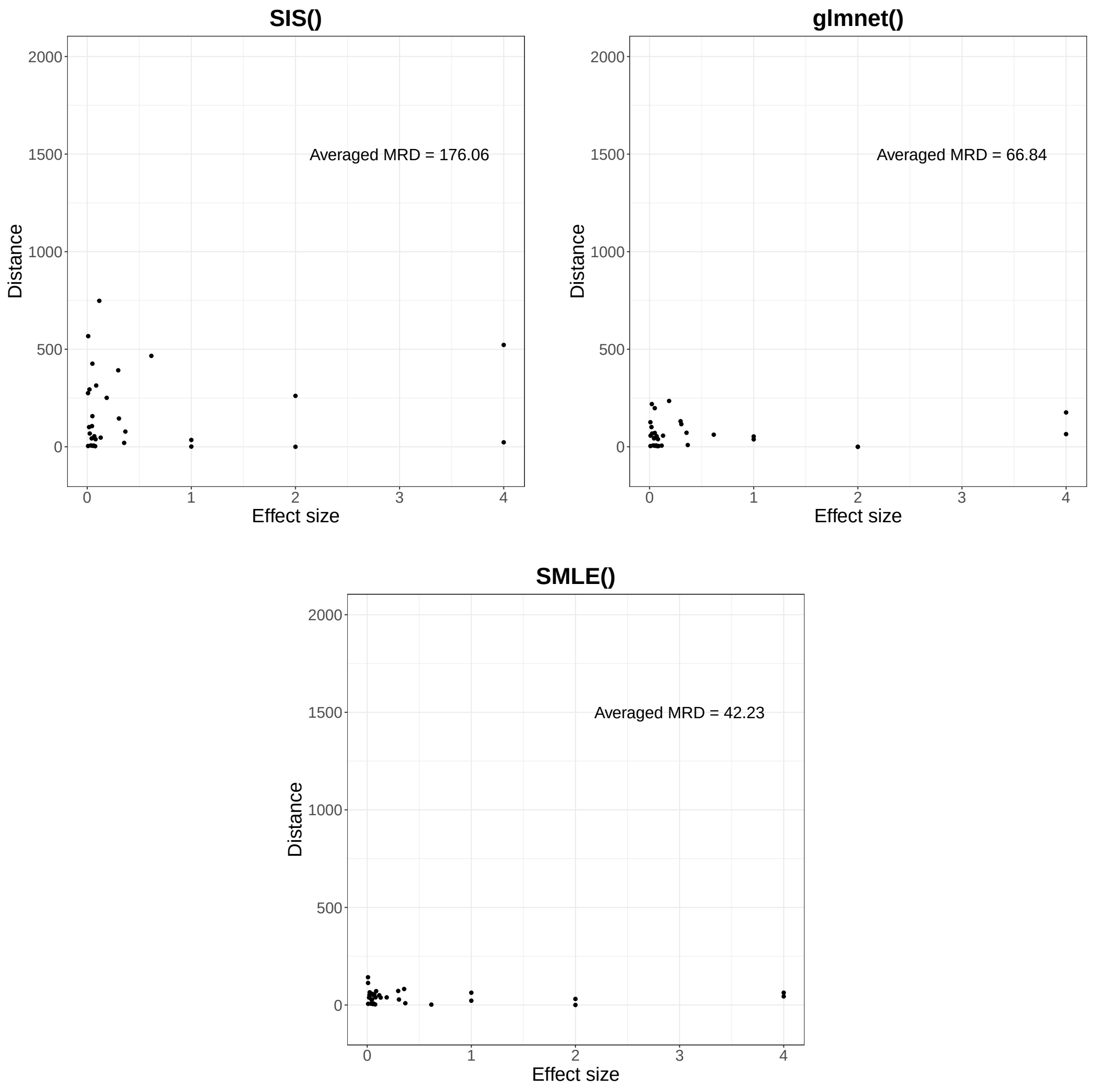


Figure 9: MRD with the casual SNPs for `SIS()` (top left), `glmnet()` (top right), and `SMLE()` (bottom).

The goal of study here is to flag genomic regions likely to contain the causal SNPs. This job can be typically done by identifying a small set of key SNPs and then marking the nearby locations. As a conventional approach, we first performed a single-SNP analysis using linear regression to test the marginal effects of the SNPs on the response. We report $-\log_{10}$ of the p value of the corresponding estimated marginal coefficients using a Manhattan plot (Figure 7). The p value of the 31 causal SNPs are colored based on their effect sizes. The blue line shows the conventional threshold used to declare suggestive significance (p value $< 10^{-5}$). We observed that only one SNP passed this threshold, which was simulated to have a large effect (rs11157211). All other SNPs did not show marginal significance and thus were not treated as the key ones. As a result, the conventional marginal-test-based approach failed to flag all the regions harboring the causal SNPs.

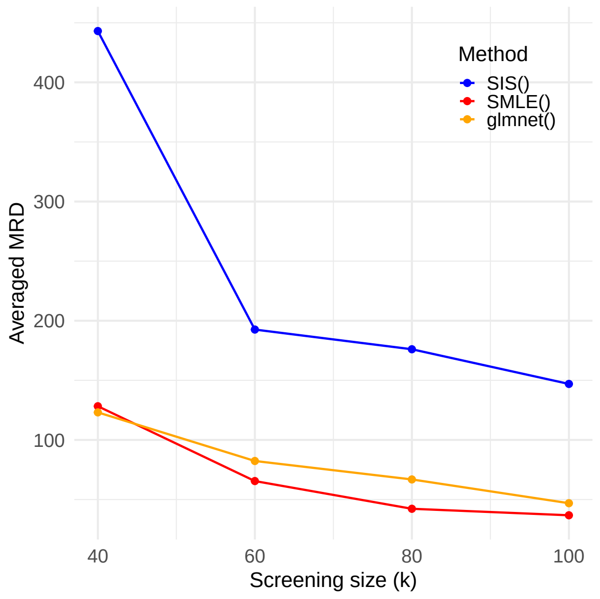


Figure 10: The averaged MRD on causal SNPs for `SIS()`, `SMLE()` and `glmnet()` with varying k.

We then ran `SMLE()` to flag genomic regions by retaining $k = 80$ key SNPs after screening. The results were compared with those obtained by `SIS()` and `glmnet()`. The three methods were carried out with the following code.

```
R> SMLE_fit <- SMLE(Y = Y_SNP, X = X_SNP, family = "gaussian", k = 80,
+    fast = FALSE)
R> SIS_fit <- SIS(y = Y_SNP, x = X_SNP, family = "gaussian", nsis = 80,
+    iter = FALSE)
R> lasso_fit <- glmnet(x = X_SNP,y = Y_SNP, family = "gaussian", pmax = 80)
```

Figure 8 shows the screening result from the above code. It is observed that `SIS()` focused on chromosomes 14, 20, and 21, where SNPs have relatively larger marginal effects; in contrast, `SMLE()` and `glmnet()` flagged regions across all chromosomes.

To assess the screening performance, for each of the 31 causal SNPs we computed the minimum distance between that SNP and the SNPs in the retained set (MRD). A small MRD indicates that the retained set contains at least one SNP that is close to the causal SNP; consequently, genetic regions suggested by the retained SNPs are likely to contain the causal SNPs. Therefore, an effective screening is expected to have small MRDs for most causal SNPs.

Figure 9 shows the MRDs of the 31 causal SNPs from the screening conducted with `SMLE()`, `SIS()`, `glmnet()`. The horizontal axis of the figure denotes the true effect sizes of the causal SNPs. Since the SIS method only uses the marginal information, the 80 locations suggested by `SIS()` tend to be very close to the peak on chromosome 20, as shown in Figure 7; this leads to large MRDs (with a mean value of 176.06) for most causal SNPs that are away from this peak. Since `SMLE()` and `glmnet()` consider joint effects, they achieved much smaller MRDs, with the mean values given by 42.23 and 66.84 respectively.

We repeated the analysis by increasing the model screening size, k. The corresponding averaged MRDs for all causal SNPs are shown in Figure 10. The performance of all three methods improves as k increases. Compared with the other two methods, `SMLE()` tends to be more stable; it conducts effective screening by consistently achieving a low averaged MRD over different choices of k. The high reliability makes it a preferable choice in practice.

To summarize, as shown in Figure 8, both `SMLE()` and `glmnet()` flagged regions across all chromosomes, while `SIS()` only focuses on chromosomes containing SNPS with large marginal effects. The benefit of using `SMLE()` is demonstrated in Figure 9, where `SMLE()` leads to about 20% improvement over `glmnet()` in the averaged MRD among all causal SNPs. `SMLE()` also shows an advantage in locating the two causal SNPs (rs12608528, rs11157211) with a large interaction effect. The stable performance of `SMLE()` is further supported by Figure 10 with varying screening sizes.

5. Concluding remarks

In this paper, we introduced a user-friendly feature screening package **SMLE** as a powerful tool to process ultrahigh-dimensional GLMs. The package provides an efficient implementation of the joint screening method SMLE, which leads to more reliable screening results compared with the commonly-used marginal methods. In **SMLE**, users can also conduct accurate post-screening selection based on an accelerated IHT procedure with a preferred selection criterion. Plotting tools are provided to visualize the screening and selection results, which can be readily used for making inference or prediction.

We illustrated the usage of the main functions in **SMLE** with discussions and examples. The effectiveness and efficiency of the package are supported by extensive numerical studies.

The idea of SMLE has been demonstrated to be attractive in processing many types of ultrahigh-dimensional data. In the current version, the package focuses on feature screening for GLMs, which have been widely used in many scientific areas. It would be promising to extend the application scope of **SMLE** to other modeling scenarios such as Cox's proportional hazards models and multivariate response regression, where the SMLE-based methodology has been discussed in the literature.

For future work, it is also of great interest to embed SMLE in a distributed (federated) computing framework (Sun, He, Wu, and Huang 2023), to better enable feature screening for big data stored in data segments. Depending on the nature of the research problem, each data segment may measure the same set of features on different groups of objects, or different sets of features on the same group of objects. Due to legal, ethical or commercial restrictions, aggregating those segments is usually not allowed, and thus it is preferable to use SMLE in a distributed manner. Equipping SMLE with recent distributed estimation techniques (e.g., Jordan, Lee, and Yang 2018) provides a possible route to explore this emerging domain.

Acknowledgments

This work was supported in part by National Key R&D Program of China under grant 2023YFA1008703, Major Key Project of PCL under grant PCL2024A06, and Natural Science Engineering Research Council of Canada under grant RGPIN-2019-06051. The content is solely the responsibility of the authors and does not necessarily represent the official views of the aforementioned funding agencies.

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723. doi:10.1109/tac.1974.1100705.

Blumensath T, Davies ME (2009). "Iterative Hard Thresholding for Compressed Sensing." *Applied and Computational Harmonic Analysis*, **27**(3), 4028–4031. doi:10.1016/j.acha.2009.04.002.

Chen J, Chen Z (2008). "Extended Bayesian Information Criteria for Model Selection with Large Model Spaces." *Biometrika*, **95**(3), 759–771. doi:10.1093/biomet/asn034.

Dimitromanolakis A, Xu J, Krol A, Briollais L (2019). "**sim1000G**: A User-Friendly Genetic Variant Simulator in R for Unrelated Individuals and Family-Based Designs." *BMC Bioinformatics*, **20**(1), 26. doi:10.1186/s12859-019-2611-1.

Donoho DL (2006). "Compressed Sensing." *IEEE Transactions on Information Theory*, **52**(4), 1289–1306. doi:10.1109/tit.2006.871582.

Fan J, Lv J (2008). "Sure Independence Screening for Ultrahigh Dimensional Feature Space." *Journal of the Royal Statistical Society B*, **70**(5), 849–911. doi:10.1111/j.1467-9868.2008.00674.x.

Fan J, Samworth R, Wu Y (2009). "Ultrahigh Dimensional Feature Selection: Beyond the Linear Model." *Journal of Machine Learning Research*, **10**, 2013–2038.

Fan J, Song R (2010). "Sure Independence Screening in Generalized Linear Models with NP-Dimensionality." *The Annals of Statistics*, **38**(6), 3567–3604. doi:10.1214/10-aos798.

Fasiolo M (2023). ***mvnfast**: Fast Multivariate Normal and Student's t Methods.* doi:10.32614/CRAN.package.mvnfast. R package version 0.2.8.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Jordan M, Lee J, Yang Y (2018). "Communication-Efficient Distributed Statistical Inference." *Journal of the American Statistical Association*, **114**, 668–681. doi:10.1080/01621459.2018.1429274.

Li R, Huang L, Dziak J (2022). ***VariableScreening**: High-Dimensional Screening for Semiparametric Longitudinal Regression.* doi:10.32614/CRAN.package.VariableScreening. R package version 0.2.1.

Li R, Zhong W, Zhu L (2012). "Feature Screening via Distance Correlation Learning." *Journal of the American Statistical Association*, **107**(499), 1129–1139. doi:10.1080/01621459.2012.695654.

Li X, Li R, Xia Z, Xu C (2020). "Distributed Feature Screening via Componentwise Debiasing." *Journal of Machine Learning Research*, **21**(24), 1–32.

Liu J, Zhong W, Li R (2015). "A Selective Overview of Feature Screening for Ultrahigh-Dimensional Data." *Science China Mathematics*, **58**(10), 1–22. doi:10.1007/s11425-015-5062-9.

Nelder JA, Wedderburn RWM (1972). "Generalized Linear Models." *Journal of the Royal Statistical Society A*, **135**(3), 370–384. doi:10.2307/2344614.

Novomestky F (2022). ***matrixcalc**: Collection of Functions for Matrix Calculations.* doi:10.32614/CRAN.package.matrixcalc. R package version 1.0-6.

Qu L, Hao M, Sun L (2022). "Sparse Composite Quantile Regression with Ultra-High Dimensional Heterogeneous Data." *Statistica Sinica*, **32**, 459–475. doi:10.5705/ss.202020.0115.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. doi:10.32614/R.manuals. URL https://www.R-project.org/.

Saldana DF, Feng Y (2018). "**SIS**: An R Package for Sure Independence Screening in Ultrahigh-Dimensional Statistical Models." *Journal of Statistical Software*, **83**(2), 1–25. doi:10.18637/jss.v083.i02.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Sun X, He Y, Wu D, Huang J (2023). "Survey of Distributed Computing Frameworks for Supporting Big Data Analysis." *Big Data Mining and Analytics*, **6**, 154–169. doi:10.26599/bdma.2022.9020014.

The 1000 Genomes Project Consortium (2015). "A Global Reference for Human Genetic Variation." *Nature*, **526**(7571), 68–74. doi:10.1038/nature11632.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, **58**(1), 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

Turner S (2018). "**qqman**: An R Package for Visualizing GWAS Results Using Q-Q and Manhattan Plots." *The Journal of Open Source Software*. doi:10.21105/joss.00731.

Wang H (2009). "Forward Regression for Ultra-High Dimensional Variable Screening." *Journal of the American Statistical Association*, **104**(488), 1512–1524. doi:10.1198/jasa.2008.tm08516.

Wu Y, Yin G (2015). "Conditional Quantile Screening in Ultrahigh-Dimensional Heterogeneous Data." *Biometrika*, **102**(1), 65–76. doi:10.1093/biomet/asu068.

Xu C, Chen J (2014). "The Sparse MLE for Ultrahigh-Dimensional Feature Screening." *Journal of the American Statistical Association*, **109**(507), 1257–1269. doi:10.1080/01621459.2013.879531.

Yang G, Hou S, Wang L, Sun Y (2018). "Feature Screening in Ultrahigh-Dimensional Additive Cox Model." *Journal of Statistical Computation and Simulation*, **88**(6), 1117–1133. doi:10.1080/00949655.2017.1422127.

Yang G, Yu Y, Li R, Buu A (2016). "Feature Screening in Ultrahigh Dimensional Cox's Model." *Statistica Sinica*, **26**(3), 881–901. doi:10.5705/ss.2014.171.

Zang Q, Xu C, Burkett K (2026). ***SMLE**: An R Package for Joint Feature Screening in Ultrahigh-Dimensional GLMs*. doi:10.32614/CRAN.package.SMLE. R package version 2.2-3.

Zhou T, Zhu L, Xu C, Li R (2020). "Model-Free Forward Screening via Cumulative Divergence." *Journal of the American Statistical Association*, **115**(531), 1393–1405. doi:10.1080/01621459.2019.1632078.

Zhu J, Wang X, Hu L, Huang J, Jiang K, Zhang Y, Lin S, Zhu J (2022). "**abess**: A Fast Best-Subset Selection Library in Python and R." *Journal of Machine Learning Research*, **23**(202), 1–7.

Zhu J, Wen C, Zhu J, Zhang H, Wang X (2020). "A Polynomial Algorithm for Best-Subset Selection Problem." *Proceedings of the National Academy of Sciences of the United States of America*, **117**(52), 33117–33123. doi:10.1073/pnas.2014241117.

Zhu L, Li L, Li R, Zhu L (2011). "Model-Free Feature Screening for Ultrahigh-Dimensional Data." *Journal of the American Statistical Association*, **106**(496), 1464–1475. doi:10.1198/jasa.2011.tm10563.

Affiliation:

Qianxiang Zang, Kelly Burkett
Department of Mathematics and Statistics
Faculty of Science
University of Ottawa, Ontario, Canada
E-mail: qzang023@uottawa.ca, kburkett@uottawa.ca

Chen Xu
Department of Mathematics and Fundamental Research
Pengcheng Laboratory, Shenzhen, China &
School of Mathematics and Statistics,
Xi'an Jiaotong University, Xi'an, China
Email: cx3@xjtu.edu.cn

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/

December 2025, Volume 115, Issue 8 *Submitted:* 2023-01-23
doi:10.18637/jss.v115.i08 *Accepted:* 2025-02-28