JDSSV

Journal of Data Science, Statistics, and Visualisation

February 2026, Volume VI, Issue I. *doi: 10.52933/jdssv.v6i1.167*

Optimal Subsampling for Linear Models with Heteroscedasticity

Jiayi Zheng
George Mason University

Dongqi Fu
George Mason University

Ziqiao Xu
George Mason University

Nicholas Rios
George Mason University

Abstract

In recent years, the size of datasets has dramatically increased. This has encouraged the use of subsampling, where only a subset of the full dataset is used to fit a model in a more computationally efficient manner. Existing methods do not provide much guidance on how to find optimal subsamples for a linear model when the variance of the errors depends on the model covariates through an unknown function. This paper presents three main contributions that aid in finding optimal subsamples in the case of heteroscedastic errors. First, a kernel-based method is proposed for estimating the error variances in the full dataset based on a Latin Hypercube subsample. Second, a generalized version of the Information-Based Optimal Subdata Selection (IBOSS) algorithm is introduced that uses the variance estimates to find subsamples with high $D-$efficiency. Third, an Approximate Nearest Neighbor Simulated Annealing (ANNSA) algorithm is used to find subsamples that are efficient under the $I-$optimality criterion, which seeks to minimize integrated prediction error variance. Simulations show that the proposed subsampling algorithms have better $D-$ and $I-$efficiencies than existing methods. The subsampling methods are used to analyze an airline dataset with over 7 million rows.

Keywords: Design of experiments, $I-$optimality, optimal subsampling, heteroscedasticity.

1. Introduction

The rapid growth of data availability in modern scientific research and computation has led to increasingly large datasets across many fields. In particular, studying linear relationships through regression analysis remains a fundamental task. A typical linear regression model involves a response vector of size $N \times 1$ and a covariate matrix of size $N \times (p+1)$, with p covariates and an intercept. As the sample size N grows, the computational cost associated with matrix operations such as matrix multiplication and inversion becomes increasingly expensive without adequate computational resources. One practical approach to alleviate this computational burden is subsampling, where only a subset of observations is selected from the full dataset for model fitting (Yao and Wang 2021). By reducing the effective sample size, subsampling can dramatically decrease computation time while still capturing key information from the data (Wang et al. 2018). In this project, we focus on developing and studying subsampling strategies in the presence of heteroskedasticity, where the variability of the response depends on the covariates. Accounting for heteroskedasticity is crucial to ensure that the subsample remains informative and that statistical efficiency is not unduly compromised.

In many real-world scenarios, the size of available data can overwhelm computational resources, making subsampling an attractive solution. For example, the chemical sensor array dataset (Fonollosa et al. 2015) consists of $N = 4{,}208{,}261$ observations with 15 variables, capturing time-series data from gas sensors exposed to turbulent environments. An airline on-time performance dataset from the 2009 ASA Data Expo contains over $N = 123{,}534{,}969$ records with 29 variables, documenting commercial flight operations in the United States over multiple decades (Wang et al. 2019). Similarly, the SUSY dataset (Baldi et al. 2014), derived from high-energy physics simulations of supersymmetric particle collisions, includes $N = 5{,}000{,}000$ observations with 18 features, designed to benchmark classification algorithms in distinguishing signal from background events. These large-scale datasets exemplify situations where subsampling methods can greatly enhance computational feasibility while preserving statistical efficiency.

Various subsampling strategies have been proposed to alleviate the computational burden of fitting models to massive datasets. Early approaches include leverage-based subsampling, which assigns sampling probabilities proportional to leverage scores derived from the projection matrix of the design matrix (Drineas et al. 2006). More recent developments focus on optimal subsampling, where sampling probabilities are directly derived by minimizing specific statistical optimality criteria. For example, Wang et al. (2018) proposed OSMAC, which is a probabalistic method that uses weights to search for A-optimal and L-optimal designs. In 2016, the information-based optimal subdata selection (IBOSS) method was introduced, which focuses on D-optimality by selecting covariate extremes deterministically (Wang et al. 2019). IBOSS has recently been extended to logistic regression Cheng et al. (2020) and cluster-wise linear regression (Liu et al. 2023). IBOSS has also been considered in a high-dimensional LASSO framework, where the true set of informative predictors in the dataset are unknown (Singh and Stufken 2023). Optimal subsampling strategies have also been proposed for prediction-oriented criteria. Deldossi and Tommasi (2021) proposed an Optimal Design-Based (ODB) method for finding subsamples that are efficient under the $A-$, $D-$, and $I-$optimality criteria for linear models with constant variance. Cia-Mina et al. (2025)

proposed a subsampling scheme that minimizes the $J-$optimality criterion, which minimizes the expected prediction error for homoscedastic linear models under a Random-X framework, which also treats covariates (in addition to responses) as random variables.

Much work has been done on optimal subsampling for linear models and generalized linear models (GLMs). In a GLM, the variance is not necessarily constant, as the variance of the response depends on the mean of the response via a known link function. However, even in linear models, the true relationship between the covariates and the error variance is not always known, and strategies for optimal subsampling should be explored in this case. In this paper, a subsample of size n (where n is the total number of points in the subsample) is constructed in two stages. The first $n_1 < n$ subsamples are selected using a Latin Hypercube technique. A kernel-based estimation procedure is proposed that uses the first n_1 points in the subsample to quickly estimate the error variances for all points in the full dataset. A weighted IBOSS algorithm is proposed that uses these variance estimates to select the remaining $n_2 = n - n_1$ subsamples in a way that is efficient under the $D-$optimality criterion. Furthermore, an Approximate Nearest Neighbors Simulated Annealing Algorithm (ANNSA) is proposed that takes the weighted IBOSS subsample and chooses the remaining n_2 subsamples to have lower integrated prediction variance. Together, these methods provide tools for modern data analysts to quickly find informative subsamples that can be useful for parameter estimation and also for prediction.

In Section 2, preliminary information on optimal subsampling methods, existing algorithms for finding optimal designs and subsamples, and methods for finding approximate nearest neighbors will be reviewed. In Section 3, new methods for quickly finding subsamples that are efficient under the $D-$ and $I-$criteria will be proposed for linear models with heteroscedasticity. In Section 4, empirical results will show that the proposed subsamples are efficient and produce fitted models with desirable properties. In Section 5, the proposed subsampling methods will be applied to real datasets where there is heteroscedasticity in the variance in linear models. Section 6 concludes the paper.

2. Preliminaries

In order to discuss the proposed methodology, two topics need to be reviewed. Section 2.1 provides an overview of existing optimal subsampling methods. Section 2.2 reviews the concept of a k-d tree, which is a helpful data structure that can be used to quickly identify neighboring rows in a massive dataset. The k-d tree is used in Algorithm 3, which is introduced in Section 3.3.

2.1. Optimal subsampling

To explain optimal subsampling, we must first review criteria for optimal experimental designs. These optimality criteria are often the same criterion used to select a set of points to include in an experimental design. Suppose that the set of available points for fitting a linear regression model is $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. An approximate subsample has the form $\xi = \{(\mathbf{x}_i, \omega_i) : i = 1, \ldots, N\}$, where $\omega_i \in (0, 1)$ for all $i = 1, \ldots, N$ and $\sum_{i=1}^{N} \omega_i = 1$. Each ω_i represents a weight for the point $\mathbf{x}_i$. This work focuses on exact subsamples, which are subsamples of exactly n points chosen from the N available data points. For exact subsamples, $n\omega_i$ is an integer for each $i = 1, \ldots, N$, where $n < N$. In this case, we may write the design matrix as an $n \times (p+1)$ matrix $\mathbf{Z}$, whose rows belong to the set $\{\mathbf{f}(\mathbf{x}_i) \mid \omega_i > 0, 1 \leq i \leq N\}$. An optimal design matrix is $\mathbf{Z}^* = \operatorname{argmin}_{\mathbf{Z} \in \mathcal{D}} \phi(\mathbf{Z})$, where $\phi(\cdot)$ is an optimality criterion, and $\mathcal{D}$ is the space of possible $n \times (p+1)$ design matrices. One of the most popular criterion is $D-$optimality, which maximizes the determinant of the normalized information matrix. If the error variance is constant, then this criterion is $\phi_D(\mathbf{Z}) = |\mathbf{Z}^T\mathbf{Z}/n|^{1/(p+1)}$. This is a popular criterion to use in deterministic or nearly-deterministic computer experiments, where the variance is assumed to be constant. In this case, points would be selected that minimize the generalized variance of the regression coefficients in a linear model (Becerra and Goos 2021; Sambo et al. 2014; Li and Deng 2021). Similarly, other optimality criteria have been examined, such as $A-$optimality in Kiefer (1974), which aims at minimizing the average variance of the parameter estimates. Of more recent interest is $I-$optimality (Goos et al. 2016), which selects designs that minimize the integrated prediction variance over a region of interest $\chi \subseteq \mathbb{R}^p$. For a linear model fit to an approximate subsample, this criterion is $\int_{\mathbf{x} \in \mathcal{X}} \mathbf{f}(\mathbf{x})(\sum_{i=1}^N \omega_i g^{-1}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)^T)^{-1}\mathbf{f}(\mathbf{x})\, d\mathbf{x}$, where $g(\mathbf{x}_i)$ is the variance of the response at $\mathbf{x}_i$. In the case of exact subsampling, this simplifies to $\phi_I(\mathbf{Z}) = \int_{\mathbf{x} \in \mathcal{X}} \mathbf{f}(\mathbf{x})^T(\mathbf{Z}^T\mathbf{\Sigma}^{-1}\mathbf{Z}/n)^{-1}\mathbf{f}(\mathbf{x})\, d\mathbf{x}$, where $\mathbf{\Sigma} = \operatorname{diag}(g(\mathbf{z}_1), \ldots, g(\mathbf{z}_n))$.

Optimal subsampling aims to select a "best" subsample with respect to an information-based optimality criterion. Many existing optimal subsampling methods focus on $D-$optimality. Wang et al. (2019) proposed an Information-Based Optimal Subsampling (IBOSS) method for linear regression models that can find subsamples of size $n < N$. The IBOSS algorithm for linear regression models examines each covariate of the full dataset, selects rows with the $r = \lfloor n/(2p) \rfloor$ lowest and r highest values of the covariate, and places the $2r$ rows into the subsample. The idea behind IBOSS was to select extreme points for the subsample, as these provide the most information for fitting a linear model and would be the closest to maximizing the $D-$optimality criterion. IBOSS has been extended to logistic regression models (Wang et al. 2018; Cheng et al. 2020), cluster-wise linear regression models (Liu et al. 2023), and LASSO models (Singh and Stufken 2023). Recently, Deldossi and Tommasi (2021) proposed Optimal-Design Based (ODB) subsampling, which constructs optimal subsamples in two phases. In the first phase, an optimal approximate design with $k < n$ support points is constructed via the R package **OptimalDesign** (Harman and Filova 2025). If these weights do not form an exact design, a rounding procedure is applied. In the second phase, the subsample is formed by selecting points that are the closest to each of the k support points in the subsample.

There are several similarities between finding optimal subsamples for GLMs and linear models with heteroscedasticitiy. This work focuses on studying linear models where $\operatorname{Var}[y \mid \mathbf{x}] = g(\mathbf{x})$ for an unknown function g. In this case, an approximate subsample has an information matrix of the form

$$\sum_{i=1}^N \omega_i g^{-1}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)^T, \tag{1}$$

where $\sum_{i=1}^N \omega_i = 1$ and $\omega_i \geq 0$ for each $i = 1, \ldots, N$. In the case of a GLM, the information matrix is

$$\sum_{i=1}^{N} \omega_i v_i \mathbf{f}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)^T, \tag{2}$$

where $v_i = (\partial\mu_i/\partial\mathbf{x}_i^T\boldsymbol{\beta})^2/\text{Var}[y_i \mid \mathbf{x}_i]$. This has motivated several two-step approaches to optimal subsampling for GLMs (Wang et al. 2018; Cheng et al. 2020), where in the first step, a random sample is used to estimate $\boldsymbol{\beta}$, and then this estimate is used to find an optimal subsample. The main difference is that in (2), each point is weighted by an unknown variance function g, which does not have a known parametric form.

2.2. k-d Trees

A k-d tree (short for k-dimensional tree) is a data structure which stores k-dimensional points as nodes of a binary tree, containing information to be retrieved by associative search. To construct a k-d tree, the point closest to the median value among the first coordinates of all points is selected as a root node. The remaining points are split into two subtrees. The left subtree contains points whose first coordinate is less than the median, and the right subtree contains the points whose first coordinates are greater than the median. The procedure is repeated for both subtrees, using the median of the second coordinate to find the root node of each subtree, and splitting the remaining nodes into additional left and right subtrees. This procedure continues recursively (looping back to the first coordinate after the k^{th} coordinate is used) until all nodes are placed in the tree. An example of a k-d tree constructed for 9 points in a 2-dimensional space is given in Figure 1. More details on these data structures can be found in Mangla (2024).

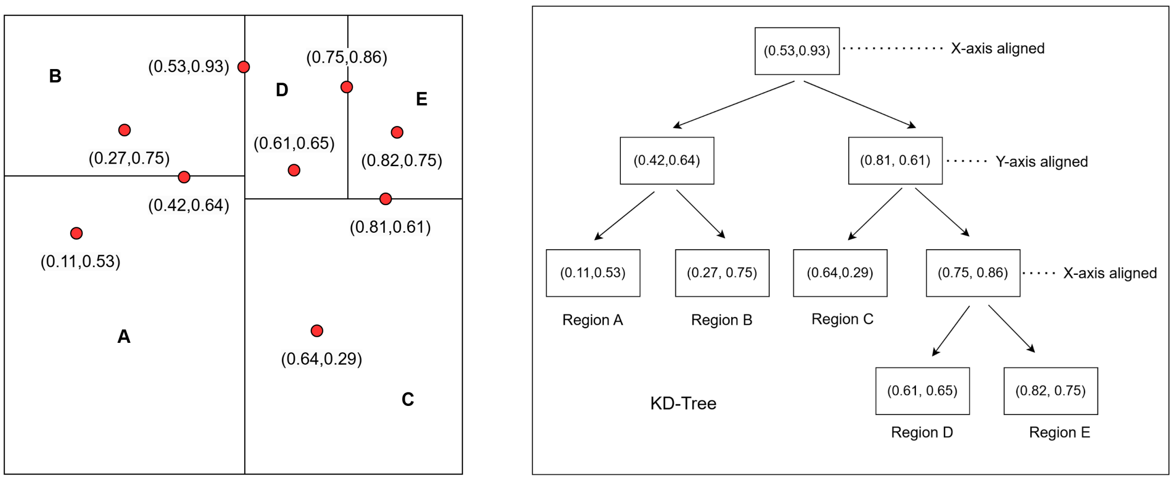

Figure 1: An example of a k-d tree for $k = 2$ with 9 points. Figures are taken from (Mangla 2024).

This data structure is a computationally efficient way to identify approximate neighbors for a set of points in k-dimensional space. For huge and high-dimensional datasets, it is computationally inefficient to compute the Euclidean distance between every pair of data points. Instead, Approximate-Nearest Neighbors (A-NN) can be found for a given point. Let $\mathbf{x} \in \mathbb{R}^k$. To find the approximate nearest neighbors of $\mathbf{x}$, the A-NN algorithm attempts to insert $\mathbf{x}$ into a leaf node of the k-d tree. This can be done by comparing the first coordinate of $\mathbf{x}$ to the root node; if it is less than the root node, it goes into the left subtree; otherwise, it goes into the right subtree. This continues recursively (using the appropriate coordinates of $\mathbf{x}$) until $\mathbf{x}$ is placed in a leaf node of the k-d tree. Once $\mathbf{x}$ is placed in a leaf node, the algorithm selects points in the current node that are within a search radius of $\mathbf{x}$. The A-NN algorithm also considers points in nearby nodes if the distance between $\mathbf{x}$ and the boundary to the node is less than the radius times $1/(1+\alpha)$, where $\alpha > 0$ and larger values of α give a smaller search radius. Once this is complete, a set of the ℓ closest points (of the selected points) to $\mathbf{x}$ are returned by the A-NN algorithm. In this article, the k-d Trees are built with Python of **scipy.spatial.cKDTree** function from Virtanen et al. (2020) library.

3. Proposed Methods

Suppose we have a full dataset $(\mathbf{x}_i, y_i), i = 1, \ldots, N$, where N is very large, $\mathbf{x}_i = [x_{i1}, \ldots, x_{ip}]^T$ is a vector of p real covariates, and $y_i \in \mathbb{R}$ is a univariate response. Let $\mathbf{f}(\mathbf{x}_i) = [1, \mathbf{x}_i^T]^T$. Let the full matrix of covariates be denoted as $\mathbf{X} = [\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_N)]^T$. Suppose that the data follow the model

$$y_i = \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i, \quad E[\epsilon_i] = 0, \mathrm{Var}[\epsilon_i] = \sigma_i^2, \quad i = 1, \ldots, N, \tag{3}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$ is an unknown vector of regression coefficients and $\sigma_i^2 = g(\mathbf{x}_i)$. That is to say, the variance of the error terms depends on a subset of the covariates in the dataset. In practice, the function g that relates the variance to the covariates is generally unknown, which creates difficulty in selecting a subsample to estimate the unknown coefficients.

If g were known, one could re-weight the points accordingly and use weighted least squares to estimate the unknown parameter vector $\boldsymbol{\beta}$. Let $(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_n, y_n)$ be a subsample of size n from the full data. Let $\mathbf{Z} = [f(\mathbf{z}_1), \ldots, f(\mathbf{z}_n)]^T$, with corresponding responses $\mathbf{y} = [y_1, \ldots, y_n]^T$. Let $\boldsymbol{\Sigma} = \mathrm{diag}(g(\mathbf{z}_1), \ldots, g(\mathbf{z}_n))$. Then, the optimal weighted least squares estimator for the coefficients is $\hat{\boldsymbol{\beta}}_{wls} = (\mathbf{Z}^T\boldsymbol{\Sigma}^{-1}\mathbf{Z})^{-1}\mathbf{Z}^T\boldsymbol{\Sigma}^{-1}\mathbf{y}$, which has a normalized information matrix of $(1/n)\mathbf{Z}^T\boldsymbol{\Sigma}^{-1}\mathbf{Z}$. It is straightforward to see that the $D-$optimality criterion for a subsample $\mathbf{Z}$ is $\phi_D(\mathbf{Z}) = |(1/n)\mathbf{Z}^T\boldsymbol{\Sigma}^{-1}\mathbf{Z}|^{1/(p+1)}$. Suppose we wanted to evaluate the $I-$optimality criterion over an experimental region $\mathcal{X} \subset \mathbb{R}^p$. Using this estimator, the $I-$optimality criterion can be written as

$$\phi_I(\mathbf{Z}) = \int_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x})^T \left(\frac{\mathbf{Z}^T\boldsymbol{\Sigma}^{-1}\mathbf{Z}}{n}\right)^{-1} \mathbf{f}(\mathbf{x})\, dx = \mathrm{trace}\left(\left(\frac{\mathbf{Z}^T\boldsymbol{\Sigma}^{-1}\mathbf{Z}}{n}\right)^{-1}\mathbf{B}\right), \tag{4}$$

where $\mathbf{B} = \int_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x})^T\, d\mathbf{x}$. Again, lower values of the $I-$optimality criterion are desirable, since they correspond to smaller prediction variance over the region of interest. In this paper, it is assumed that the region of interest $\mathcal{X}$ is the region occupied by the rows of the full dataset $\mathbf{X}$; i.e., we wish to make accurate predictions for any point in the neighborhood of any point currently in the full dataset. Formally, this region can be written as $\mathcal{X} = \cup_{i=1}^N \{\mathbf{x} \in \mathbb{R}^p \mid ||\mathbf{x} - \mathbf{x}_i||_2^2 \leq \delta\}$ for some small $\delta > 0$. In this case, we approximate $\mathbf{B}$ by replacing the integral with the following average over the set of full data points:

$$\mathbf{B} \approx \frac{1}{N}\sum_{i=1}^N \mathbf{f}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)^T. \tag{5}$$

The matrix $\mathbf{B}$ only needs to be computed once for the full dataset, and then it can be used in subsequent methods. For ease of notation, equation (5) will be used as the definition of $\mathbf{B}$ from here onward.

Suppose that the variance of the error terms can be estimated. Let $\hat{\mathbf{\Sigma}} = \text{diag}(\hat{g}(\mathbf{z}_1), \ldots, \hat{g}(\mathbf{z}_n))$. Then, the $I-$optimality criterion of interest becomes

$$\phi_I(\mathbf{Z}) = \text{trace}\left(\left(\frac{\mathbf{Z}^T\hat{\mathbf{\Sigma}}^{-1}\mathbf{Z}}{n}\right)^{-1}\mathbf{B}\right). \tag{6}$$

This work will focus on answering three questions:

1. How can we efficiently and effectively estimate the error variances $\sigma_1^2, \ldots, \sigma_N^2$?
2. How can we find subsamples (of a given size $n_2 < N$) that have very high $D-$optimality?
3. How can we find subsamples (of a given size $n_2 < N$) that have very low $I-$optimality?

3.1. Estimation of variances

In order to find optimal subsamples, variance estimates $\hat{\sigma}_i^2, i = 1, \ldots, N$ need to be computed. The main idea is to use a space-filling technique to select the first n_1 points of the subsample, fit a local linear model to the first n_1 points of the subsample, and then use the fitted model to construct a weighted estimator for the variance of an arbitrary point in the full dataset. To do this, Algorithm 1 is used. This algorithm first uses a Latin Hypercube Design McKay et al. (1979) to select n_1 points $\mathbf{d}_1, \ldots, \mathbf{d}_{n_1}$ in the space of data points χ. The LHD partitions χ into n_1 equally sized blocks. LHDs have the property that, when projected into any of the p dimensions, there is exactly one observation in each of the n_1 blocks. This makes them well-suited for the estimation of unknown functions (Stein 1987), such as the unknown variance function $g(\cdot)$. A random LHD is drawn by selecting (at random) a point inside each of the n_1 selected blocks. For each $i = 1, \ldots, n$, a k-d tree is used to find the approximate nearest neighbor of $\mathbf{d}_i$ from the data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. This is how the first n_1 points in the subsample are selected. Denote the first n_1 points of the subsample as $\{(\tilde{\mathbf{x}}_i, \tilde{y}_i)_{i=1}^{n_1}\}$, and for any $\mathbf{x} \in \mathbb{R}^p$, let

$$m(\mathbf{x}) = E(y|\mathbf{x}) \tag{5}$$

$$K_H(\tilde{\mathbf{x}}_i - \mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{H}|^{1/2}} \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}}_i - \mathbf{x})^\top \mathbf{H}^{-1}(\tilde{\mathbf{x}}_i - \mathbf{x})\right), \tag{6}$$

where $\mathbf{H} = h^2\mathbf{I}_p$ is a diagonal matrix that depends on a scalar bandwidth $h > 0$. The kernel function $K_H(\cdot)$ assigns weights to the sub-sample points as a function of their Mahalanobis distance from the evaluation point $\mathbf{x}$. Algorithm 1 is motivated by (Fan and Yao 1998), an essential part of the algorithm is to perform local linear regression on the random subsample to estimate the conditional expectation $m(\mathbf{x}_i) = E(y|\mathbf{x}_i)$ for each $i = 1, \ldots, n_1$, which has been learned as a consistent estimator. This is done by solving

$$(\hat{a}_i, \hat{\mathbf{b}}_i) = \underset{a,\mathbf{b}}{\operatorname{argmin}} \sum_{j=1}^{n_1} \{\tilde{y}_j - a - \mathbf{b}(\tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i)\}^2 \cdot K_H(\tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i), \tag{7}$$

for each $i = 1, \ldots, n_1$. This is a least squares problem with a known solution (shown in Step 5 of Algorithm 1). Using the solution in (7), the conditional means can be estimated as $\hat{m}(\tilde{\mathbf{x}}_i) = \hat{a}_i$ for each $i = 1, \ldots, n_1$. Algorithm 1 then computes the squared residuals $\hat{r}_i^2 = (\tilde{y}_i - \hat{m}(\tilde{\mathbf{x}}_i))^2$ for $i = 1, \ldots, n_1$.

Compared to the variance estimation procedure from (Fan and Yao 1998), the variance $\sigma^2(\mathbf{x})$ is approximated via residual smoothing using Nadaraya-Watson Estimator (NWE) for computational efficiency. Let $\hat{\sigma}_H^2(\mathbf{x})$ be a weighted average of the squared estimated residuals, i.e.,

$$\hat{\sigma}_H^2(\mathbf{x}) := \frac{\sum_{i=1}^{n_1} K_H(\tilde{\mathbf{x}}_i - \mathbf{x}) \cdot \hat{r}_i^2}{\sum_{j=1}^{n_1} K_H(\tilde{\mathbf{x}}_j - \mathbf{x})}. \tag{8}$$

The estimator in (8) is based on the Nadaraya-Watson Estimator (NWE) for estimating the conditional mean (Nadaraya 1964). The traditional NWE is the same as (8), except $\hat{r}_i^2$ is replaced with $\tilde{y}_i$. The modified NWE can be used to find $\hat{\sigma}_H^2(\mathbf{x}_i)$ for all points $i = 1, \ldots, N$ in the full dataset.

Algorithm 1: Data-driven variance estimation.

```
Inputs: Data $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$, a scalar $h > 0$, $n_1$, and a k-d tree KD for the full data
  $\mathbf{X}$.
1. Generate a Latin Hypercube Design in $\chi$ with $n_1$ points, denoted as $\mathbf{d}_1, \ldots, \mathbf{d}_{n_1}$.
  For $i = 1, \ldots, n_1$, use the k-d tree KD to find the nearest neighbor $\tilde{\mathbf{x}}_i$ of $\mathbf{d}_i$ and
  the corresponding response $\tilde{y}_i$. Denote the subsample as $\{(\tilde{\mathbf{x}}_i, \tilde{y}_i)_{i=1}^{n_1}\}$.
2. Construct the diagonal matrix $\mathbf{H} = \mathrm{diag}(h^2, \cdots, h^2)_{n_1 \times n_1}$;
for $i = 1$ to $n_1$ do
    3. Let $\mathbf{W}_i = \mathrm{diag}(K_H(\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_i), \ldots, K_H(\tilde{\mathbf{x}}_{n_1} - \tilde{\mathbf{x}}_i))$.
    4. Let $\tilde{\mathbf{X}}_i = [\mathbf{1}, (\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_i, \ldots, \tilde{\mathbf{x}}_{n_1} - \tilde{\mathbf{x}}_i)^T]$ and let $\tilde{\mathbf{y}} = [\tilde{y}_1, \ldots, \tilde{y}_{n_1}]^T$.
    5. Calculate the closed-form solution
       $$\begin{bmatrix} \hat{a}_i \\ \hat{\mathbf{b}}_i \end{bmatrix} = \left(\tilde{\mathbf{X}}_i^T \mathbf{W}_i \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}_i^T \mathbf{W}_i \tilde{\mathbf{y}}$$
     and set $\hat{m}(\tilde{\mathbf{x}}_i) := \hat{a}_i$;
    6. Store the squared residual $\hat{r}_i^2 = (y_i - \hat{m}(\tilde{\mathbf{x}}_i))^2$.
end
7. For $i = 1, \ldots, N$, compute $\hat{\sigma}_H^2(\mathbf{x}_i)$ using $\hat{r}_1^2, \ldots, \hat{r}_{n_1}^2$ and the expression in (8).
Return: The estimated variances $\hat{\sigma}_H^2(\mathbf{x}_i)$ for $i = 1, \ldots, N$.
```

The bandwidth h is a tuning parameter for the variance estimation. There are multiple choices for selecting h. In general, points that are within a distance of h of $\mathbf{x}$ will have higher weight than points that are further away from $\mathbf{x}$. In this sense, the kernel estimates become "smoother" as h increases. There is extended research in the bandwidth selection using a cross-validation scheme, or a plug-in estimators. In Theorem 2 of Greblicki and Krzyzak (1980), it was proven that (under regularity conditions satisfied by the Gaussian kernel) if $h \to 0$ as $n_1 \to \infty$ and $n_1 h \to \infty$ as $n_1 \to \infty$, then the traditional Nadaraya-Watson Estimator is a consistent estimator of the conditional mean of the response. This is satisfied by several choices of h, such as $h = \mathcal{O}(n_1^{-1/(p+4)})$, which was also found to be an optimal bandwidth in spatial kernel estimation problems (Van Lieshout 2020). This provides motivation for using the modified NWE in Equation (8), since the variance of the true error terms are equal to the conditional means of the squared true error terms.

3.2. A weighted IBOSS algorithm

In this section, a Weighted IBOSS algorithm is proposed. This algorithm aims to find a subsample that is close to $D-$optimal. This means it aims to minimize the generalized variance of the Weighted Least Squares (WLS) estimator $\hat{\beta}_{WLS}$, which is given by

$$|\operatorname{Var}(\hat{\beta}_{WLS})| = |(\mathbf{Z}^T\hat{\mathbf{\Sigma}}^{-1}\mathbf{Z})^{-1}|. \tag{7}$$

This is equivalent to maximizing the determinant of the information matrix. This optimization can be written in the following form:

$$\max_{\mathbf{Z}} |(\mathbf{Z}^T\hat{\mathbf{\Sigma}}^{-1}\mathbf{Z})| = \max_{\mathbf{Z}} |(\hat{\mathbf{\Sigma}}^{-1/2}\mathbf{Z})^T(\hat{\mathbf{\Sigma}}^{-1/2}\mathbf{Z})|. \tag{8}$$

If the data are homoscedastic, then $\mathbf{\Sigma} = \sigma^2\mathbf{I}_n$ and the expression in (8) reduces to maximizing $|\mathbf{Z}^T\mathbf{Z}|$. This is the same problem that is addressed by the IBOSS algorithm in Wang et al. (2019). Therefore, the proposed method is to re-weight each observation by the reciprocal square root of its estimated error variance. This is a generalization of IBOSS that accounts for the fact that observations with smaller error variance contain more information.

Algorithm 2: Weighted IBOSS algorithm for subsampling.

```
Inputs: full $(N \times (p+1))$ data matrix $\mathbf{X} = [\mathbf{f}(\mathbf{x}_1), \ldots, \mathbf{f}(\mathbf{x}_N)]^T$, error variance
  estimates $\hat{\sigma}_i^2$, $i = 1, \ldots, N$, target subsample size $n$, current subsample indices $S$
  $= (i_1, \ldots, i_{n_1})$.
for $i = 1, \ldots, N$ do
    1. Let $\mathbf{x}_i^* = \mathbf{x}_i/\hat{\sigma}_i \triangleq (x_{i1}^*, \ldots, x_{ip}^*)$.
end
2. Let $r = \lfloor n_2/2p \rfloor$, where $n_1$ is the length of $S$ and $n_2 = n - n_1$.
for $j = 1, \ldots, p$ do
    3. Find the indices of the rows in $\mathbf{X}$ containing the $r$ smallest values of $(x_{1j}^*,
    \ldots, x_{Nj}^*)$. Add these indices to $S$.
    5. Find the indices of the rows in $\mathbf{X}$ containing the $r$ largest values of $(x_{1j}^*, \ldots,
    x_{Nj}^*)$. Add these indices to $S$.
end
Return $\mathbf{Z} = [\mathbf{f}(\mathbf{x}_{i_1}), \ldots, \mathbf{f}(\mathbf{x}_{i_n})]^T$, where $S = (i_1, \ldots, i_n)$.
```

As inputs, the Algorithm 2 takes the full covariates matrix $\mathbf{X}$, the estimated error variance, $\hat{\sigma}_i^2$ for each row of $\mathbf{X}$, the current row indices of the subsample, denoted as $S = (i_1, \ldots, i_{n_1})$, and the desired subsample size, n. In Steps 1 and 2 of the algorithm, each point $\mathbf{x}_i$ is divided by its corresponding estimated standard deviation $\hat{\sigma}_i$. Hence, Algorithm 2 depends on the quality of the error variance estimator. For example, if the error variance is severely underestimated for a certain point, then Algorithm 2 will give a much higher weight to this point's contribution to the overall information in the subsample. From Steps 3–6, the IBOSS algorithm developed by Wang et al. (2019) is used to select the remaining points. Specifically, in Step 3, a value $r = (n_2)/(2p)$ is determined. This r represents the number of observations to be selected from both the highest and lowest tails for each of the p covariates. Steps 4 and 5 loop over each of the p covariates and repeat the process. The algorithm implicitly selects rows without replacement for each covariate to achieve total subsample size of n. In Step 6, the algorithm returns the subsample covariate matrix $\mathbf{Z}$.

There is some theoretical motivation for using Algorithm 2. Let $\sigma(\mathbf{x}) = \sqrt{g(\mathbf{x})}$. Suppose that, for a subsample $\mathbf{Z}$, the true error standard deviations $\sigma(\mathbf{z}_1), \ldots, \sigma(\mathbf{z}_n)$ were known. Then, the information matrix of the subsample can be expressed as $\mathbf{M}^* = \mathbf{Z}^{*T}\mathbf{Z}^*$, where $\mathbf{Z}^* = [\mathbf{f}(\mathbf{z}_1)/\sigma(\mathbf{z}_1),\ \mathbf{f}(\mathbf{z}_2)/\sigma(\mathbf{z}_2),\ \ldots,\ \mathbf{f}(\mathbf{z}_n)/\sigma(\mathbf{z}_n)]^T$. An upper bound of the determinant of the moment matrix of the subdata $\mathbf{M}^*$ can be found by applying Theorem 2 from Wang et al. (2019) by replacing $\mathbf{x}_i$ with $\mathbf{x}_i^*, i = 1, \ldots, N$ and removing the constant variance σ^2, because the variance terms are absorbed into $\mathbf{Z}^*$. This allows the upper bound to be re-expressed as:

$$|\mathbf{M}^*| \leq \frac{n^{p+1}}{4^p} \prod_{j=1}^{p} (x^*_{(n)j} - x^*_{(1)j})^2. \tag{9}$$

Similar to Theorem 2 of Wang et al. (2019), this result suggests that $D-$optimal subsamples can be obtained by selecting rows that contain extreme values of the standardized covariates. In practice, the true error standard deviations used to standardize the covariates are unknown, but they can be estimated using the methods proposed in Section 3.1.

3.3. Approximated nearest neighbor simulated annealing

In this section, a modified Simulated Annealing (SA) algorithm is proposed to efficiently search for subsamples that optimize a criterion ϕ. Simulated Annealing is a metaheruistic optimization algorithm that was originally proposed by Kirkpatrick et al. (1983). SA is a probabilistic algorithm that searches for values that minimize an objective function (Bertsimas and Tsitsiklis 1993). At a high level, SA starts with an initial solution, and then randomly generates a neighboring solution. If the neighboring solution is better, then it is accepted; otherwise it is accepted with a probability that decreases as the algorithm runs for more iterations. Accepting suboptimal solutions early on in the search helps avoid local optima.

In the context of subsampling, the SA algorithm starts with a random subsample, i.e., a set of n row indices from $\{1, \ldots, N\}$. To generate a neighboring solution, a random row index in the current subsample is selected and exchanged with the row index of a nearby point. The difficulty lies in finding a neighborhood of points in the dataset for a particular row of the current subsample. It is computationally inefficient to find the distance between all $\binom{N}{2}$ pairs of points in the full dataset. Therefore, Approximate Nearest Neighbors (A-NN), described in Section 2.2, is used to quickly define an approximate neighborhood for a given row of $\mathbf{X}$. This algorithm, called Approximated Nearest Neighbor Simulated Annealing (ANNSA), is summarized in Algorithm 3.

Algorithm 3: Approximated nearest neighbor simulated annealing.

```
Input: Covariate matrix $\mathbf{X} \in \mathcal{M}_{N,(p+1)}$, $\mathbf{Z}_1 \in \mathcal{M}_{n_1,(p+1)}$ and $\mathbf{Z}_2^{(1)} \in \mathcal{M}_{n_2,(p+1)}$,
  where $\mathcal{M}_{n,p}$ is the set of all $n \times p$ matrices, and the rows of $\mathbf{Z}_1$ and $\mathbf{Z}_2^{(1)}$ are
  drawn from the rows of $\mathbf{X}$.
1. Generate a k-d tree for the full data $\mathbf{X}$.
for $i = 1, 2, \dots, i_{max}$ do
    2. Randomly select a data row $\mathbf{f}(\mathbf{z})$ from $\mathbf{Z}_2^{(i)}$, and obtain 100 nearest neighbors
      of it with the k-d tree.
    3. Randomly select a data row $\mathbf{f}(\mathbf{z}^*)$ from the 100 rows, then swap it with $\mathbf{f}(\mathbf{z})$
      to form $\mathbf{Z}_2^{(i*)}$.
    4. Compute $\phi(\mathbf{Z}_2^{(i)})$ and $\phi(\mathbf{Z}_2^{(i*)})$.
    5. Generate $U \sim U(0,1)$.
    if $U < \exp(-(\phi(\mathbf{Z}_2^{(i)}) - \phi(\mathbf{Z}_2^{(i)}))(i+1))$ then
        $\mathbf{Z}_2^{(i+1)} = \mathbf{Z}_2^{(i*)}$
    else
        $\mathbf{Z}_2^{(i+1)} = \mathbf{Z}_2^{(i)}$
    end
end
6. Store the best $n - n_1$ points so far as $\mathbf{Z}_2^*$ and the corresponding efficiency $\phi^* =$
$\phi(\mathbf{Z}_2^*)$.
return $\mathbf{Z}^* = [\boldsymbol{Z}_1^T, (\boldsymbol{Z}_2^*)^T]^T$.
```

Algorithm 3 is written to find a minimum value of $\phi(\cdot)$, i.e., it is written so that lower values of ϕ are more optimal, which is the case for the $I-$efficiency criterion. This algorithm can easily be modified for the $D-$efficiency criterion by minimizing the negative $D-$efficiency. As an input, Algorithm 3 takes an $n_1 \times (p+1)$ matrix $\mathbf{Z}_1$. This represents the partial subsample collected to estimate the variance. The other input is an $(n_2) \times (p+1)$ matrix $\mathbf{Z}_2^{(1)}$. This matrix can be randomly sampled from the full dataset, or can be selected using the IBOSS Algorithm.

In Step 1 of the algorithm, a k-d tree is generated for the full data $\mathbf{X}$. Then, Steps 2–5 of Algorithm 3 are conducted iteratively for $i = 1, \dots, i_{max}$. In Step 2, one row of the current subsample $\mathbf{Z}^{(i)}$ is randomly selected (and denoted as $\mathbf{f}(\mathbf{z})$) from the last $n - n_1$ rows of $\mathbf{Z}^{(i)}$, and the k-d tree is used to find its 100 approximated nearest neighbors. Next, one of the neighboring rows $\mathbf{f}(\mathbf{z}^*)$ is randomly selected and swapped with $\mathbf{f}(\mathbf{z})$ to obtain a new subsample $\mathbf{Z}^{(i*)}$. The $\phi-$efficiencies of $\mathbf{Z}^{(i)}$ and $\mathbf{Z}^{(i*)}$ are computed respectively and a uniform random number U is generated. If $U < \mathbf{exp}(-(\phi(\mathbf{Z}^{(i)}) - \phi(\mathbf{Z}^{(i*)}))i)$, then the swap would be kept, i.e. $\mathbf{Z}^{(i+1)} = \mathbf{Z}^{(i*)}$. Otherwise the old design would be kept for the next iteration. After i_{max} iterations, the subsample $\mathbf{Z}^*$ throughout all iterations with the smallest value of ϕ is kept as the final subsample.

4. Simulation Results

4.1. Variance estimation

In this section, we apply Algorithm 1 to estimate variances $\hat{g}(\mathbf{x}_i), i = 1, \ldots, N$ under heteroskedasticity, using three distinct variance functions to evaluate the performance of the estimation procedure. For a given subsample size $n \ll N$, and number of predictors p, the data are generated as follows. The covariates $x_{ij} \sim U(0,1)$ for $i = 1, \ldots, N$ and $j = 1, \ldots, p$. The regression coefficients β_0 and $\boldsymbol{\beta} \in \mathbb{R}^p$ are each independently sampled from a $U(0.3, 1)$ distribution, rounded to one decimal place. The outcome variable y_i is then generated according to Model (3) with $\epsilon_i \sim \mathcal{N}(0, g_k(\mathbf{x}_i))$, where $g_k(\mathbf{x})$ is one of four variance functions:

1. $g_1(\mathbf{x}) = (1/2p)\sum_{j=1}^{p}(x_j - 0.9)^2$
2. $g_2(\mathbf{x}) = (1/2p)\sum_{j=1}^{p}(x_j - 0.1)^2$
3. $g_3(\mathbf{x}) = (1/2)x_1^2 + (1/10)x_2^2 + (1/5)x_p^2$
4. $g_4(\mathbf{x}) = 1$,

and x_j denotes the jth element of $\mathbf{x}$. The functions g_2 and g_3 represent cases where the variance of the errors increases as certain values of $\mathbf{x}$ increase. In g_3, the variance of the errors only depends on 3 of the p total covariates, while in the g_2, the variance depends on all covariates. In g_1, the variance of the errors increases the further away each covariate is from 0.9. Since the covariates are each $U(0,1)$ random variables, this will assign smaller variances to points whose entries are larger. Finally, the function g_4 assigns constant variance to all points. This function was used to see how well the methods perform in the case where there is no heterogeneity. We compare the performance of the variance estimates $\hat{g}(\mathbf{x}_i)$ across varying subsample sizes $n_1 =$ 100,200,300 and predictor dimensions $p = 5, 10$. The full dataset $\mathcal{D}_N$ consists of $N =$ 1,000,000 observations. The bandwidth was chosen to be $h = n_1^{-1/(p+5)}$.

For each generated dataset, 100 subsamples,each of size n_1, were taken using a Latin Hypercube Design, as described in Section 3.1. For each subsample, the value of $\text{median}_{i=1,\ldots,N}|\hat{g}(\mathbf{x}_i) - g(\mathbf{x}_i)|$ was found; denote this as the Median Absolute Error (MAE).

As shown in Table 1, when the initial subsample size n_1 increased, the MAEs decreased. When the number of covariates increases from $p = 5$ to $p = 10$, the MAE tends to increase. The average CPU times for estimating the variances on all $N =$ 1,000,000 points were quite fast, especially for smaller values of n_1. There is a trade-off between computation time and accuracy; smaller values of n_1 leads to less CPU time, but at the cost of higher MAEs. The MAEs were the lowest for variance functions g_1 and g_3, with MAEs less than 0.10 for $n_1 \geq 200$ in these cases. Across all four considered variance functions, the highest MAE occurs for the function g_4 when $n_1 = 100$. When the variance function is g_4, the errors are homoscedastic, so one might expect that the proposed method is not accurate. However, when n_1 increases from 100 to 200 (and then 200 to 300), the average MAE is shown to decrease, leading to more accurate

Table 1: Median Absolute Errors (MAEs) and CPU times (in seconds) for variance estimation for $n_1 = 100, 200, 300$ and $p = 5, 10$ for four variance functions.

		g_1		g_2		g_3		g_4	
n_1	p	MAE	CPU	MAE	CPU	MAE	CPU	MAE	CPU
100	5	0.04	5.95	0.15	6.11	0.10	5.75	0.18	5.71
200	5	0.03	12.08	0.14	12.09	0.10	10.84	0.12	11.56
300	5	0.03	18.07	0.14	17.38	0.10	16.33	0.09	17.25
100	10	0.05	6.34	0.33	5.62	0.11	5.63	0.42	5.66
200	10	0.04	12.70	0.24	11.10	0.10	11.16	0.27	11.20
300	10	0.03	19.18	0.23	17.31	0.10	16.92	0.21	16.97

predictions. These results are achieved using values of n_1 that are $0.01\%, 0.02\%$, and 0.03% of N, respectively for n_1 = 100,200,300.

4.2. Optimality comparison of subsampling methods

In this section, the IBOSS, Weighted IBOSS, ANNSA, and ODB subsampling methods will be compared in terms of their $D-$ and $I-$optimality criteria. These comparisons were made using the four variance functions (g_1, g_2, g_3, g_4) discussed in Section 3.1. For each subsample found, the $D-$ and $I-$optimality criteria were then evaluated using the true variance functions. As a reminder, subsamples with higher $D-$ optimality and lower $I-$optimality criteria are preferred. The data were generated as in Section 4.1, with N = 1,000,000 observations in the full dataset. The first n_1 points in the Weighted IBOSS and ANNSA subsamples were found using the methods described in Section 3.1. These n_1 points were used with Algorithm 1 to estimate the variances, and then the remaining n_2 points were found using Algorthms 2 and 3 for Weighted IBOSS and ANNSA, respectively. We used $n_1 = 100$ for the first subsampling phase and $n_2 = 4p, 6p$ for the second (optimal) subsample. These are very small subsets of the full dataset; $n_1 = 100$ is 0.01% of N. As a reminder, the covariates were $x_{ij} \sim U(0, 1)$ for $i = 1, \ldots, N$ and $j = 1, \ldots, p$. The bandwidth here was chosen to be $h = n_1^{-1/(p+5)}$, which is the same bandwidth used in Section 4.1.

For each combination of n, p, and true variance function, a dataset of size N was simulated using the methods described in Section 4.1. The IBOSS and ODB algorithms were used to find a subsample of size n_2, and their $D-$ and $I-$optimality criteria were calculated. For each dataset, 20 subsamples of size n were found using the Weighted IBOSS and ANNSA algorithms. The ANNSA algorithm used the $I-$optimality criterion. Then, the median $D-$ and $I-$optimality criteria of Weighted IBOSS and ANNSA were compared to those of IBOSS. ANNSA used i_{max} = 5,000 iterations and 100 nearest neighbors. The simulation was conducted this way because IBOSS is a deterministic algorithm—given the same dataset, it will always produce the same subsample. However, the subsamples produced by the Weighted IBOSS and ANNSA algorithms can change depending on the variance estimates, which depend on the first n_1 points in the subsample that are selected by a random Latin Hypercube design. The simulation results are summarized in Table 2.

Table 2: Susbample optimality comparison, N = 1,000,000, $n_1 = 200$, $p = 5, 10$, $n_2 = 4p, 6p$. WIBOSS stands for "Weighted IBOSS." ANNSA was run using the $I-$optimality criterion. Larger $D-$optimality values are better, and lower $I-$optimality values are better.

			D-optimality				I-optimality			
p	n_2	g	IBOSS	WIBOSS	ANNSA	ODB	IBOSS	WIBOSS	ANNSA	ODB
5	20	g_1	1.42	2.59	2.00	1.46	0.63	0.84	0.58	1.69
5	20	g_2	0.98	2.83	1.10	0.30	1.65	0.91	1.18	6.77
5	20	g_3	0.71	3.88	1.31	0.56	1.38	1.04	1.05	5.27
5	20	g_4	0.14	0.14	0.19	0.21	6.31	8.64	4.33	9.03
5	30	g_1	1.39	2.75	1.79	1.60	0.63	0.70	0.61	1.18
5	30	g_2	0.91	3.07	1.41	0.28	1.48	0.67	1.01	11.03
5	30	g_3	0.78	4.77	2.95	0.56	1.21	0.86	0.84	3.72
5	30	g_4	0.15	0.14	0.16	0.19	5.29	8.11	4.81	12.97
10	40	g_1	0.91	0.92	1.19	1.20	1.55	1.93	1.07	1.61
10	40	g_2	0.18	0.42	0.25	0.17	8.85	5.17	5.39	13.95
10	40	g_3	0.63	2.45	1.19	0.87	2.52	1.39	1.55	3.79
10	40	g_4	0.10	0.10	0.17	0.18	13.60	16.02	7.13	11.11
10	60	g_1	1.05	1.13	1.18	1.18	1.22	1.52	1.07	1.80
10	60	g_2	0.17	0.38	0.23	0.19	8.13	5.01	5.41	8.30
10	60	g_3	0.54	4.26	1.24	1.03	2.50	1.01	1.36	2.55
10	60	g_4	0.12	0.11	0.17	0.18	10.31	13.18	7.09	9.31

Table 2 shows the median $D-$ and $I-$optimality criteria for several subsamples of size n found using the Weighted IBOSS (WIBOSS) and ANNSA algorithms, and compares them to the $D-$ and $I-$optimality criteria of a subsample found using IBOSS and ODB on the same dataset. When the variance function was g_1, g_2, or g_3, the Weighted IBOSS algorithm always provided subsamples with higher $D-$optimality than those provided by IBOSS and ODB. This is consistent with the theoretical bound in Equation (9), which argued that dividing $\mathbf{x}_i$ by the true error standard deviation would yield subsamples with high $D-$optimality. In these cases, the simulation results show that using a good estimator in place of the true error standard deviations also gives good results in terms of $D-$optimality. The subsamples provided by ANNSA had $I-$optimality criteria that were less than to those provided by IBOSS and ODB, even when there was constant variance (g_4). This is expected, since ANNSA was run to minimize the $I-$optimality criterion. These results suggest that IBOSS does not always provide subsamples with the lowest $I-$optimality. For all cases, either ANNSA or WIBOSS had the lowest $I-$optimality. When the variance function was g_2 or g_3, WIBOSS had the lowest $I-$optimality; otherwise, ANNSA had the lowest $I-$optimality. The $D-$optimality for the ANNSA subsample is greater than or equal to the $D-$optimality for IBOSS in all cases. This result suggests that subsamples with low integrated prediction variance can be found without sacrificing $D-$optimality that one would get from the (computationally faster) IBOSS method. When the variance function was g_4, the variance was constant. In these cases, it was expected that IBOSS and WIBOSS would have similar $D-$optimality criteria, since the variance estimates used for WIBOSS should be very similar across all data points. Indeed, IBOSS and WIBOSS had similar D-optimality criteria. When the variance was constant, the ODB method had the highest $D-$optimality value, followed by ANNSA. These results show that, even when the variance is constant, WIBOSS performs well in terms of $D-$optimality.

The time cost for $n = 20, p = 5$, and variance function g_1 was measured. It took 0.17 seconds for IBOSS algorithm and 10.4 seconds for the ODB algorithm to find subsamples. The average time for WIBOSS and ANNSA algorithms were 11.40 and 51.24 seconds, respectively. Weighted IBOSS and ANNSA use more time because they need to build the K-D tree and estimate the variance for all N points in the full dataset.

5. Example

In this section, an example on airline data is used to demonstrate the efficacy of the proposed subsampling methods in a case where the variance is not constant. The full dataset is available in the Harvard Dataverse (Harvard 2008) at `https://doi.org/10.7910/DVN/HG7NV7`. Each row of the dataset corresponds to a flight in the US. In this example, we consider flights in the year 2007. After flights with missing values were removed, this led to a full dataset size of $N = 7{,}275{,}288$ flights. The response of interest was taken to be the arrival delay (in minutes), which is negative if the flight arrived early, and positive if the flight was late. It was of interest to see how a flight's elapsed time (in minutes), departure delay (in minutes), and travel distance (in miles) impacted the flight's arrival delay. The elapsed time, departure delay, travel distance, and arrival delays were each mean-centered and scaled by their standard deviations prior to analysis.

To illustrate the heteroscedasticity of the errors in this dataset, a random subsample of size 100,000 was used to fit Model (3). The residuals were then plotted against all $p = 3$ covariates, and these plots are shown in Figure 2. Figure 2 shows that as the elapsed time, departure delay, and distance increase, the variance of the residuals tends to decrease. This shows evidence that the variance of the errors depends on the values of the covariates.

The dataset was randomly divided into 70% training data and 30% testing data. The variances were estimated using training data via Algorithm 1 with a bandwidth of $h = 1$. The weighted IBOSS algorithm and ANNSA were used these variance estimates. The IBOSS, weighted IBOSS, and ANNSA algorithms were used to find subsamples with sizes of $n =$ 400,450,500. In each case, $n_1 = 200$ subsamples were used to estimate the variance, and the remaining $n_2 = n - n_1$ subsamples were found using either IBOSS, weighted IBOSS, or ANNSA. The estimated $I-$optimality criteria (using the estimated variances) the IBOSS and weighted IBOSS subsamples were compared, and the subsample with the lower estimated $I-$optimality was used as the initial subsample for ANNSA. For each of the obtained subsamples, a weighted least squares regression model was fitted and used to predict the responses for the testing set. For IBOSS and weighted IBOSS, the MSPE was found on the testing dataset. For each value of n in Table 3, the ANNSA algorithm was run 50 times to produce 50 different subsamples, each of size n. For each of these 50 subsamples, a weighted least squares model was fit to the training data, and the MSPE was calculated on the test data. These values

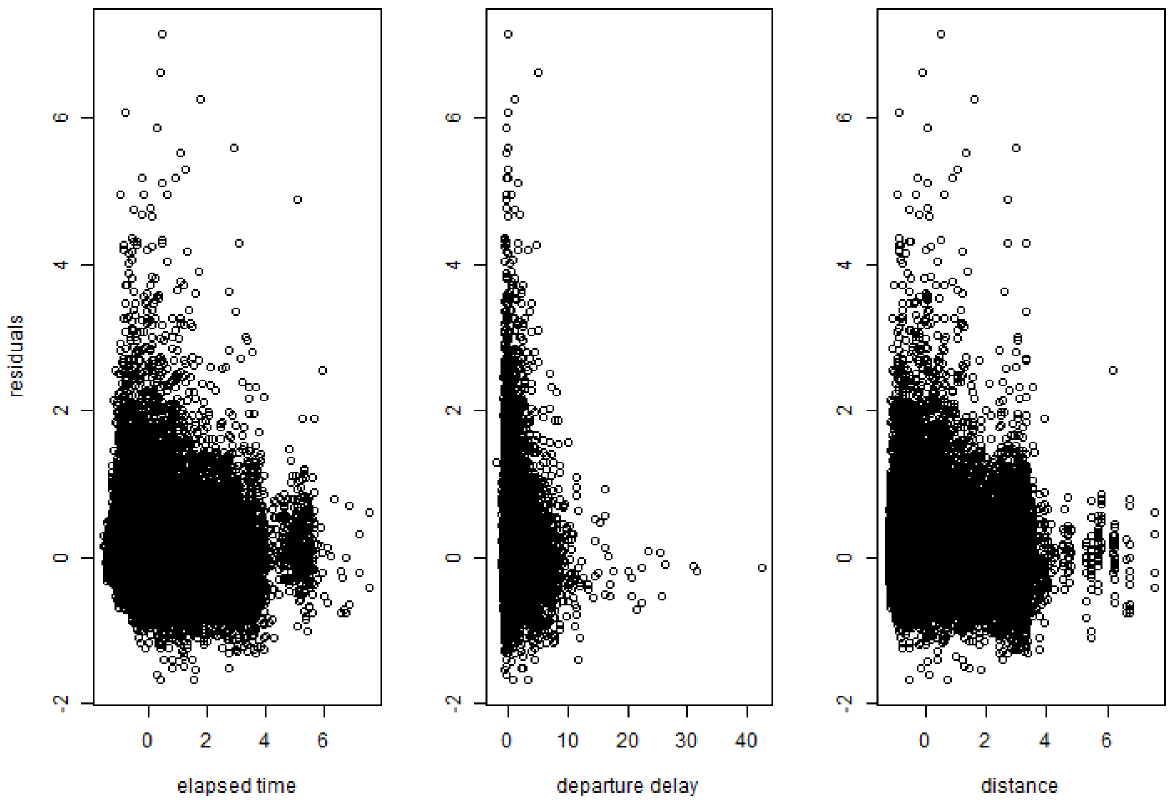

Figure 2: Residuals versus Predictors in a Random Subsample of size 100,000 from the Airline Dataset. All covariates are standardized (mean-centered and scaled by standard deviation).

were averaged to produce the values in Table 3 under the ANNSA column. The MSPEs are summarized in Table 3. The weights used in the WLS model were the estimated variances obtained from the training set. From Table 3, it can be seen that in this example, ANNSA had the lowest MSPEs in all cases, and this was followed by IBOSS.

Table 3: Mean squared prediction errors (using Weighted Least Squares).

			Subsampling Method		
n_1	n_2	n	IBOSS	WIBOSS	ANNSA
200	200	400	0.1842	0.1913	0.1701
200	250	450	0.1877	0.1949	0.1596
200	300	500	0.1996	0.2043	0.1568

It was also of interest to compare the regression coefficients obtained from each subsampling method with the regression coefficients obtained by using the entire training dataset. Table 4 below shows the coefficients obtained for each model. The first row shows the coefficients for the full model ($N = 7{,}275{,}288$), and the remaining rows shows the coefficients obtained for subsamples of size $n = 400$, with $n_1 = 200$ and $n_2 = 200$.

Since ANNSA returns a stochastic subsample, the coefficients shown for ANNSA are averages over 50 different subsamples.

Table 4: Mean coefficients when $n_1 = 200$, $n_2 = 200$.

variables	IBOSS	WIBOSS	ANNSA	full model
Intercept	0.228	0.243	0.069	0.001
Scheduled Elapsed Time	0.003	0.019	-0.121	-0.055
Departure Delay	0.914	0.914	0.892	0.935
Distance	-0.040	-0.059	0.263	0.035

Table 4 shows that all three methods (IBOSS, WIBOSS, ANNSA) had similar estimates for the effect of the departure delay, which appears to be the most important coefficient for predicting the arrival delay. The ANNSA coefficients for elapsed time and distance had the same signs as those found when using the full dataset, but this was not the case for IBOSS. The effect estimates for IBOSS and WIBOSS were very similar in this case. Among the three algorithms, ANNSA's fitted coefficients are closest to those obtained in the full model.

6. Conclusion and Future Work

This article introduces the Weighted IBOSS (WIBOSS) and ANNSA subsampling algorithms, as well as a variance estimation algorithm based on the NWE. While the established IBOSS algorithm from Wang et al. (2019) provides a framework for subsampling from data under the assumption of homoscedasticity, it does not address scenarios where the variance of the errors depends on the covariates. The proposed methods offer reliable procedures for subsampling from large datasets with heteroskedasticity by identifying subsamples with high D-optimality or low I-optimality. Based on the simulation results, ANNSA consistently delivered subsamples with the highest D-optimality and lowest I-optimality among three methods. The subsamples found by the weighted IBOSS algorithm generally had higher D-optimality than those found by IBOSS. When the variance was constant, weighted IBOSS and IBOSS had similar D-optimality. For the real world example as shown in Section 5, which exhibits heteroskedasticity, the ANNSA subsampling method was shown to produce weighted least squares estimates that resulted in lower mean squared prediction error than those obtained from IBOSS and weighted IBOSS.

While the article validates the proposed algorithms through simulations and real data analysis, more work that can be done on this topic. One key area is the theoretical properties of the ANNSA algorithm. Future research could focus on finding the theoretical limits of both I- and D-optimality and investigating the impact of errors from variance estimation on ANNSA's performance. Secondly, there is room for future work with respect to the NWE variance estimator such as a formal derivation of the bias. As a major component of the proposed framework, the variance estimation algorithm could also be further optimized to speed up the computing process. For example, weighted IBOSS requires an evaluation of the estimated variance of each data point in the full dataset. Future work could focus on developing a subsampling method leveraging a subset of data. Additionally, the proposed framework was built to subsample from datasets with continuous covariates. Modifying the current algorithms to handle categorical covariates would significantly expand the versatility of the proposed methods.

Acknowledgments

The authors confirm that the data supporting the findings of this study are available within the article and its Supplementary Material.

This research was not supported by any grants or funding.

No potential conflict of interest was reported by the authors.

References

Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Commun.*, 5:4308, DOI: 10.1038/ncomms5308.

Becerra, M. and Goos, P. (2021). Bayesian I-optimal designs for choice experiments with mixtures. *Chemometrics and Intelligent Laboratory Systems*, 217:104395, DOI: 10.1016/j.chemolab.2021.104395.

Bertsimas, D. and Tsitsiklis, J. (1993). Simulated annealing. *Statistical Science*, 8(1):10–15, DOI: 10.1214/ss/1177011077.

Cheng, Q., Wang, H., and Yang, M. (2020). Information-based optimal subdata selection for big data logistic regression. *Journal of Statistical Planning and Inference*, 209:112–122. *Statistical Science*, 8(1):10–15, DOI: 10.1016/j.jspi.2020.03.004.

Cia-Mina, A., Lopez-Fidalgo, J., and Wong, W. K. (2025). Optimal subdata selection for prediction based on the distribution of the covariates. *IEEE Transactions on Big Data*, https://ieeexplore.ieee.org/document/10930599.

Deldossi, L. and Tommasi, C. (2021). Optimal design subsampling from big datasets. *Journal of Quality Technology*, 54(1):93–101, DOI: 10.1080/00224065.2021.1889418.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. (2006). Sampling algorithms for l_2 regression and applications. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1127–1136, https://dl.acm.org/doi/10.5555/1109557.1109682.

Fan, J. and Yao, Q. (1998). Efficient estimation of conditional variance functions in stochastic regression. *Biometrika*, 85(3):645–660, https://www.jstor.org/stable/2337393.

Fonollosa, J., Sheik, S., Huerta, R., and Marco, S. (2015). Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical*, 215:618–629, DOI: 10.1016/j.snb.2015.03.028.

Goos, P., Jones, B., and Syafitri, U. (2016). I-optimal design of mixture experiments. *Journal of the American Statistical Association*, 111(514):899–911, DOI: 10.1080/01621459.2015.1136632.

Greblicki, W. and Krzyzak, A. (1980). Asymptotic properties of kernel estimates of a regression function. *Journal of Statistical Planning and Inference*, 4(1):81–90, DOI: 10.1016/0378-3758(80)90036-1.

Harman, R. and Filova, L. (2025). *OptimalDesign: A Toolbox for Computing Efficient Designs of Experiments*, https://CRAN.R-project.org/package=OptimalDesign. R package version 1.0.3.

Harvard (2008). Data Expo 2009: Airline on time data, DOI: 10.7910/DVN/HG7NV7.

Kiefer, J. (1974). General equivalence theory for optimum designs (approximate theory). *The Annals of Statistics*, 2(5):849–879, DOI: 10.1214/aos/1176342810.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680, DOI: 10.1126/science.220.4598.67.

Li, Y. and Deng, X. (2021). An efficient algorithm for elastic I-optimal design of generalized linear models. *Canadian Journal of Statistics*, 49(2):438–470, DOI: 10.1002/cjs.11571.

Liu, Y., Stufken, J., and Yang, M. (2023). Information-based optimal subdata selection for clusterwise linear regression. *arXiv:2309.00720*, https://arxiv.org/abs/2309.00720.

Mangla, P. (2024). Implementing approximate nearest neighbor search with KD-trees. https://pyimagesearch.com/2024/12/23/implementing-approximate-nearest-neighbor-search-with-kd-trees/.

McKay, M., Beckman, R., and Conover, W. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, https://www.jstor.org/stable/1268522.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, DOI: 10.1137/1109020.

Sambo, F., Borrotti, M., and Mylona, K. (2014). A coordinate-exchange two-phase local search algorithm for the D-and I-optimal designs of split-plot experiments. *Computational Statistics & Data Analysis*, 71:1193–1207, DOI: 10.5555/2749482.2749877.

Singh, R. and Stufken, J. (2023). Subdata selection with a large number of variables. *The New England Journal of Statistics in Data Science*, 1(3), DOI: 10.51387/23-NEJSDS36.

Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, https://www.jstor.org/stable/1269769.

Van Lieshout, M. (2020). Infill asymptotics and bandwidth selection for kernel estimators of spatial intensity functions. *Methodology and Computing in Applied Probability*, 22(3):995–1008, DOI: 10.1007/s11009-019-09749-x.

Virtanen, P., Gommers, R., Oliphant, T.E. and et al. (2020). **SciPy 1.0**: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, DOI: 10.1038/s41592-019-0686-.

Wang, H., Yang, M., and Stufken, J. (2019). Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association*, 114(525):393–405, DOI: 10.1080/01621459.2017.1408468.

Wang, H., Zhu, R., and Ma, P. (2018). Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association*, 113(522):829–844, DOI: 10.1080/01621459.2017.1292914.

Yao, Y. and Wang, H. (2021). A review on optimal subsampling methods for massive datasets. *Journal of Data Science*, 19(1):151–172, ISSN: 1680-743X, DOI: 10.6339/21-JDS999.

Affiliation:

Jiayi Zheng
Department of Statistics
George Mason University
4511 Patriot Cir, Suite 1705 Fairfax, Virginia 22030
E-mail: jzheng24@gmu.edu

Dongqi Fu
Department of Statistics
George Mason University
4511 Patriot Cir, Suite 1705 Fairfax, Virginia 22030
E-mail: dfu5@gmu.edu

Ziqiao Xu
Department of Statistics
George Mason University
4511 Patriot Cir, Suite 1705 Fairfax, Virginia 22030
E-mail: zxu30@gmu.edu

Nicholas Rios
Department of Statistics
George Mason University
4511 Patriot Cir, Suite 1705 Fairfax, Virginia 22030
E-mail: nrios4@gmu.edu

Journal of Data Science, Statistics, and Visualisation https://jdssv.org/
published by the International Association for Statistical Computing
http://iasc-isi.org/

ISSN 2773-0689
February 2026, Volume VI, Issue I *Submitted:* 2025-12-11
doi:10.52933/jdssv.v6i1.167 *Accepted:* 2026-01-29