Statistica Sinica Preprint No: SS-2021-0257	
Title	Subsampling and Jackknifing: A Practically Convenient Solution for Large Data Analysis With Limited Computational Resources
Manuscript ID	SS-2021-0257
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202021.0257
Complete List of Authors	Shuyuan Wu, Xuening Zhu and Hansheng Wang
Corresponding Author	Xuening Zhu
E-mail	xueningzhu@fudan.edu.cn

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Statistica Sinica

Subsampling and Jackknifing: A Practically Convenient Solution for Large Data Analysis with Limited Computational Resources

Shuyuan Wu[1], Xuening Zhu[2], and Hansheng Wang[1]

[1]*Peking University and* [2]*Fudan University*

Abstract: Modern statistical analysis often involves large data sets, for which conventional estimation methods are not suitable, owing to limited computational resources. To solve this problem, we propose a novel subsampling-based method with jackknifing. The key idea is to treat the whole sample as if it were the population. Then, we obtain multiple subsamples with greatly reduced sizes using simple random sampling with replacement. We do not recommend sampling methods without replacement, because this would incur a significant data processing cost when the processing occurs on a hard drive. However, such a cost does not exist if the data are processed in memory. Because subsampled data have relatively small sizes, they can be comfortably read into computer memory and processed. Based on subsampled data sets, jackknife-debiased estimators can be obtained for the target parameter. The resulting estimators are statistically consistent, with an extremely small bias. Finally, the jackknife-debiased estimators from different subsamples are averaged to form the final estimator. We show theoretically that the final estimator is consistent and asymptotically

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

normal.Furthermore, its asymptotic statistical efficiency can be as good as that of the whole sample estimator under very mild conditions. The proposed method is easily implemented on most computer systems, and thus is widely applicable.

Key words and phrases: GPU, Jackknife, Large Dataset, Subsampling.

1. INTRODUCTION

Modern statistical analysis often involves large data sets. However, many researchers operate under limited computational resources, and do not have access to powerful computation systems, such as a distributed system like Hadoop or Spark. Thus, how to practically analyze large data sets with limited computational resources has become a problem of great importance.

To solve this problem, various subsampling methods have been proposed (Mahoney, 2011; Drineas et al., 2011; Ma et al., 2015; Wang et al., 2018; Wang, 2019; Yu et al., 2020; Ma et al., 2020). The key idea of most existing methods is to design a novel sampling strategy so that excellent statistical efficiency can be achieved with small sample sizes. For example, Ma et al. (2015) developed a novel method of selecting an optimal subsample based on leverage scores. Wang et al. (2018) studied a similar problem and proposed the A-optimality criterion. Yu et al. (2020) developed an optimal

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Poisson subsampling approach. Ma et al. (2020) derived the asymptotic distribution of the sampling estimator based on a linear regression. Despite their usefulness, these methods suffer from two limitations. First, specific sampling strategies must be designed carefully for different analysis purposes. Second, they are computationally expensive. In most cases, the sampling cost is at least $O(N)$, where N represents the whole sample size.

To overcome these challenges, we have developed a novel method with the following unique features. First, our method is simple enough to be easily implemented on most practical computer systems. We argue that this simplicity is particularly relevant and important, because it implies wider applicability. Second, due to jackknifing, our estimators lead to a significant bias reduction compared with other methods. As a result, the same asymptotic efficiency can be achieved with a much reduced subsample size, as long as the number of subsamples is sufficiently large. Moreover, our method supports fully automatic and unified inferences. Most real applications require valid statistical inferences (e.g., confidence interval). However, the analytical formula for the asymptotic distribution of the estimator may be too complicated to be derived analytically. Our proposal is automatic in the sense that the standard errors of various statistics can be computed automatically without referring to an analytical formula of their asymptotic

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

distributions. In addition, our proposal is unified in the sense that it can be readily applied to many different statistics.

Specifically, we have develop a subsampling method with jackknifing. To implement our method, multiple subsamples are obtained by simple random sampling with replacement. For each subsampled data set, a jackknife-debiased estimator is computed for the parameter of interest, after which the estimators are averaged, yielding the final estimator. We show theoretically that the resulting estimator is consistent and asymptotically normal. In addition, its statistical efficiency can be asymptotically as good as that of the whole sample estimator under very mild conditions. This useful property remains valid, even if the subsample size is very small. The desirable property is mainly attributed to jackknifing. As a byproduct, a jackknife estimator for the standard error of the proposed estimator can be obtained, enabling automatic statistical inference. For practical implementation, we develop an algorithm based on a graphical processing unit (GPU), which empirical show is extremely computationally efficient. Extensive numerical studies are presented to demonstrate the finite-sample performance.

Despite its usefulness, the proposed method suffers from several limitations. The main limitation is that it is computationally less efficient than the one-pass full-sample mean estimators computed by distributed

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

approaches (Suresh et al., 2017). However, the propose method is a practically more convenient alternative under the following two important situations. The first is when the whole sample size N is extremely large. In this case, a significant cost is incurred when processing the whole sample (e.g., computing the one-pass full-sample mean). This is particularly true if no distributed computation system is available. However, for most practical data analysis, the demand for estimation precision is limited, and it is more important to control processing costs, leading to a trade-off between the two. Accordingly, we do not expect our method to be implemented with a very large subsample size n and a very large number of subsamples K. Instead, it should be implemented with reasonably large n and K, as long as the desired statistical precision can be achieved.

The second situation is when automatic statistical inferences are required. In this case, if the one-pass full-sample mean is used, then an analytical formula for the asymptotic distribution of the estimator has to be derived manually. It is then preferable to have an automatic and unified solution for statistical inference.

The rest of the paper is organized as follows. Section 2 presents the proposed estimators and their asymptotic properties. Numerical studies are presented in Section 3, including the GPU-based algorithm, simulation

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

experiments, and a real data set analysis. Finally, Section 4 concludes the paper. All technical details are relegated to the Supplementary Material.

2. THE METHODOLOGY

2.1 Model and Notation

Let X_i be an independent random variable observed from the ith subject, where $1 \leq i \leq N$ and N is the whole sample size. Let $\mathbb{S} = \{1, \ldots, N\}$ be the index set of the whole sample, and let μ be one particular moment about X_i. For simplicity, we assume $\mu = E(X_i)$ is a scalar. The theory presented here can be extended easily to a more general situation with multivariate moments and M estimators. Let $\theta = g(\mu)$ be the parameter of interest, where $g(\cdot)$ is a known nonlinear function. We assume that $g(\cdot)$ is sufficiently smooth. To estimate θ, one can use a sample moment estimator $\widehat{\theta} = g(\widehat{\mu})$, where $\widehat{\mu} = N^{-1} \sum_{i \in \mathbb{S}} X_i$.

For convenience, we refer to $\widehat{\theta}$ as the whole sample estimator to emphasize the fact that this is an estimator computed based on the whole sample. The merit of $\widehat{\theta}$ is that it offers excellent statistical efficiency. However, it may be difficult to compute if the whole sample size N is too large. This is particularly true if researchers have limited computational resources. Accordingly, we must consider other estimation methods that

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

are more computationally feasible. Here, we study one type of subsampling method (Mahoney, 2011; Drineas et al., 2011; Ma et al., 2015; Wang et al., 2018; Wang, 2019; Yu et al., 2020; Ma et al., 2020) as an excellent and practical solution.

Let n be the subsample size, which is typically much smaller than N. Let K be the number of subsamples. Write $\mathcal{S}_k = \{i_1^{(k)}, \ldots, i_n^{(k)}\} \subset \mathbb{S}$ as the kth subsample set, where $i_m^{(k)}$ (for any $1 \leq m \leq n$, $1 \leq k \leq K$) are generated independently from $\mathbb{S}$ by the method of simple random sampling with replacement. In other words, conditional on $\mathbb{S}$, $i_m^{(k)}$ are independently and identically distributed (i.i.d.) with probability $\mathrm{P}(i_m^{(k)} = j) = N^{-1}$, for any $j \in \mathbb{S}$. Accordingly, a moment estimator based on $\mathcal{S}_k$ can be computed as $\widehat{\theta}^{(k)} = g\left(\widehat{\mu}^{(k)}\right)$, where $\widehat{\mu}^{(k)} = n^{-1} \sum_{i \in \mathcal{S}_k} X_i$. One can then combine these subsample estimators to form a more accurate one as $\widehat{\theta}_{SOS} = K^{-1} \sum_{k=1}^{K} \widehat{\theta}^{(k)}$. This is referred to as a subsample one-shot (SOS) estimator. It is similar to the so-called one-shot estimator developed for distributed systems (Mcdonald et al., 2009; Zinkevich et al., 2011; Zhang et al., 2013). However, the key difference is that the subsamples used by the standard one-shot estimator should not overlap. In contrast, the subsamples used by our proposed subsampling method are allowed to partially overlap.

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

2.2 Variance and Bias Analysis of the SOS Estimator

To motivate our method, we offer an informal analysis of the bias and variance of the SOS estimator $\widehat{\theta}_{SOS}$. Formal theoretical results are provided in Section 2.4. Specifically, using Taylor's expansion, we can approximate $\widehat{\theta}^{(k)}$ as

$$\widehat{\theta}^{(k)} \approx \theta + \dot{g}(\mu)\left(\widehat{\mu}^{(k)} - \mu\right) + \frac{1}{2}\ddot{g}(\mu)\left(\widehat{\mu}^{(k)} - \mu\right)^2,$$

where $\dot{g}(\mu)$ and $\ddot{g}(\mu)$ are the first- and second-order derivatives of $g(\mu)$, respectively, with respect to μ. Accordingly, we have

$$\widehat{\theta}_{SOS} = \frac{1}{K}\sum_{k=1}^{K}\widehat{\theta}^{(k)} \approx \theta + \frac{\dot{g}(\mu)}{K}\sum_{k=1}^{K}\left(\widehat{\mu}^{(k)} - \mu\right) + \frac{\ddot{g}(\mu)}{2K}\sum_{k=1}^{K}\left(\widehat{\mu}^{(k)} - \mu\right)^2. \quad (2.1)$$

By equation (2.1), we know that $\mathrm{var}(\widehat{\theta}_{SOS})$ can be approximated by the variance of $\dot{g}(\mu)K^{-1}\sum_{k=1}^{K}\left(\widehat{\mu}^{(k)} - \mu\right)$. Let $\widehat{\sigma}^2 = N^{-1}\sum_{i\in\mathbb{S}}(X_i - \widehat{\mu})^2$. With a slight abuse of notation, we use $\mathbb{S}$ to represent the information contained in the whole sample, that is, the σ-field generated by $\{X_1, \ldots, X_N\}$. Recall that, conditional on $\mathbb{S}$, X_i are i.i.d. for any $i \in \mathcal{S}_k$ and $1 \le k \le K$. We then have $E(\widehat{\mu}^{(k)}|\mathbb{S}) = n^{-1}E\left(\sum_{i\in\mathcal{S}_k} X_i|\mathbb{S}\right) = \widehat{\mu}$ and $\mathrm{var}\left(\widehat{\mu}^{(k)}|\mathbb{S}\right) = n^{-2}\,\mathrm{var}\left(\sum_{i\in\mathcal{S}_k} X_i|\mathbb{S}\right) = n^{-1}\widehat{\sigma}^2$. Assume that the second moment of X_1

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

is finite with $\sigma^2 = \text{var}(X_1)$. We then have

$$\begin{aligned} E\left\{\text{var}\left(\widehat{\theta}_{SOS}|\mathbb{S}\right)\right\} &\approx \frac{\dot{g}(\mu)^2}{K} E\left\{\text{var}\left(\widehat{\mu}^{(k)}|\mathbb{S}\right)\right\} \approx \frac{\dot{g}(\mu)^2}{nK}\sigma^2, \\ \text{var}\left\{E\left(\widehat{\theta}_{SOS}|\mathbb{S}\right)\right\} &\approx \dot{g}(\mu)^2 \,\text{var}\left(\widehat{\mu} - \mu\right) = \frac{\dot{g}(\mu)^2}{N}\sigma^2. \end{aligned} \tag{2.2}$$

From equation (2.2), we find that $\text{var}(\widehat{\theta}_{SOS})$ can be approximated by $\tau_1\{1/N + 1/(nK)\}$, with $\tau_1 = \dot{g}(\mu)^2\sigma^2$. Under the condition $nK \gg N$, we then determine that the variance of the subsample estimator can be further approximated by τ_1/N, which is the asymptotic variance of the whole sample estimator $\widehat{\theta}$.

Next, we study the bias of $\widehat{\theta}_{SOS}$. We define the bias of T_n as $\text{Bias}(T_n) = E(T_n) - \theta$, for any estimator T_n of θ. Then, by equation (2.1), we have

$$\text{Bias}\left(\widehat{\theta}_{SOS}\right) = E\left(\frac{1}{K}\sum_{k=1}^{K}\widehat{\theta}^{(k)}\right) - \theta = E\left(\widehat{\theta}^{(k)}\right) - \theta \approx \frac{\ddot{g}(\mu)}{2n}\sigma^2 + \frac{\ddot{g}(\mu)}{2N}\sigma^2. \tag{2.3}$$

The leading term of $\text{Bias}(\widehat{\theta}_{SOS})$ is given by τ_2/n, with $\tau_2 = \ddot{g}(\mu)\sigma^2/2$. Unfortunately, it does not improve as K increases. This indicates that the bias of $\widehat{\theta}_{SOS}$ is of order $O(n^{-1})$, and is a smaller order term compared with $\widehat{\theta} - \theta = O_p(1/\sqrt{N})$, as long as $n \gg \sqrt{N}$. This condition seems to be quite reasonable for a distributed system (Huang and Huo, 2015; Jordan et al.,

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

2019), in which, K is the number of distributed computers. As a result, K is typically much smaller than n, where n is the subsample size allocated to each distributed computer. However, this condition could be problematic for a subsampling method. In this case, K is the total number of subsamples, and could be very large. In contrast, for computational convenience, the subsample size n could be much smaller than $\sqrt{N}$. This makes the bias introduced in equation (2.3) possibly non-negligible. To fix this problem, we are motivated to search for an improved estimator for θ with a greatly reduced bias. In this regard, the jackknife method is well known to reduce the bias of estimators (Quenouille, 1949; Efron and Stein, 1981; Cameron and Trivedi, 2005). However, the performance of the jackknife method in the subsampling scenario is not clear. This leads to the novel jackknife estimators presented in the next subsection.

2.3 Jackknife Estimators

This subsection has two objectives. The first is to develop a jackknife debiased subsample (JDS) estimator for θ. The second is to propose a jackknife standard error (JSE) estimator for the JDS estimator.

First, we develop the JDS estimator to reduce the estimation bias. To this end, we define a jackknife estimator $\widehat{\theta}_{-j}^{(k)}$ for the kth subsample as

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

follows:

$$\widehat{\theta}_{-j}^{(k)} = g\left(\widehat{\mu}_{-j}^{(k)}\right), \text{ where } \widehat{\mu}_{-j}^{(k)} = \frac{1}{n-1}\sum_{i\in\mathcal{S}_k}^{i\neq j} X_i.$$

By similar analysis to that for equation (2.1), we know that $\text{Bias}\big(\widehat{\theta}_{-j}^{(k)}\big)$ is approximately equal to $\tau_2/(n-1)$. Then, $n^{-1}\sum_{j\in\mathcal{S}_k}\text{Bias}\big(\widehat{\theta}_{-j}^{(k)}\big) \approx \tau_2/(n-1)$, and $E\big(n^{-1}\sum_{j\in\mathcal{S}_k}\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)}\big) \approx \tau_2/\{n(n-1)\}$. This inspires an estimator for the bias given by $\widehat{\text{Bias}}^{(k)} = (n-1)n^{-1}\sum_{j\in\mathcal{S}_k}\widehat{\theta}_{-j}^{(k)} - (n-1)\widehat{\theta}^{(k)}$. Accordingly, we propose a bias-corrected estimator for the kth subsample as $\widehat{\theta}_{JDS}^{(k)} = \widehat{\theta}^{(k)} - \widehat{\text{Bias}}^{(k)}$. Thereafter, $\widehat{\theta}_{JDS}^{(k)}$ can be averaged across different k. As a result, we obtain the final JDS estimator $\widehat{\theta}_{JDS} = K^{-1}\sum_{k=1}^{K}\widehat{\theta}_{JDS}^{(k)}$. Next, we rigorously verify that $\text{Bias}(\widehat{\theta}_{JDS})$ is much smaller than the bias of $\widehat{\theta}_{SOS}$. Specifically, $\text{Bias}(\widehat{\theta}_{JDS}) = O(1/n^2) + O(1/N)$ and $\text{Bias}(\widehat{\theta}_{SOS}) \approx \tau_2/n$; see equation (2.3). Furthermore, we can prove theoretically that the asymptotic variance of $\widehat{\theta}_{JDS}$ remains the same as that of the whole sample estimator. As a result, assuming that K is sufficiently large, excellent statistical efficiency can be achieved by $\widehat{\theta}_{JDS}$ with a very small subsample size n.

In addition to bias correction, the jackknife method also serves as an excellent estimator for the standard error, that is, the standard deviation of the JDS estimator $\widehat{\theta}_{JDS}$. The basic idea is as follows. Recall that by

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

equation (2.2), we know that

$$\mathrm{var}(\widehat{\theta}_{SOS}) \approx \dot{g}(\mu)^2\sigma^2\left(\frac{1}{nK}+\frac{1}{N}\right). \tag{2.4}$$

Because N, n, and K are all known to the user, the key objective here is to estimate the unknown parameter $\tau_1 = \dot{g}(\mu)^2\sigma^2$. Moreover, by the definition of the jackknife estimator and Taylor's expansion, we have

$$\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)} \approx \dot{g}(\mu)\left(\widehat{\mu}_{-j}^{(k)} - \widehat{\mu}^{(k)}\right) = \frac{\dot{g}(\mu)}{n-1}\left(\widehat{\mu}^{(k)} - X_j\right)$$

for any $j \in \mathcal{S}_k$ and $1 \le k \le K$. We know immediately that

$$E\left\{\left(\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)}\right)^2\right\} = E\left[E\left\{\left(\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)}\right)^2 \middle| \mathbb{S}\right\}\right] \approx \frac{\dot{g}(\mu)^2\sigma^2}{n(n-1)} = \frac{\tau_1}{n(n-1)},$$

which is closely related to the unknown parameter τ_1 in equation (2.4). Note that the sample mean of $\left(\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)}\right)^2$ across different j and k is a reasonable approximation of $E\left\{(\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)})^2\right\}$. This inspires the following JSE estimator $\widehat{\mathrm{SE}}$:

$$\widehat{\mathrm{SE}}^2 = \left(\frac{1}{K}+\frac{n}{N}\right)\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{S}_k}\left(\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)}\right)^2.$$

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

We prove theoretically that $\widehat{\mathrm{SE}}^2$ is a consistent estimator of $\mathrm{var}(\widehat{\theta}_{JDS})$. In addition, $\mathrm{var}(\widehat{\theta}_{JDS})/\mathrm{var}(\widehat{\theta}_{SOS}) = 1 + o(1)$. Consequently, $\widehat{\mathrm{SE}}^2$ is also a consistent estimator of $\mathrm{var}(\widehat{\theta}_{SOS})$.

2.4 Theoretical Properties

In this subsection, we study the theoretical properties of the three estimators (i.e., the SOS, JDS, and JSE estimators). To this end, we need the following standard technical conditions.

(C1) (Sub-Gaussian Distribution) Assume X_i follow a sub-Gaussian distribution, that is, there exist positive constants C and ν such that $P(|X_i| > t) \leq C\exp\{-\nu t^2\}$, for every $t > 0$.

(C2) (Smoothness condition) Define $g^{(k)}(\cdot)$ as the kth order derivative function of $g(\cdot)$, and assume $g^{(k)}(\cdot)$ is a continuous function, for $k \leq 8$.

(C3) (Subsampling condition) As $N \to \infty$, the subsample size $n \to \infty$. In addition, assume that $n < N, N = o(n^4)$, and $\log K = o(\sqrt{n})$.

Condition (C1) is a classical and flexible assumption on covariates (Jordan et al., 2019; Zhu et al., 2021). Condition (C2) requires the g-function to be sufficiently smooth so that Taylor's expansion can be obtained around μ. We require a slightly stronger condition, because we derive the asymptotic bias

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

in a more explicit form. The condition can be relaxed to requiring $g(\cdot)$ to be a fourth continuously differentiable function to guarantee the asymptotic normality (Wu, 1986; Lehmann and Casella, 2006). Lastly, Condition (C3) states the relationships between n, N, and K. It requires that the subsample size should be large enough to facilitate an asymptotic analysis of higher order terms. In addition, we require $\log K = o(\sqrt{n})$ to guarantee a uniform convergence for all subsamples, which is easily satisfied in practice.

We next consider how to understand the asymptotic behavior of various subsample estimators without finite moment constraints. Inspired by the asymptotic theory of Shao (2003), we adopt a Taylor expansion approach. Consider $\widehat{\theta}_{SOS}$ as an example. By Taylor's expansion, we have $\widehat{\theta}_{SOS} = K^{-1}\sum_{k=1}^{K}\widehat{\theta}^{(k)} = K^{-1}\sum_{k=1}^{K} g(\widehat{\mu}^{(k)}) = \theta + \widehat{\Delta}_{SOS}^{(1)} + \widehat{\Delta}_{SOS}^{(2)} + \mathcal{O}$, where $\widehat{\Delta}_{SOS}^{(1)} = \dot{g}(\mu)K^{-1}\sum_{k=1}^{K}(\widehat{\mu}^{(k)} - \mu)$, $\widehat{\Delta}_{SOS}^{(2)} = K^{-1}\sum_{k=1}^{K}\big\{\ddot{g}(\mu)\ (\widehat{\mu}^{(k)} - \mu)^2/2 + g^{(3)}(\mu)(\widehat{\mu}^{(k)} - \mu)^3/6\big\}$, and $\mathcal{O}$ represents for some higher order terms. As discussed informally in Section 2.2, this suggests that the asymptotic behavior of $\widehat{\theta}_{SOS} - \theta$ can be fully determined by $\widehat{\Delta}_{SOS}^{(1)}$ and $\widehat{\Delta}_{SOS}^{(2)}$. Here, $\widehat{\Delta}_{SOS}^{(1)}$ is unbiased and mainly contributes to the variance, whereas $\widehat{\Delta}_{SOS}^{(2)}$ has ignorable variance and mainly controls the bias. Accordingly, we can understand the asymptotic performance of the bias of $\widehat{\theta}_{SOS}$ by $E(\widehat{\Delta}_{SOS}^{(2)})$ and the variance of $\widehat{\theta}_{SOS}$'s variance by $\mathrm{var}(\widehat{\Delta}_{SOS}^{(1)})$. Specifically, we have the following theorem.

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Theorem 1. *Assume conditions (C1)–(C3) hold. Then we have* $\widehat{\theta}_{SOS} - \theta = \widehat{\Delta}^{(1)}_{SOS} + \widehat{\Delta}^{(2)}_{SOS} + \mathcal{O}$, *with* $E(\widehat{\Delta}^{(1)}_{SOS}) = 0$, $\mathrm{var}(\widehat{\Delta}^{(2)}_{SOS}) = o\{1/(nK) + 1/N\}$, $\mathcal{O} = o_p\big(1/n + \sqrt{1/(nK) + 1/N}\big)$, *and*

$$E(\widehat{\Delta}^{(2)}_{SOS}) = \tau_2 \left(\frac{1}{n} + \frac{1}{N}\right) + o\left(\frac{1}{n}\right) \tag{2.5}$$

$$\mathrm{var}(\widehat{\Delta}^{(1)}_{SOS}) = \tau_1 \left(\frac{1}{nK} + \frac{1}{N}\right) + o\left(\frac{1}{nK} + \frac{1}{N}\right). \tag{2.6}$$

From Theorem 1, we first find that the higher order terms $\mathcal{O}$ may be ignorable compared with $\widehat{\Delta}^{(1)}_{SOS}$ and $\widehat{\Delta}^{(2)}_{SOS}$. In addition, the asymptotic bias behavior of $\widehat{\theta}_{SOS}$ is decided by $\widehat{\Delta}^{(2)}_{SOS}$, whereas the asymptotic variance behavior of $\widehat{\theta}_{SOS}$ is determined by $\widehat{\Delta}^{(1)}_{SOS}$. Then, by equation (2.5), we know that the bias of $\widehat{\Delta}^{(2)}_{SOS}$ is affected by both N and n. The $1/n$ and $1/N$ terms represent the asymptotic bias due to the subsampling and overall sampling errors, respectively. The leading term of the variance for $\widehat{\Delta}^{(1)}_{SOS}$ also includes two quantities, namely, the $1/(nK)$ and $1/N$ terms. The first term is due to the subsampling error, and the second term is due to the overall sampling error. Recall that the asymptotic variance of the whole sample estimator $\widehat{\theta}$ is approximately equal to τ_1/N. Then, for the SOS estimator to achieve the same asymptotic efficiency as $\widehat{\theta}$, we must have $nK/N \to \infty$. Unfortunately, the subsampling error term of Bias$(\widehat{\Delta}^{(2)}_{SOS})$ is

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

$O(1/n)$, which does not reduce at all as $K \to \infty$. Consequently, we need to have $n \gg \sqrt{N}$ so that the asymptotic bias is of $o(1/\sqrt{N})$. Otherwise, the SOS estimator can never be asymptotically as efficient as the whole sample estimator $\widehat{\theta}$. Similarly to $\widehat{\theta}_{SOS}$, we can express $\widehat{\theta}_{JDS}$ using Taylor's expansion as $\widehat{\theta}_{JDS} = \widehat{\Delta}^{(1)}_{JDS} + \widehat{\Delta}^{(2)}_{JDS} + \mathcal{O}$, a detailed expression is given in Appendix B. Define $\tau_3 = g^{(3)}(\mu)\mu_3/6, \tau_4 = g^{(4)}(\mu)\sigma^4/8$, and $\mu_3 = E(X_i - \mu)^3$. We next analyze the properties of the JDS estimator in the following theorem.

Theorem 2. *Assume conditions (C1)–(C3) hold. Then,* $\widehat{\theta}_{JDS} - \theta = \widehat{\Delta}^{(1)}_{JDS} + \widehat{\Delta}^{(2)}_{JDS} + \mathcal{O}$, *with* $E(\widehat{\Delta}^{(1)}_{JDS}) = 0$, $\mathrm{var}(\widehat{\Delta}^{(2)}_{JDS}) = o\{1/(nK) + 1/N\}$, $\mathcal{O} = o_p\big(1/n^2 + 1/N + \sqrt{1/(nK) + 1/N}\big)$, *and*

$$E(\widehat{\Delta}^{(2)}_{JDS}) = \frac{\tau_2}{N} + \frac{\tau_3 + \tau_4}{n^2} + o\left(\frac{1}{N} + \frac{1}{n^2}\right) \tag{2.7}$$

$$\mathrm{var}(\widehat{\Delta}^{(1)}_{JDS}) = \tau_1\left(\frac{1}{nK} + \frac{1}{N}\right) + o\left(\frac{1}{nK} + \frac{1}{N}\right). \tag{2.8}$$

Comparing (2.5) and (2.7), we find that for the JDS estimator, the bias term due to the subsampling error is substantially reduced to $O(1/n^2)$. In contrast, that of the SOS estimator is much larger, and is of order $O(1/n)$. Comparing (2.6) and (2.8), we conclude that the leading terms of the variances of the two estimators are identical, and can be estimated consistently using the proposed JSE estimator $\widehat{\mathrm{SE}}$. Its asymptotic property is given as

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

follows.

Theorem 3. *Define* $\tau^2 = \tau_1 \{1/(nK) + 1/N\}$, *and further assume conditions (C1)–(C3) hold. The JSE estimator is then ratio consistent for* τ, *that is,* $\widehat{\mathrm{SE}}^2/\tau^2 \to_p 1$, *where "*$\to_p$*" stands for "convergence in probability."*

Lastly, for a valid asymptotic inference, we need to study the asymptotic distributions of the JDS estimator $\widehat{\theta}_{JDS}$ and the SOS estimator $\widehat{\theta}_{SOS}$. Consequently, we develop the following theorem to establish the asymptotic normality for both $\widehat{\theta}_{JDS}$ and $\widehat{\theta}_{SOS}$.

Theorem 4. *Assume conditions (C1)–(C3) hold. The JDS estimator* $\widehat{\theta}_{JDS}$ *is then asymptotically normal, with* $\left(\widehat{\theta}_{JDS} - \theta\right)/\tau \to_d N(0,1)$, *where "*$\to_d$*" represents "convergence in distribution." If one can impose the stronger condition that* $n/N^{1/2} \to \infty$, *then the SOS estimator* $\widehat{\theta}_{SOS}$ *is also asymptotically normal with* $\left(\widehat{\theta}_{SOS} - \theta\right)/\tau \to_d N(0,1)$.

From Theorem 4, we know that the SOS and JDS estimators are both asymptotically normal. However, the technical conditions required by the two estimators are different. The JDS estimator requires $n/N^{1/4} \to \infty$. This is a condition that can be very easily satisfied. However, for the SOS estimator, a much stronger condition (i.e., $n/N^{1/2} \to \infty$) is required (Huang and Huo, 2015; Jordan et al., 2019; Wang et al., 2020).

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

3. NUMERICAL ANALYSIS

3.1 Why Sampling with Replacement

We aim to develop a GPU-based algorithm for the proposed method when the data are stored on a hard drive. Thus, it is important to understand the sampling mechanism on the hard drive. In particular, we examine the computational efficiency of the different sampling mechanisms (i.e., simple random sampling with replacement and simple random sampling without replacement on a hard drive) in the following steps:

(1) First, we assume there are N data points (representing a massive data set) stored on a hard drive. They are displayed in the top left of Figure 6. There are two columns. The first is the sample ID (ID $= 1, 2, 3, 4, 5, \ldots, N$), and the second is the variable of interest $Y = (Y_1, \ldots, Y_N)$.

(2) Second, to conduct random sampling, we randomly generate an integer between 1 and N. This determines which data line should be sampled. Without loss of generality, assume that the sampled unit is i^*. Then, we read Y_{i^*} into memory. (Note that this sampling procedure is a simplified version. In practice, we cannot access a data line by its sample ID on a hard drive. Instead, we refer to it according

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

to its physical address on a hard drive. However, this is also not a straightforward operation). We then update the index set $\mathcal{S}_0$ from $\mathcal{S}_0 = \{\emptyset\}$ to $\mathcal{S}_0 = \{i^*\}$.

(3) Third, we conduct random sampling without replacement. To this end, we generate another integer i_2^* from 1 to N randomly and independently. It is possible that i_2^* has already been sampled in $\mathcal{S}_0$, which leads to duplicated sampling. To avoid this, i_2^* needs to be compared with every unit in $\mathcal{S}_0$ that has already been sampled. If we find $i_2^* \in \mathcal{S}_0$ already, then i_2^* needs to be re-generated. Otherwise, $\mathcal{S}_0$ can be updated to $\mathcal{S}_0 := \mathcal{S}_0 \cup \{i_2^*\}$, and $Y_{i_2^*}$ is read into memory.

(4) Assume that we need to generate a total of K subsamples of size n. Then, the size of the index set is about $|\mathcal{S}_0| = O(nK)$. To avoid duplicated sampling, every sampled unit needs to be compared with every unit in $\mathcal{S}_0$. This leads to a computation cost of order $O(nK)$ for every sampled unit, on average. The total computation cost is of order $O\{(nK)^2\}$, on average, which is expensive. The process is illustrated in Figure 6.

(5) Lastly, if we conduct random sampling with replacement, we **avoid** the need to (a) keep updating $\mathcal{S}_0$ and determining whether $i_2^* \in \mathcal{S}_0$,

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

and (b) keep updating $\mathcal{S}_1$. This makes our proposed method computationally more efficient.

To summarize, compared with subsampling with replacement, subsampling without replacement with massive data sets is practically challenging. Therefore, for massive data sets on a hard drive, we prefer sampling methods with replacement. In this case, no recording and comparison operations are needed. Next, to further demonstrate this point, we develop an experiment to compare sampling with and without replacement on a hard drive. To this end, we generate i.i.d. $X_i = (X_{i_1}, X_{i_2})$ from a standard bivariate normal distribution, with $N = 10^9$. The parameter of interest is the population mean μ. To estimate μ, the sample mean is calculated based on the two sampling strategies. We use $\widehat{\mu}_{rep}$ and $\widehat{\mu}_{worep}$ to represent the estimators based on sampling with and without replacement, respectively. We repeat the experiment $R = 100$ times. Then, the average mean square error (MSE) and time cost (TC) for the sampling are reported for both sampling strategies across R replications. The results are summarized in Table 1.

From Table 1, we draw the following conclusions. First, the MSE values of the two sampling strategies are comparable, and they both decrease with increasing n or K. However, the TC values of the two strategies are quite different. Sampling with replacement is much faster than sampling without

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Table 1: Comparison of sampling with and without replacement on a hard drive based on $R = 100$ simulation replications for various (n, K) combinations.

n	K	TC (s)		MSE ($\times 10^{-4}$)	
		$\widehat{\mu}_{rep}$	$\widehat{\mu}_{worep}$	$\widehat{\mu}_{rep}$	$\widehat{\mu}_{worep}$
100	50	0.13	0.32	4.04	4.04
	100	0.25	0.98	1.95	1.88
	200	0.51	3.34	0.91	0.93
500	50	0.29	4.79	0.78	0.79
	100	0.60	18.33	0.35	0.35
	200	1.17	71.65	0.18	0.18

replacement. As nK increases, the gap between the two strategies increases significantly. For instance, when $n = 500$ and $K = 200$, it takes only 1.17 seconds for the sampling with replacement method to complete the procedure, whereas the time required by the sampling without replacement method is almost 71.65 seconds.

3.2 An Algorithm for a GPU

We next develop a GPU-based algorithm for fast computation. Note that the proposed method exhibits many theoretically and practically useful properties. Theoretically, it guarantees the statistical efficiency of the subsample estimators with small subsample sizes. Practically, it is simple, automatic, and flexible. However, the associated computation cost

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

is expensive, because the new method requires subsampling K times and jackknifing n times for each subsample. Accordingly, its implementation on a central processing unit (CPU) might be inefficient, because a standard CPU usually has a limited number of computation cores. For example, the MacBook Pro (13-inch, 2020) uses an Intel Core i5 processor with only four cores. In contrast, a standard GPU may hold tens of thousands of cores. Accordingly, GPUs are extremely powerful tools for parallel computation (Krüger and Westermann, 2005; Che et al., 2008). Furthermore, our method (particularly the jackknifing part) is highly suitable for parallel computation. This inspires us to develop a GPU-based algorithm for the proposed method.

A standard GPU system has two unique features. To make full use of its computational power, we need to take both features into consideration. The first feature of the GPU system is that it suffers from two types of

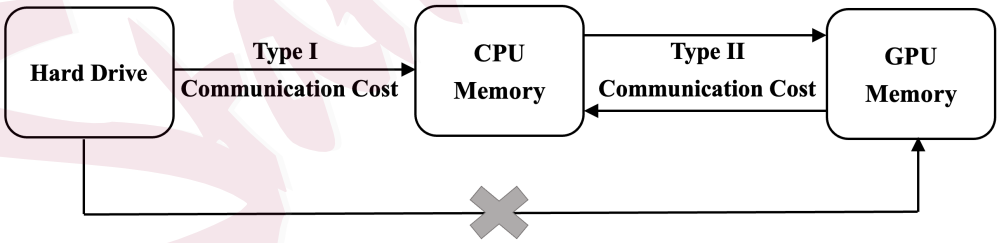


Figure 1: Two types of communication cost for a GPU system.

communication cost; see Figure 1. The first type refers to the time cost

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

required to transfer data from the hard drive to the CPU memory. This is a standard communication cost that is essentially required by any computation system. For our algorithm, this type of cost is primarily due to subsampling. The second type of communication cost refers to the time cost required to transfer data from the CPU memory to the GPU memory. The main purpose of transferring data from the CPU memory to the GPU memory is to prepare the data for the parallel execution of the jackknifing. Consequently, we consider that this part of the communication cost is mainly due to jackknifing. Note that current GPU architectures do not allow the GPU to directly read data from the hard drive. As a result, a good algorithm should simultaneously minimize both types of communication costs. Multiple communication between the hard drive, CPU memory and GPU memory should be avoided.

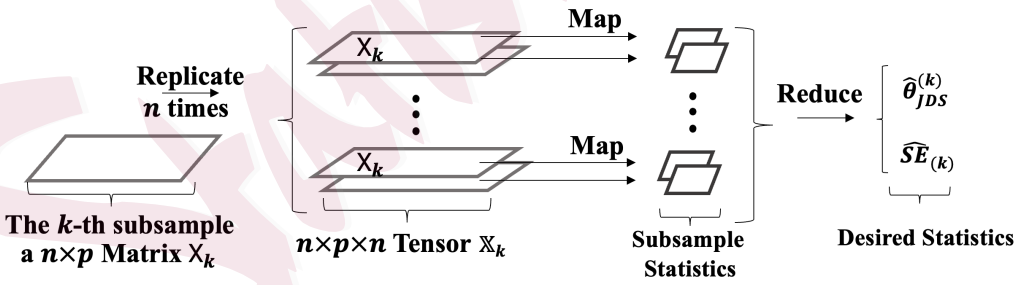


Figure 2: A graphical illustration of the proposed GPU algorithm.

The second unique feature is that GPU systems are extremely suitable

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

for tensor-type parallel computation, which makes full use of the parallel computation power of a GPU system. This suggests that the jackknifing computation should be formulated into a tensor-type computation problem. Specifically, we develop a three-step algorithm to implement the proposed method. The process is shown in Figure 2.

Input: Data $X_1, \ldots, X_N$ on a hard drive, $X_i \in \mathbb{R}^p$;
$g(\cdot)$: the function of interest of the moment;
n: the subsample size K: the number of subsamples;
Output: A JDS estimator $\widehat{\theta}_{JDS}$ and a JSE estimator $\widehat{\mathrm{SE}}$.
for $k \leftarrow 1$ **to** K **do** subsampling
Generate $\mathcal{S}_k \subset \mathbb{S}$, and place the $n \times p$ matrix X_k into the GPU memory ;
Compute $\widehat{\theta}^{(k)} \leftarrow g(\widehat{\mu}^{(k)})$ in the GPU memory;
Generate an $n \times p \times n$ tensor $\mathbb{X}_k$ in the GPU memory ;
Map the function $g(\cdot)$ to each channel of X_k, which leads to $\{\widehat{\theta}^{(k)}_{-j}, j \in \mathcal{S}_k\}$;
Compute $\widehat{\theta}^{(k)}_{JDS} = \widehat{\theta}^{(k)} - (n-1)\{n^{-1}\sum_{j=1}^{n}\widehat{\theta}^{(k)}_{-j} - \widehat{\theta}^{(k)}\}$ and
$\widehat{\mathrm{SE}}^2_{(k)} = \sum_{j\in\mathcal{S}_k}\left(\widehat{\theta}^{(k)}_{-j} - \widehat{\theta}^{(k)}\right)^2$ in the GPU memory ;
end
Compute $\widehat{\theta}_{JDS} = K^{-1}\sum_{k=1}^{K}\widehat{\theta}^{(k)}_{JDS}$ and
$\widehat{\mathrm{SE}}^2 = (1/K + n/K)K^{-1}\sum_{k=1}^{K}\widehat{\mathrm{SE}}^2_{(k)}$ in the GPU memory;
return $\widehat{\theta}_{JDS}$ and $\widehat{\mathrm{SE}}^2$.

Algorithm 1: The GPU algorithm

First, we obtain the kth subsample $\mathcal{S}_k \subset \mathbb{S}$ from the hard drive, and place it in the CPU memory. With a slight abuse of notation, we assume that, X_i is a p-dimensional vector, for any $i \in \mathbb{S}$. Next, we formulate the

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

kth subsample into an $n \times p$ matrix format as $\mathsf{X}_k = [X_i, i \in S_k] \in \mathbb{R}^{n \times p}$. We pass X_k to the GPU memory and replicate X_k n times to construct a three-dimensional tensor $\mathbb{X}_k = \left[\mathsf{X}_k, \ldots, \mathsf{X}_k\right] \in \mathbb{R}^{n \times p \times n}$. Next, we define a function to compute the intended statistics with jackknifing. We map this function to different channels of $\mathbb{X}_k$, where each X_k represents one channel of $\mathbb{X}_k$. By doing so, the jackknifing computation can be executed by the GPU system in a parallel fashion. We collect the computation results from each channel, and reduce them to the desired statistics $\widehat{\theta}_{JDS}^{(k)}$ and $\widehat{\mathrm{SE}}_{(k)}^2 = \sum_{j \in \mathcal{S}_k} \left(\widehat{\theta}_{-j}^{(k)} - \widehat{\theta}^{(k)}\right)^2$ for the kth subsample. Then, we obtain the final estimators. This leads to the entire GPU algorithm. The details are provided in Algorithm 1.

3.3 The Communication and Computation Cost

To evaluate the finite-sample performance of the proposed method, we present a number of numerical experiments. We first consider how to generate the whole sample with a very large $N = 10^9$. For every $1 \leq i \leq N$, we generate an i.i.d. two-dimensional random variable $X_i = (X_{i1}, X_{i2})^\top$ from a bivariate normal distribution with mean zero and covariance $\Sigma = \{\sigma_{ij}\}_{2\times 2}$, where $\sigma_{11} = 25, \sigma_{12} = \sigma_{21} = 10$, and $\sigma_{22} = 5$. We then define the parameter

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

of interest as the correlation coefficient $\mathrm{Corr}(X_{i1}, X_{i2})$, as follows:

$$\theta = \mathrm{Corr}(X_{i1}, X_{i2}) = \frac{\mathrm{Cov}(X_{i1}, X_{i2})}{\sqrt{\mathrm{var}(X_{i1})\,\mathrm{var}(X_{i2})}} = \frac{2}{\sqrt{5}}.$$

This parameter is a complex nonlinear function of various moments about X_i. Once the whole sample is generated, it is placed as a single file on the hard drive, requiring approximately 38.3 GB, which is clearly too large for the CPU memory. Once the data are stored on the hard drive, they are fixed for the remainder of the simulation experiments. In other words, we do not update the whole sample data set on the hard drive across different simulation replications. For a reliable evaluation, we replicate the subsequent experiment $M = 1000$ times. All computations are performed using TensorFlow 2.2.0 on a single GPU device (NVIDIA Tesla P100).

In this subsection, we focus on performance in terms of the time cost. We study both the communication cost and the computation cost. The communication cost can be divided further into two parts. The first part is the time cost required for transferring data from the hard drive to the CPU memory. The second part is the time cost required for transferring data from the CPU memory to GPU memory. Next, we vary the subsample size n from 100 to 3,000 and K from 10 to 200. We use $\mathcal{S}_k^{(m)} \subset \mathbb{S}$ to represent the

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

kth subsample obtained in the mth simulation replication. The time cost for obtaining $\mathcal{S}_k^{(m)}$ is recorded as $T_{1k}^{(m)}$. Based on $\mathcal{S}_k^{(m)}$, we obtain the matrix $\mathsf{X}_k^{(m)}$. We then transfer $\mathsf{X}_k^{(m)}$ from the hard drive to the CPU memory, where the associated time cost is recorded as $T_{2k}^{(m)}$. The computation cost required for computing $\widehat{\theta}_{JDS}$ and $\widehat{\mathrm{SE}}$ is given by $T_{3k}^{(m)}$. Consequently, the total time cost is given by $T_k^{(m)} = T_{1k}^{(m)} + T_{2k}^{(m)} + T_{3k}^{(m)}$. Their averages are obtained as $T_1 = M^{-1}\sum_{k,m} T_{1k}^{(m)}$, $T_2 = M^{-1}\sum_{k,m} T_{2k}^{(m)}$, and $T_3 = M^{-1}\sum_{k,m} T_{3k}^{(m)}$. Next, we examine their relationships with both K and n.

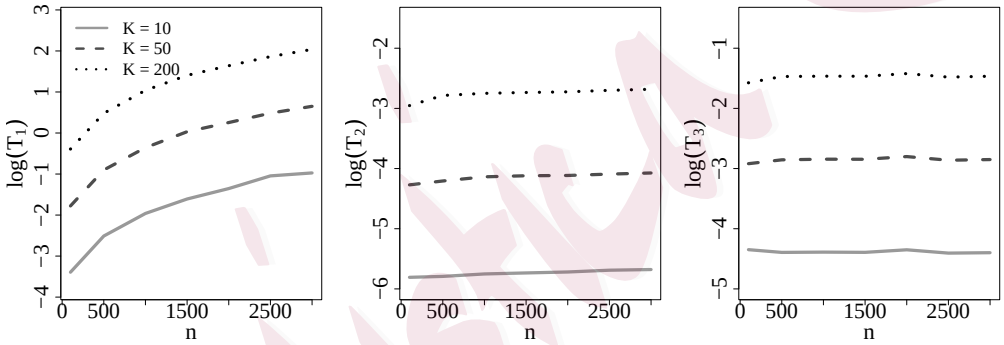


Figure 3: The log-transformed time cost for different (n, K) combinations with $K = 10, 50$ and 200. The communication cost due to subsampling T_1 is given in the left panel. The communication cost due to jackknifing T_2 is reported in the middle panel. The computation cost T_3 is presented in the right panel. The reported time costs (in log-scale) are averaged based on $M = 1000$ simulations.

The detailed results are given in Figure 3. All types of time cost increase as the number of subsamples K increases. In particular, the communication cost required by the subsampling (i.e., T_1) is substantially larger than

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

the other two types of time cost. Comparatively, the communication cost required by the jackknifing (i.e., T_2) is the smallest. Note that although T_3 is significant in a CPU-only system, it is not significant in a GPU system. To understand this idea, consider the case of $K = 50$ and $n = 3000$. Here $T_1 = 1.917\ s, T_2 = 0.017\ s$ and $T_3 = 0.057\ s$. In addition, the middle and right panels of Figure 3 show that for a fixed total subsample size K, T_2 and T_3 remain almost unchanged as n increases. This result demonstrates the excellent parallel capability of a GPU-based system, and suggests that better computation efficiency can be achieved by setting the subsample size n to be as large as possible, given the amount of computer memory.

Next, we demonstrate the computational advantage of a GPU system. To this end, we define $T^{(m)}_{GPU}$ as the total time cost required by the mth simulation replication, except for the communication cost due to subsampling (such a cost is required by any computation system). We then execute the same algorithm on a CPU-only system (in our case, TensorFlow 2.2.0 can also be executed on a CPU-only system). This leads to the total time cost, except for the communication cost due to subsampling required by the CPU-only system, which is recorded as $T^{(m)}_{CPU}$. We compute their ratio for the mth replication as $R^{(m)} = T^{(m)}_{GPU}/T^{(m)}_{CPU}$, and define the averaged ratio as $\mathrm{AR} = M^{-1}\sum_{m=1}^{M} R^{(m)}$. The relationships of the log-transformed AR val-

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

ues for different (n, K) combinations are reported in Figure 4, which shows that the log(AR) values are always smaller than zero. This suggests that the computational time cost required by a GPU-based system is always smaller than that required by a CPU-only system, on average. In fact, the reported log(AR) values seem to be rather insensitive to the number of subsamples (i.e., K). Furthermore, for a fixed number of subsamples K, the log(AR) value decreases as the subsample size n increases. This is because a larger n requires a higher computation cost. Accordingly, the parallel computational power of a GPU system can be better demonstrated. For instance, considering the case with $K = 50$ and $n = 3000$, the averaged time cost of the GPU system is approximately 0.074 s, while that of the CPU system is approximately 5.191 s. The corresponding AR value is AR = 0.014. This suggests that the computation cost required by a GPU system is only approximately 1.4% that of a CPU system, on average.

3.4 Simulation Results of the JSE Estimator

In this subsection, we focus on the finite-sample performance of the JSE estimator $\widehat{\mathrm{SE}}$. To this end, we follow the simulation setup in the previous subsection. Note that the data on the hard drive are generated only once to conserve time. Once the data are generated, we replicate the

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

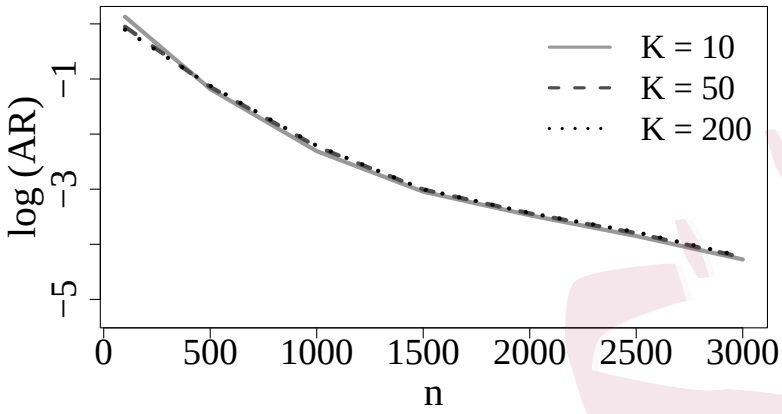
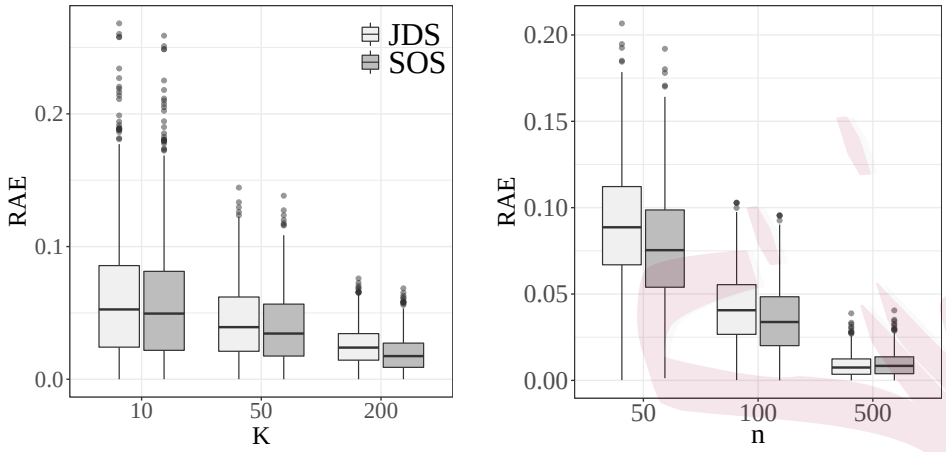


Figure 4: Comparison of the computation efficiency between a GPU system and a CPU system. The AR is reported in log-scale based on $M = 1000$ simulation replications. The numbers of subsamples are fixed to $K = 10, 50,$ and 200.

experiments $M = 1000$ times based on the same whole sample data set. Specifically, for the mth replication, we obtain an SOS estimator $\widehat{\theta}^{(m)}_{SOS}$, a JDS estimator $\widehat{\theta}^{(m)}_{JDS}$, and a JSE estimator $\widehat{\mathrm{SE}}^{(m)}$. Define SE_{SOS} and SE_{JDS} as the respective sample standard deviations of $\{\widehat{\theta}^{(m)}_{SOS}, m = 1, \ldots, M\}$ and $\{\widehat{\theta}^{(m)}_{JDS}, m = 1, \ldots, M\}$. Accordingly, SE_{SOS} and SE_{JDS} measure the variabilities of $\widehat{\theta}_{SOS}$ and $\widehat{\theta}_{JDS}$, respectively, conditional on the whole sample data set on the hard drive. Because we have $N \gg nK$, they should be good approximations of the true variabilities of $\widehat{\theta}_{SOS}$ and $\widehat{\theta}_{JDS}$; see Theorems 2 and 3. Next, for the mth replication, we define the relative absolute errors as $\mathrm{RAE}^{(m)}_{SOS} = \left|\widehat{\mathrm{SE}}^{(m)}/\,\mathrm{SE}_{SOS} - 1\right|$ and $\mathrm{RAE}^{(m)}_{JDS} = \left|\widehat{\mathrm{SE}}^{(m)}/\,\mathrm{SE}_{JDS} - 1\right|$, shown

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

in box plots in Figure 5.



(a) $n = 100$ (b) $K = 100$

Figure 5: Box plots of the RAE values for the JDS (light box) and SOS (dark box) estimators. The left panel corresponds to the case with n fixed to $n = 100$. The right panel corresponds to the case with K fixed to $K = 100$. Each box is summarized based on $M = 1000$ simulation replications.

The left panel of Figure 5 shows that, the RAE values of the SOS and JDS estimators are similar. They both decrease to zero as K increases, suggesting that a larger K leads to more accurate JSE estimators, under the condition that n is fixed. Qualitatively similar patterns are also observed for the right panel. We find that a larger n leads to a more accurate JSE estimation, under the condition that K is fixed. To summarize, both box plots in Figure 5 suggest that the proposed JSE estimator is consistent as $nK \to \infty$.

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

3.5 Simulation Results of the JDS Estimator

Finally, we evaluate the finite-sample performance of the point estimation $\widehat{\theta}_{JDS}$ and its statistical inference in terms of the confidence interval. For comparison, we also evaluate the SOS estimator $\widehat{\theta}_{SOS}$. Specifically, following the simulation setup in the previous subsection, we replicate the experiments $M = 1000$ times based on the same whole data set on the hard drive. For the mth replication, we calculate the JDS estimator $\widehat{\theta}_{JDS}^{(m)}$ and the corresponding JSE estimator $\widehat{\mathrm{SE}}^{(m)}$. This leads to a total of M estimators $\{(\widehat{\theta}_{JDS}^{(m)}, \widehat{\mathrm{SE}}^{(m)}) : 1 \leq m \leq M\}$. Based on these estimators, the average bias can be computed as $\mathrm{Bias} = M^{-1}\sum_{m=1}^{M}\big(\widehat{\theta}_{JDS}^{(m)} - \theta\big)$, and the corresponding standard error (SE) can be obtained. In addition, for each estimator $\widehat{\theta}_{JDS}^{(m)}$, a $(1-\alpha)$th level confidence interval for θ is constructed as $\mathrm{CI}^{(m)} = \big[\widehat{\theta}_{JDS}^{(m)} - \widehat{\mathrm{SE}}^{(m)} Z_{1-\alpha/2}, \widehat{\theta}_{JDS}^{(m)} + \widehat{\mathrm{SE}}^{(m)} Z_{1-\alpha/2}\big]$, where $\alpha = 0.05$ and Z_α represents the lower α quantile of the standard normal distribution. The empirical coverage probabilities are then also evaluated as $\mathrm{ECP}_{JDS} = M^{-1}\sum_{m=1}^{M} I(\theta \in \mathrm{CI}^{(m)})$, where $I(\cdot)$ is the indicator function. The SOS estimator $\widehat{\theta}_{SOS}$ is evaluated similarly. The detailed results are given in Table 2.

From Table 2, we find that the two estimators perform similarly in terms of the standard error (SE) for various (n, K) combinations. However, they

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

are quite different in terms of their bias. The bias of the SOS estimator $\widehat{\theta}_{SOS}$ is much larger than that of $\widehat{\theta}_{JDS}$. For example, when $n = 200$ and $K = 200$, the bias of $\widehat{\theta}_{SOS}$ is 4.49×10^{-4}, whereas that of $\widehat{\theta}_{JDS}$ is only 1.1×10^{-5}. Thus, the former is approximately 40 times larger than the latter. Moreover, the bias of $\widehat{\theta}_{SOS}$ is quite comparable to its standard error. As a result, the confidence interval of $\widehat{\theta}_{SOS}$ is poor, because the corresponding ECP is significantly smaller than 95%. In contrast, the confidence interval of the JDS estimator is good, because the corresponding ECP values of $\widehat{\theta}_{JDS}$ are quite close to 95%.

Table 2: Comparison of the SOS estimator $\widehat{\theta}_{SOS}$ and the JDS estimator $\widehat{\theta}_{JDS}$ based on $M = 1000$ simulation replications for various (n, K) combinations.

K	SE ($\times 10^{-3}$)		Bias ($\times 10^{-3}$)		ECP (%)	
	$\widehat{\theta}_{SOS}$	$\widehat{\theta}_{JDS}$	$\widehat{\theta}_{SOS}$	$\widehat{\theta}_{JDS}$	$\widehat{\theta}_{SOS}$	$\widehat{\theta}_{JDS}$
			$n = 50$			
100	2.964	2.929	2.038	0.050	92.1	95.8
200	2.103	2.077	2.054	0.066	87.8	97.0
500	1.335	1.314	1.956	0.032	73.6	96.8
1000	0.972	0.959	1.907	0.078	50.8	95.4
			$n = 100$			
100	2.068	2.057	0.910	0.029	93.3	96.1
200	1.446	1.437	0.960	0.019	91.7	95.3
500	0.938	0.932	0.877	0.065	84.8	95.6
1000	0.672	0.667	0.904	0.038	72.9	95.6
			$n = 200$			
100	1.436	1.432	0.487	0.029	93.7	94.8
200	1.029	1.025	0.449	0.011	92.9	95.4
500	0.656	0.654	0.442	0.017	89.4	95.0
1000	0.441	0.440	0.477	0.018	83.3	96.4

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

3.6 Real-Data Analysis

In this subsection, we study a U.S. airline data set, available on the official website of the American Statistical Association (ASA). The airline data set contains approximately 120 million records, and requires approximately 12 GB of space on a hard drive. Each record contains detailed information for one commercial flight in the United States between from October 1987 and April 2008. The data set contains 13 continuous variables and 16 categorical variables. For illustration, we focus on the 13 continuous variables. However, a significant portion of records are missing for many continuous variables. Only five have missing rates less than 10%: ActualElapsedTime (actual elapsed time), CRSElapsedTime (scheduled elapsed time), Distance, DepDelay (departure delay), and ArrDelay (arrival delay). Therefore, we focus on these five variables. For more detailed information on the variables, refer to the ASA official website at http://stat-computing.org/dataexpo/2009.

For each variable, we apply the following signed-log-transformation: $\log|x| \cdot \text{sign}(x)$. This transformation is conducted purely for illustration. Otherwise, many variables (e.g., ArrDelay) are so heavy tailed that the existence of finite moments becomes questionable. For each transformed variable, we examine the mean, standard deviation, and kurtosis. Their whole

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Table 3: Descriptive statistics for five continuous variables based on the whole airline data set after a signed-log transformation. The descriptive statistics are given by the sample mean (Mean), sample standard deviation (SD), and sample kurtosis (Kurt).

	Actual ElapsedTime	CRS ElapsedTime	Distance	DepDelay	ArrDelay
Mean	4.656	4.670	6.272	0.492	0.236
SD	0.525	0.513	0.777	1.905	2.463
Kurt	2.586	2.597	2.750	2.272	1.594

sample estimators are given in Table 3. These whole sample estimators are then treated as if they were the true parameters. Accordingly, simulation experiments can be conducted as in the previous subsections. In this case, we fixed $nK = 6 \times 10^4$, with different (n, K) combinations, and replicated the experiments $M = 1000$ times. The results are summarized in Table 4.

From Table 4, we obtain the following interesting observations. First, note that the sample mean is an exactly unbiased estimator for the mean. Accordingly, both the SOS and JDS estimators are unbiased. In fact, they are identical in this case. As a result, both estimators demonstrate identical simulation results, with ECP values very close to their nominal level of 95% for all five variables. Second, for the other two parameters (i.e., standard deviation and kurtosis), the sample estimators are no longer unbiased. Accordingly, the SOS and JDS estimators are no longer identical. As expected, both estimators are similar in terms of the standard error (SE). However,

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

they are quite different in terms of the empirical bias. Obviously, the bias of the SOS estimator is substantially larger than that of the JDS estimator for all reported cases. As a result, the ECP values of the SOS estimator depart significantly from their nominal level of 95%. In contrast, those of the JDS estimator remain very close to 95%. Consider, for example, the case of the kurtosis of ArrDelay, with $n = 200$ and $K = 300$. The ECP value of the SOS estimator is only 35.6%. In contrast, that of the JDS is 95.7%.

4. CONCLUSION

We have developed a novel statistical method for large data sets. The new method is designed particularly for practitioners with limited computational resources, and combines the ideas of subsampling and jackknifing. Subsampling allows our method to work with large data sets. Jackknifing further enhances this capability by significantly reducing the bias. To implement our method, we have provided an algorithm developed for GPU systems. We show theoretically that the resulting estimator can be as good as the whole sample estimator under very mild regularity conditions. Extensive numerical studies using simulations and a real data set are presented to demonstrate outstanding performance of the proposed method.

To conclude this paper, we discuss several interesting topics for future

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

research. First, the statistics considered in this work are relatively simple, representing nonlinear transformations of various moments. It would be of interest to develop similar methods for more general M estimators. Second, the data considered here are collected from independent samples. This makes the theoretical understanding of the resulting subsample estimator analytically simple. Thus, similar methods for data with a sophisticated dependence structure (e.g., spatial temporal data) are also worth studying.

Supplementary Material

The online Supplementary Material contains proofs of all theoretical results in the main text.

Acknowledgments

We thank the editor, associate editor, and two referees for their insightful comments. Xuening Zhu's research was supported by the National Natural Science Foundation of China (nos. 11901105, 71991472, U1811461), Shanghai Science and Technology Commission Grant (No. 17JC1420200), and Shanghai Sailing Program for Youth Science and Technology Excellence (19YF1402700). Hansheng Wang's research was partially supported by the National Natural Science Foundation of China (no. 11831008).

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

References

Cameron, A. C. and Trivedi, P. K. (2005), *Microeconometrics: methods and applications*, Cambridge university press.

Che, S., Michael, B., Meng, J., Tarjan, D., Sheaffer, J., and Skadron, K. (2008), "A performance study of general-purpose applications on graphics processors using CUDA," *Journal of Parallel and Distributed Computing*, 68, 1370–1380.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlós, T. (2011), "Faster least squares approximation," *Numerische mathematik*, 117, 219–249.

Efron, B. and Stein, C. (1981), "The jackknife estimate of variance," *The Annals of Statistics*, 586–596.

Huang, C. and Huo, X. (2015), "A distributed one-step estimator," *arXiv preprint arXiv:1511.01443*.

Jordan, M. I., Lee, J. D., and Yang, Y. (2019), "Communication-efficient distributed statistical inference," *Journal of the American Statistical Association*, 114, 668–681.

Krüger, J. and Westermann, R. (2005), "Linear algebra operators for GPU implementation of numerical algorithms," in *ACM SIGGRAPH 2005 Courses*, pp. 234–es.

Lehmann, E. L. and Casella, G. (2006), *Theory of point estimation*, Springer Science & Business Media.

Ma, P., Mahoney, M. W., and Yu, B. (2015), "A statistical perspective on algorithmic leverag-

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

ing," *The Journal of Machine Learning Research*, 16, 861–911.

Ma, P., Zhang, X., Xing, X., Ma, J., and Mahoney, M. (2020), "Asymptotic analysis of sampling estimators for randomized numerical linear algebra algorithms," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 1026–1035.

Mahoney, M. W. (2011), "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, 3, 123–224.

Mcdonald, R., Mohri, M., Silberman, N., Walker, D., and Mann, G. S. (2009), "Efficient large-scale distributed training of conditional maximum entropy models," in *Advances in neural information processing systems*, pp. 1231–1239.

Quenouille, M. H. (1949), "Approximate Tests of Correlation in Time-Series," *Journal of the Royal Statistical Society*, 11, 68–84.

Shao, J. (2003), "Mathmetical Statistics," *New York, Springer*.

Suresh, A. T., Felix, X. Y., Kumar, S., and McMahan, H. B. (2017), "Distributed mean estimation with limited communication," in *International Conference on Machine Learning*, PMLR, pp. 3329–3337.

Wang, F., Huang, D., Zhu, Y., and Wang, H. (2020), "Efficient Estimation for Generalized Linear Models on a Distributed System with Nonrandomly Distributed Data," *arXiv preprint arXiv:2004.02414*.

Wang, H. (2019), "More Efficient Estimation for Logistic Regression with Optimal Subsamples,"

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Journal of Machine Learning Research, 20, 1–59.

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

Wu, C.-F. J. (1986), "Jackknife, bootstrap and other resampling methods in regression analysis," *the Annals of Statistics*, 14, 1261–1295.

Yu, J., Wang, H., Ai, M., and Zhang, H. (2020), "Optimal Distributed Subsampling for Maximum Quasi-Likelihood Estimators with Massive Data," *Journal of the American Statistical Association*, Accepted on May 20, 2020.

Zhang, Y., Duchi, J. C., and Wainwright, M. J. (2013), "Communication-efficient algorithms for statistical optimization," *The Journal of Machine Learning Research*, 14, 3321–3363.

Zhu, X., Pan, R., Wu, S., and Wang, H. (2021), "Feature Screening for Massive Data Analysis by Subsampling," *Journal of Business & Economic Statistics*, 0, 1–31.

Zinkevich, M., Weimer, M., Smola, A. J., and Li, L. (2011), "Parallelized Stochastic Gradient Descent," in *Advances in Neural Information Processing Systems 23: Conference on Neural Information Processing Systems A Meeting Held December.*

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Guanghua School of Management, Peking University, Beijing, China

E-mail: shuyuan.w@pku.edu.cn

School of Data Science, Fudan University, Shanghai, China

E-mail:xueningzhu@fudan.edu.cn (Corresponding author)

Guanghua School of Management, Peking University, Beijing, China

E-mail: hansheng@pku.edu.cn

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

Table 4: Simulation results for the airline data set based on $M = 1000$ simulation replications. The parameters of interest include the mean (Mean), standard deviation (SD), and kurtosis (Kurt) for signed-log-transformed variables. Both the SOS estimator $\widehat{\theta}_{SOS}$ and the JDS estimator $\widehat{\theta}_{JDS}$ are compared in terms of the Bias($\times 10^{-3}$), SE($\times 10^{-2}$), and ECP(%). The nominal ECP level is 95%.

		Actual ElapsedTime		CRS ElapsedTime		Distance		DepDelay		ArrDelay	
		SOS	JDS	SOS	JDS	SOS	JDS	SOS	JDS	SOS	JDS
		$n = 300, \quad K = 200$									
Mean	Bias	0.03	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.26	0.26
	SE	0.22	0.22	0.22	0.22	0.33	0.33	0.77	0.77	0.98	0.98
	ECP	94.6	94.6	94.1	94.1	94.1	94.1	94.6	94.6	95.7	95.7
SD	Bias	1.30	0.08	1.27	0.07	1.94	0.08	4.24	0.04	4.76	0.02
	SE	0.13	0.13	0.13	0.13	0.21	0.21	0.42	0.42	0.37	0.37
	ECP	83.9	95.7	84.3	95.6	84.7	96.5	86.2	96.0	78.6	96.5
Kurt	Bias	6.82	0.32	6.79	0.25	16.85	0.56	4.31	0.14	7.89	0.02
	SE	1.58	1.78	1.71	2.10	1.98	2.12	1.20	1.21	0.49	0.49
	ECP	89.6	93.7	89.9	94.7	85.2	94.7	94.3	95.4	66.2	95.2
		$n = 200, \quad K = 300$									
Mean	Bias	0.03	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.26	0.26
	SE	0.22	0.22	0.22	0.22	0.33	0.33	0.77	0.77	0.98	0.98
	ECP	94.8	94.8	94.2	94.2	94.2	94.2	94.6	94.6	95.7	95.7
SD	Bias	1.92	0.08	1.87	0.07	2.88	0.08	6.34	0.02	7.11	0.00
	SE	0.13	0.13	0.13	0.13	0.21	0.21	0.41	0.42	0.37	0.37
	ECP	71.4	95.3	72.6	95.2	72.5	96.2	70.2	96.2	55.7	96.6
Kurt	Bias	10.35	0.25	10.24	0.12	25.42	0.23	6.31	0.19	1.79	0.08
	SE	1.51	1.75	1.57	1.98	1.95	2.16	1.20	1.23	0.49	0.50
	ECP	85.4	93.8	85.0	94.8	75.3	92.6	93.0	96.1	35.6	95.7

Statistica Sinica: Preprint
doi:10.5705/ss.202021.0257

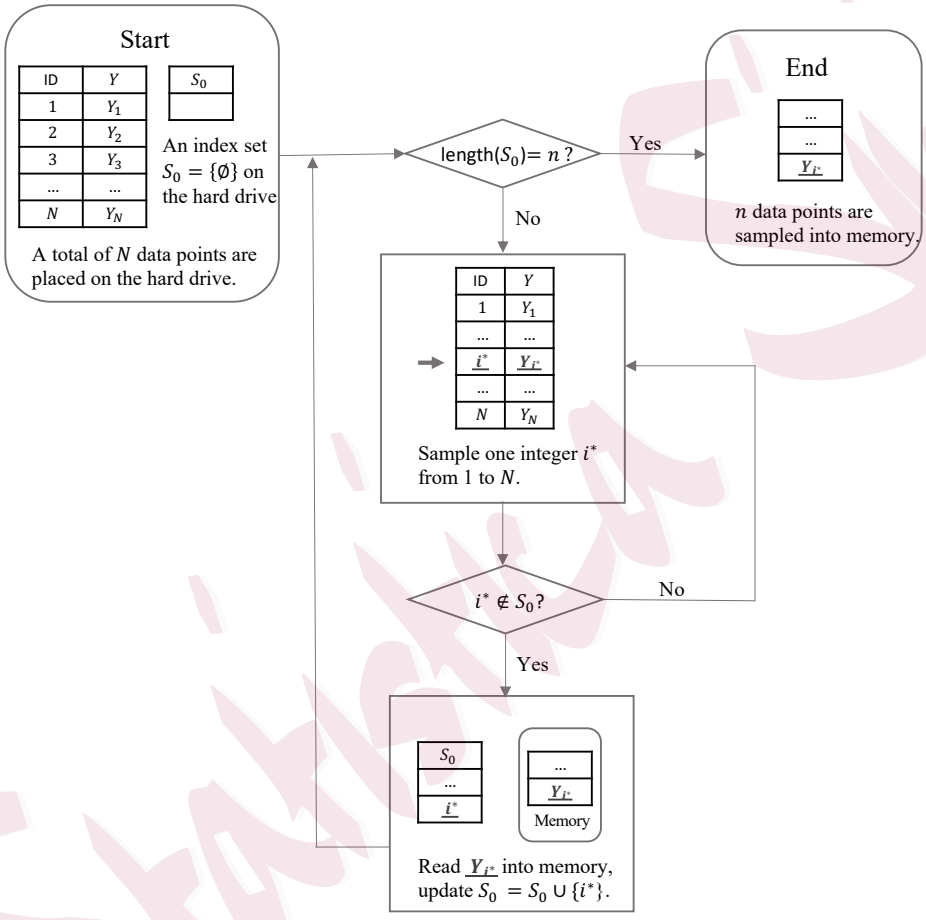


Figure 6: The process for sampling without replacement on a hard drive.