| Statistica Sinica Preprint No: SS-2018-0309 | |
|---|---|
| **Title** | Accounting for Factor Variables in Big Data Regression |
| **Manuscript ID** | SS-2018-0309 |
| **URL** | http://www.stat.sinica.edu.tw/statistica/ |
| **DOI** | 10.5705/ss.202018.0309 |
| **Complete List of Authors** | Tonglin Zhang and<br>Baijian Yang |
| **Corresponding Author** | Tonglin Zhang |
| **E-mail** | tlzhang@purdue.edu |
| Notice: Accepted version subject to English editing. | |

Statistica Sinica

# Accounting for Factor Variables in Big Data Regression

Tonglin Zhang and Baijian Yang

*Purdue University*

*Abstract:*

Continuous and factor explanatory variables are both important in linear regression. To fit a linear model with factor variables, the traditional implementation of the least squares approach is to define a number of dummy variables. This approach is difficult to apply to big data since the size of the design matrix can be significantly inflated by a factor variable, even if the number of factor levels is only moderately large. By treating the factor variable as an index, the article proposes a new approach, called the index least squares approach, to overcome the difficulty. Combining it with the technique of scanning data by rows, the index least squares approach can provide exact solutions to a group of linear models with factor variables simultaneously. Therefore, it avoids the memory barrier caused by the size of the design matrix. As the memory needed is unrelated to the number of observations, the index least squares approach can be used, even if the size of a massive data set is hundreds of times higher than the memory size of the computing system.

*Key words and phrases:* Big Data, Factor Variables, Index Least Squares, Parallel or Cluster Computation, Scanning Data by Rows, Index Array of Sufficient

Statistics.

## 1. Introduction

The traditional implementation of the least squares approach in linear regression, called the traditional implementation for short, is developed for small or moderate sized data. It cannot be applied to big data because of the memory and computational efficiency barriers (Lin and Xi, 2011; Meeker and Hong, 2014). Previous work on big data focuses on solutions to individual models. Examples include the regression cube technique (Chen and Dong, 2006), the divide-and-recombine approach (Guha, et. al, 2012), the sampling (or subsampling) approach (Dhillon, et. al, 2013; Ma and Sun, 2015; Wang, Yang, and Stufken, 2018), the updating estimation approach (Ener, 2009), the online updating approach (Schifano, et. al, 2016), and the aggregated estimating equation approach (Lin and Xi, 2011). In all of these, the main task is to develop an efficient algorithm to fit a specific model. If another model is studied, then the entire algorithm must be used again, which is extremely inefficient in fitting statistical models for big data. Another issue, which is ignored in all of these approaches, is the impact of types of explanatory variables. At least three types have been identified: continuous variables, nominal variables (also called factor variables), and ordinal variables. The traditional implementation to a model with factor

variables is to define a set of dummy variables. Although this works well in small or moderate data, it has a serious concern in big data as the size of the design matrix can be significantly inflated, even if the number of factor levels is only moderately large. The goal of the article is to propose a new approach, called the index least squares approach, such that it can efficiently and simultaneously provide exact solutions to a group of linear models with factor variables.

The size to determine whether a data set is big is relative to available computing resources. It is suggested that a data set be considered large if its size exceeds 20% of the size of the Random Access Memory (RAM) and massive if it exceeds 50% of the size of the RAM (Emerson and Kane, 2012). Memory barriers may be partially solved by the external memory algorithm (EMA) (Vitter, 2008). The EMA does not change mathematical formulae in the computation, instead it attempts to solve the out-of-memory problem by storing information used by the computation on a hard disk and process partial information to memory once at a time. After the usage is over, the EMA releases the memory and puts the partial information back to the hard disk. For an extremely big data set which cannot be loaded to memory of a single processor, a common solution in computer science is to partition the data set into a number of subsets by parallel or cluster computa-

tion (Battey, et. al, 2018; Lin and Xi, 2011; Meeker and Hong, 2014). Examples include MapReduce (Dean and Ghamawat, 2008; Miner and Shook, 2012) and Spark (Zaharia, et. al, 2010). In MapReduce or Spark, most of the information processing tasks consider a similar structure and the same computation is applied over a large number of records by many processors (Fernández, et. al, 2014). From the view of computational perspectives, much effort has been put into the most active and open source statistical R packages, such as biglm and RHIPE. Spark uses similar methods. However, one of the most important features of Spark is that it processes all information in memory. It can be hundreds of times faster than other parallel or cluster computation frameworks that process information in hard disks (Fang, Yang, and Zhang, 2017).

Statistical approaches and algorithms are important in formulating parallel or cluster computation. It is well-known that storage and computational sizes are two different problems. One may have a scenario that the size of data is not large but the memory needed in a computation is huge. Although the entire data set has been successfully loaded to memory, it cannot be analyzed by any traditional ways. We initially studied this scenario and gained some ideas. Assume that a regression model with $n$ observations has one response variable, $q$ continuous explanatory variables, and one

factor variable. The factor variable may be constructed by original levels of a factor variable or combined levels of a few factor variables. Therefore, the number of factor levels, denoted by $I$, may be large. If the factor variable is ignored, then the number of columns of the design matrix in the main-effects model is $q$. If the factor variable is studied, then the number of columns of the design matrix can be as high as $qI$. A computer needs $O(nq)$ memory size to load the data set, but it needs $O(nqI)$ memory size to construct the design matrix in the traditional implementation. For instance, if $n = 10^6$, $q = 10^2$, and $I = 10^2$, then the size of the data is about 800MB, but the size of the design matrix is about 80GB, implying that the traditional implementation cannot be used.

An extremely difficult situation can appear in linear regression with factor variables. Such a situation may still be present in parallel or cluster computation. The situation can be serious even if the number of factor levels is only moderately large. We propose a new approach, called the index least squares approach, to overcome the difficulty. The idea is motivated from the technique of scanning data by rows (Zhang and Yang, 2017a), where only individual rows are loaded sequentially from hard disks to memory. With just one access of the entire data set, exact solutions to a group of statistical models can be simultaneously derived. With a

slight modification, it can also be applied to variable or feature selection (Yang and Zhang, 2016a,b), penalized likelihood (Zhang and Yang, 2017b), and dimension reduction (Zhang and Yang, 2016, 2018). A great advantage is that the technique can be easily combined with well-known parallel or cluster computation frameworks, such as MapReduce and Spark. The index least squares approach is developed under the technique of scanning data by rows. It can be used to fit linear models with factor variables. Since the design matrix is sparse, we compare our approach with the sparse matrix approach (Pinar and Heath, 1999) in our simulation studies. An advantage is that the index least squares approach only uses sufficient statistics but the sparse matrix approach needs the entire data.

The remainder of this article is organized as follows. In Section 2, we briefly review the traditional implementation of the least squares approach with factor variables. In Section 3, we review the technique of scanning data by rows without factor variables. In Section 4, we propose our approach. In Section 5, we migrate our approach to parallel or cluster computation. In Section 6, we evaluate the performance of our approach with comparison to the traditional implementation and the sparse matrix approaches by simulations. In Section 7, we apply our approach to an airline data set. In Section 8, we provide a discussion.

## 2. Traditional Implementation

Based on individual observations of a data set with $\tilde{p}$ explanatory variables and the first one for the intercept, a linear regression model is proposed as

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i \tag{2.1}$$

for all $i \in \{1, \cdots, n\}$, where $y_i$ is a numeric value for the $i$th record of the response, $\mathbf{x}_i = (1, x_{i1}, \cdots, x_{i(p-1)})^\top$ is a vector constructed from the $i$th record of explanatory variables, $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_{p-1})^\top$ is a parameter vector for regression coefficients, $\epsilon_i \sim^{iid} N(0, \sigma^2)$ is the error term, and $n$ is the sample size. In this setting, we have $p = \tilde{p}$ if all explanatory variables are continuous and only main-effects are considered, or $p = \tilde{p}(\tilde{p}+1)/2$ if all of the interaction-effects are also considered. Therefore, $p$ varies in (2.1) subjective to interests.

The traditional implementation of the least squares approach, called the traditional implementation for short, is proposed under the matrix expression of (2.1) as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2.2}$$

where $\mathbf{y} = (y_1, \cdots, y_n)^\top$ is the response vector, $\mathbf{X} = (\mathbf{x}_1^\top, \cdots, \mathbf{x}_n^\top)^\top$ is the design matrix, and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ is the error vector. The least squares estimator (LSE) or the uniform minimum variance unbiased es-

timator (UMVUE) of $\boldsymbol{\beta}$ is $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$. The UMVUE of $\sigma^2$, which is also the MSE of the model, is $\hat{\sigma}^2 = \mathbf{y}^\top[\mathbf{I} - \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top]\mathbf{y}/(n-p)$. For significance of regression coefficients, one also needs $\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2(\mathbf{X}^\top\mathbf{X})^{-1}$. To ensure the existence of $\hat{\boldsymbol{\beta}}$ and $\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\beta}})$, one needs to assume that $p < n$ and $\mathbf{X}$ is full rank (i.e., $\mathrm{rank}(\mathbf{X}) = p$).

The traditional implementation can also be applied when factor variables are involved. Assume that a model contains $q$ continuous explanatory variables and one factor variable with $I$ levels. Let $n_i$ be the number of observations under the $i$th level of the factor variable. Then, variables of the model can be expressed as

$$\mathcal{D} = \{(y_{ij}, \mathbf{x}_{ij}^\top, i) : i = 1, \cdots, I, j = 1, \cdots, n_i\}, \tag{2.3}$$

where $\mathbf{x}_{ij} = (1, x_{ij1}, \cdots, x_{ij(q-1)})^\top$ represents the $(i,j)$th vector of continuous explanatory variables.

A common way is to take interaction-effects between a few continuous explanatory variables and the factor variable into account. Without loss of generality, we assume that the interaction-effects between the former $q_0$ continuous explanatory variables and the factor variable are included but the next $q - q_0$ are not. By the dummy variable approach, a baseline

ANOCVA (analysis of covariance) model is proposed as

$$y_{ij} = \mathbf{w}_{ij}^\top \boldsymbol{\alpha} + \mathbf{w}_{ij}^\top \boldsymbol{\omega}_i + \mathbf{z}_{ij}^\top \boldsymbol{\delta} + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma^2) \tag{2.4}$$

for $i = 1, \cdots, I$ and $j = 1, \cdots, n_i$, where $\mathbf{w}_{ij} = (1, x_{ij1}, \cdots, x_{ij(q_0-1)})^\top$ represents the vector of explanatory variables included in the interaction-effects, $\mathbf{z}_{ij} = (x_{ijq_0}, \cdots, x_{ij(q-1)})^\top$ represents that of those not included, $\boldsymbol{\alpha} = (\alpha_0, \cdots, \alpha_{q-1})^\top$, $\boldsymbol{\omega}_1 = \mathbf{0}$, $\boldsymbol{\omega}_i = (\omega_{i0}, \cdots \omega_{i(q_0-1)})^\top$ when $i \neq 1$, and $\boldsymbol{\delta} = (\delta_{q_0}, \cdots, \delta_{q-1})^\top$. Let $\mathbf{y}_i = (y_{i1}, \cdots, y_{in_i})^\top$, $\mathbf{W}_i = (\mathbf{w}_{i1}^\top, \cdots, \mathbf{w}_{in_i}^\top)^\top$, and $\mathbf{Z}_i = (\mathbf{z}_{i1}^\top, \cdots, \mathbf{z}_{in_1}^\top)^\top$. Then, (2.4) becomes

$$\mathbf{y}_i = \mathbf{W}_i \boldsymbol{\alpha} + \mathbf{W}_i \boldsymbol{\omega}_i + \mathbf{Z}_i \boldsymbol{\delta} + \boldsymbol{\epsilon}_i, \boldsymbol{\epsilon}_i \sim^{ind} N(\mathbf{0}, \sigma^2 \mathbf{I}_{n_i}). \tag{2.5}$$

To apply the traditional implementation, one needs to define $\mathbf{X}$ in (2.2) as

$$\mathbf{X} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{Z}_1 \\ \mathbf{W}_2 & \mathbf{W}_2 & \cdots & \mathbf{0} & \mathbf{Z}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{W}_I & \mathbf{0} & \cdots & \mathbf{W}_I & \mathbf{Z}_I \end{pmatrix} \tag{2.6}$$

with $\boldsymbol{\beta} = (\boldsymbol{\alpha}^\top, \boldsymbol{\omega}_2^\top, \cdots, \boldsymbol{\omega}_I^\top, \boldsymbol{\delta}^\top)^\top$. The setting $\boldsymbol{\omega}_1 = \mathbf{0}$ in (2.4) can make $\mathbf{X}$ in (2.2) full rank, implying that the traditional implementation can be used.

Although it performs well in small or moderate data, the traditional implementation is difficult to apply to big data due to the size of $\mathbf{X}$. The

reason is that the size of $\mathbf{X}$, which is $n(q_0 I + q - q_0)$, can be much higher than the size of observed data. Since $n$ is often large (e.g., $n \geq 10^7$), the size of $\mathbf{X}$ can be over a few hundred GB if $I$ is only moderately large.

We have identified two distinct situations. In the first, an extremely large $n$ but only a small or moderately large $p$ are induced. This is common if factor variables are not involved or factor variables with only a few levels are involved. Since the size of $p$ is not a concern, previous parallel or cluster computation algorithms via MapReduce or Spark can be applied. In the second, an extremely large $p$ is induced although $n$ may be even larger. This may appear if factor variables with many factor levels are involved. Note that factor levels in (2.3) may be constructed by combinations of observed factor levels of several factor variables. The number of combined levels can be extremely large. If (2.2) is applied to parallel or cluster computation, then the size of individual $\mathbf{X}$ in subsets may also be large. Therefore, the presence of factor variables can significantly affect approaches in big data.

## 3. Scanning Data By Rows

In the traditional implementation, such as that used by R, the first step is to load the entire data to memory. Statistical approaches and algorithms can only be used in the second step. If the first step fails, then it is im-

possible to carry out any fitting procedures in the second step. The EMA (Vitter, 2008), which is used by SAS, can partially overcome the difficulty. In the case where the memory needed in an algorithm is higher than that available in a computer, the EMA attempts to store information used by the algorithm to the hard disk, and processes partial information to memory once at a time. The EMA does not make any change of the algorithm, and it still faces the memory barrier caused by factor variables.

To overcome the difficulty, the technique of scanning data by rows is proposed. It attempts to combine loading data from hard disks to memory with statistical analysis together (Zhang and Yang, 2017a). It can provide exact solutions to a number of linear models simultaneously. The technique can be applied even if the size of data is hundreds of times higher than the memory size of the computing system. The idea of the technique is to focus on (2.1) and avoid (2.2). It uses a concept called *regression array of sufficient statistics (or regression array for short)* as

$$\mathcal{S} = (s_{yy}, \mathbf{s}_{xy}, \mathbf{S}_{xx}) = \left( \sum_{i=1}^{n} y_i^2, \sum_{i=1}^{n} y_i \mathbf{x}_i, \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top \right). \tag{3.1}$$

The regression array $\mathcal{S}$, which is well-known in the statistical literature (Klotz, 1995, e.g.), is an unstructured array because the sizes of its three components are not identical. It cannot be interpreted by usual ways of arrays or vectors.

The implementation of scanning data by rows based on the regression array is straightforward. Let $\mathcal{S}_m = (s_{m,yy}, \mathbf{s}_{m,xy}, \mathbf{S}_{m,xx}) = \left(\sum_{i=1}^m y_i^2, \sum_{i=1}^m y_i \mathbf{x}_i, \sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^\top\right)$ be the partial sum of previous $m$ rows. If $\mathcal{S}_m$ is derived after the $m$th row of the data is scanned, then $\mathcal{S}_{m+1} = (s_{m,yy} + y_{m+1}^2, \mathbf{s}_{m,xy} + y_{m+1}\mathbf{x}_{m+1}, \mathbf{S}_{m,xx} + \mathbf{x}_{m+1}\mathbf{x}_{m+1}^\top)$ is its updated value after the $(m+1)$th row is scanned. The final result of $\mathcal{S}$ is derived after the last row is scanned. Thus, we have

$$\hat{\boldsymbol{\beta}} = \mathbf{S}_{xx}^{-1}\mathbf{s}_{xy}, \tag{3.2}$$

and

$$\hat{\sigma}^2 = (s_{yy} - \mathbf{s}_{xy}^\top \mathbf{S}_{xx}^{-1}\mathbf{s}_{xy})/(n-p), \tag{3.3}$$

Moreover, we also have $\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2 \mathbf{S}_{xx}^{-1}$.

A great advantage is that the technique of scanning data by rows avoids the major difficulty caused by the size of $\mathbf{X}$. As rows are loaded sequentially, the computation of $\mathcal{S}$ only needs $O((p+1)^2)$ memory size, irrelevant to $n$. The technique can be used to an extremely large data set by a computer with a limit memory size if $p$ is not extremely large. As formulations of regression arrays depend on statistical models, statistical approaches are involved in scanning data. The time of the scanning is proportional to $n$, but the memory size is not. Once $\mathcal{S}$ is derived, the entire data set can be discarded in the rest computation. The technique of scanning data by rows is extremely efficient in fitting a group of linear models together.

The implementation of the technique of scanning data by rows relies on specifications of statistical models. It has been accepted for a long time that (2.2) is treated as an equivalent expression of (2.1) in traditional statistics, but this is not true under the framework of big data regression. The usage of (2.2) implies that the design matrix $\mathbf{X}$ can be used in fitting procedures, but the usage of (2.1) does not. The technique of scanning data by rows is developed under (2.1) but not (2.2), implying that (2.1) has more computational advantages than (2.2). In the following of this article, we treat (2.2) as an invalid expression.

## 4. Proposed Approach

We propose our approach based on the index array of sufficient statistics. We focus our approach on the case when only one factor variable is involved. The factor levels can be interpreted as original levels of a factor variable or combined levels of a few factor variables. We present the basic theory of our approach in Section 4.1. We study the multiple model problem in Section 4.2. We evaluate the connection of our approach with the traditional implementation in Section 4.3. We briefly introduce our idea to the case when two or more factor variables are involved in Section 4.4.

## 4.1 Index Least Squares

We treat the factor variable in (2.4) as an index variable and propose the index least squares approach. We modify (2.4) as

$$y_{ij} = \mathbf{w}_{ij}^\top \boldsymbol{\gamma}_i + \mathbf{z}_{ij}^\top \boldsymbol{\delta} + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma^2), i = 1, \cdots, I, j = 1, \cdots, n_i, \quad (4.1)$$

where $\boldsymbol{\gamma}_1 = \boldsymbol{\alpha}$ and $\boldsymbol{\gamma}_i = \boldsymbol{\omega}_i + \boldsymbol{\alpha}$ for $i \neq 1$. Then, (2.5) becomes

$$\mathbf{y}_i = \mathbf{W}_i \boldsymbol{\gamma}_i + \mathbf{Z}_i \boldsymbol{\delta} + \boldsymbol{\epsilon}_i, \boldsymbol{\epsilon}_i \sim^{ind} N(\mathbf{0}, \sigma^2 \mathbf{I}_{n_i}). \quad (4.2)$$

To apply the traditional implementation, one can define $\mathbf{X}$ in (2.2) as

$$\mathbf{X} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{Z}_1 \\ \mathbf{0} & \mathbf{W}_2 & \cdots & \mathbf{0} & \mathbf{Z}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{W}_I & \mathbf{Z}_I \end{pmatrix} \quad (4.3)$$

with $\boldsymbol{\beta} = (\boldsymbol{\gamma}_1^\top, \cdots, \boldsymbol{\gamma}_I^\top, \boldsymbol{\delta}^\top)^\top$. This can be used when the size of $\mathbf{X}$ is not large. For an index variable with $I$ levels when $\mathbf{X}$ is large, the index least squares approach is proposed under a modification of (4.1) as

$$y_{ij} = \mathbf{w}_{ij}^\top \boldsymbol{\gamma}_i + \mathbf{z}_{ij}^\top \boldsymbol{\delta}_i + \epsilon_{ij} = \tilde{\mathbf{w}}_{ij}^\top \tilde{\boldsymbol{\gamma}}_i + \epsilon_{ij}, \ \epsilon_{ij} \sim^{iid} N(0, \sigma^2), \quad (4.4)$$

for $i = 1, \cdots, I$ and $j = 1, \cdots, n_i$, where $\boldsymbol{\delta}_i = (\delta_{iq}, \cdots, \delta_{i(q-1)})^\top$, $\tilde{\mathbf{w}}_{ij} = (\mathbf{w}_{ij}^\top, \mathbf{z}_{ij}^\top)^\top$, and $\tilde{\boldsymbol{\gamma}}_i = (\boldsymbol{\gamma}_i^\top, \boldsymbol{\delta}_i^\top)^\top$. The sample size of the data is $n = \sum_{i=1}^I n_i$.

We assess (4.1) under (4.4) by testing

$$H_0 : \boldsymbol{\delta}_i = \boldsymbol{\delta}_{i'}, \ \forall \ i, i' = 1, \cdots, I. \tag{4.5}$$

We treated (4.4) as a full model and (4.1) as a reduced model. We wanted to fit both (4.1) and (4.4) together, and test whether (4.4) can be reduced to (4.1). Therefore, we need solutions to both estimation and hypotheses testing problems.

Let $\mathbf{x}_{ij} = (\mathbf{0}_q^\top, \cdots, \mathbf{0}_q^\top, \tilde{\mathbf{w}}_{ij}^\top, \mathbf{0}_q^\top, \cdots, \mathbf{0}_q^\top)^\top$ and $\boldsymbol{\beta} = (\tilde{\boldsymbol{\gamma}}_1^\top, \cdots, \tilde{\boldsymbol{\gamma}}_I^\top)^\top$, where $\mathbf{0}_q$ is the $q$-dimensional vector with all components equal to 0 and $\tilde{\mathbf{w}}_{ij}$ resides at the $i$th position of the expression of $\mathbf{x}_{ij}$. Then, (4.4) becomes

$$y_{ij} = \mathbf{x}_{ij}^\top \boldsymbol{\beta} + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma^2), \tag{4.6}$$

leading to the regression array of sufficient statistics as

$$\mathcal{S} = (s_{yy}, \mathbf{s}_{xy}, \mathbf{S}_{xx}) = \left( \sum_{i=1}^{I} \sum_{j=1}^{n_i} y_{ij}^2, \sum_{i=1}^{I} \sum_{i=1}^{n_i} y_{ij} \mathbf{x}_{ij}, \sum_{i=1}^{I} \sum_{j=1}^{n_i} \mathbf{x}_{ij} \mathbf{x}_{ij}^\top \right), \tag{4.7}$$

where $s_{yy}$ is a value, $\mathbf{s}_{xy}$ is a $qI$-dimensional vector, and $\mathbf{S}_{xx}$ is a $qI \times qI$-dimensional matrix. The implementation of (4.7) relies on the size of $\mathbf{S}_{xx}$. If $qI$ is small or moderately large (e.g., $qI \leq 10^4$) such that the size of $\mathbf{S}_{xx}$ (e.g., $\leq$ 800MB correspondingly) is lower than the memory size of the computer, then we can use (3.2) and (3.3) to fit (4.4). If $qI$ is extremely large (e.g. $qI \geq 10^5$), then the size of $\mathbf{S}_{xx}$ (e.g., $\geq$ 80GB correspondingly)

is higher than the memory size of the computing system. Thus, we cannot use (3.2) and (3.3).

We want to precisely fit (4.1) and (4.4) even when $qI$ is extremely large. The approach is developed under properties of the loglikelihood function of (4.4) as

$$\begin{aligned}\ell(\tilde{\boldsymbol{\gamma}}_1,\cdots,\tilde{\boldsymbol{\gamma}}_I,\sigma^2) = \; & -\tfrac{n}{2}\log(2\pi) - \tfrac{n}{2}\log\sigma^2 - \tfrac{1}{2\sigma^2}[s_{yy} - 2\textstyle\sum_{i=1}^{I}(\mathbf{s}_{i,wy}^\top\boldsymbol{\gamma}_i + \mathbf{s}_{i,zy}^\top\boldsymbol{\delta}_i) \\ & + \textstyle\sum_{i=1}^{I}(\boldsymbol{\gamma}_i^\top\mathbf{S}_{i,ww}\boldsymbol{\gamma}_i + 2\boldsymbol{\gamma}_i^\top\mathbf{S}_{i,wz}\boldsymbol{\delta}_i + \boldsymbol{\delta}_i^\top\mathbf{S}_{i,zz}\boldsymbol{\delta}_i)],\end{aligned} \tag{4.8}$$

where $s_{yy} = \sum_{i=1}^{I}\sum_{j=1}^{n_i} y_{ij}^2$, $\mathbf{s}_{i,wy} = \sum_{j=1}^{n_i} y_{ij}\mathbf{w}_{ij}$, $\mathbf{s}_{i,zy} = \sum_{j=1}^{n_i} y_{ij}\mathbf{z}_{ij}$, $\mathbf{S}_{i,ww} = \sum_{j=1}^{n_i}\mathbf{w}_{ij}\mathbf{w}_{ij}^\top$, $\mathbf{S}_{i,wz} = \sum_{j=1}^{n_i}\mathbf{w}_{ij}\mathbf{z}_{ij}^\top$, $\mathbf{S}_{i,zz} = \sum_{j=1}^{n_i}\mathbf{z}_{ij}\mathbf{z}_{ij}^\top$, $\tilde{\mathbf{s}}_{i,wy} = \sum_{j=1}^{n_i} y_{ij}\tilde{\mathbf{w}}_{ij}$, and $\tilde{\mathbf{S}}_{i,ww} = \sum_{j=1}^{n_i}\tilde{\mathbf{w}}_{ij}\tilde{\mathbf{w}}_{ij}^\top$. Then,

$$\tilde{\mathbf{S}}_{i,ww} = \begin{pmatrix} \mathbf{S}_{i,ww} & \mathbf{S}_{i,wz} \\ \mathbf{S}_{i,zw} & \mathbf{S}_{i,zz} \end{pmatrix} \tag{4.9}$$

and

$$\tilde{\mathbf{s}}_{i,wy} = \begin{pmatrix} \mathbf{s}_{i,wy} \\ \mathbf{s}_{i,wz} \end{pmatrix} \sim^{ind} N(\tilde{\mathbf{W}}_i\tilde{\boldsymbol{\gamma}}_i, \sigma^2\tilde{\mathbf{S}}_{i,ww}), \tag{4.10}$$

where $\mathbf{S}_{i,zw} = \mathbf{S}_{i,wz}^\top$ and $\tilde{\mathbf{W}}_i = (\tilde{\mathbf{w}}_{i1}^\top,\cdots,\tilde{\mathbf{w}}_{in_i}^\top)^\top$. If (4.5) holds, then

$$\tilde{\mathbf{s}}_{i,wy} = \begin{pmatrix} \mathbf{s}_{i,wy} \\ \mathbf{s}_{i,wz} \end{pmatrix} \sim^{ind} N\left[\begin{pmatrix} \mathbf{W}_i\boldsymbol{\gamma}_i \\ \mathbf{Z}_i\boldsymbol{\delta} \end{pmatrix}, \sigma^2\begin{pmatrix} \mathbf{S}_{i,ww} & \mathbf{S}_{i,wz} \\ \mathbf{S}_{i,zw} & \mathbf{S}_{i,zz} \end{pmatrix}\right]. \tag{4.11}$$

By the first expression of (4.8), we obtain the sufficient statistics of (4.4)

as

$$\mathcal{S} = (s_{yy}, \tilde{\mathbf{s}}_{1,wy}, \cdots, \tilde{\mathbf{s}}_{I,wy}, \tilde{\mathbf{S}}_{1,ww}, \cdots, \tilde{\mathbf{S}}_{I,ww}). \tag{4.12}$$

By the second expression of (4.8) together with (4.9) and (4.10), the above is equivalent to

$$\begin{aligned} \mathcal{S} = \quad & (s_{yy}, \mathbf{s}_{1,wy}, \cdots, \mathbf{s}_{I,wy}, \mathbf{s}_{1,zy}, \cdots, \mathbf{s}_{I,zy}, \\ & \mathbf{S}_{1,ww}, \cdots, \mathbf{S}_{I,ww}, \mathbf{S}_{1,zw}, \cdots, \mathbf{S}_{I,zw}, \mathbf{S}_{1,zz}, \cdots, \mathbf{S}_{I,zz}). \end{aligned} \tag{4.13}$$

**Definition 1.** The factor variable used in the derivation of (4.12) or (4.13) is called the index variable and the corresponding $\mathcal{S}$ is called the index array of sufficient statistics of (4.4) or the index array for short. The statistical approach based on the index array only is called the index least squares approach.

The size of the regression array given by (4.7) is $(qI)^2 + qI + 1$. The size of the index array given by (4.12) or (4.13) is $q^2I + qI + 1$. After adjusted by symmetry, the index array is equivalent to the minimum sufficient statistics of (4.4). The usage of the index array is more efficient than the usage of the regression array as the memory needed is reduced from a quadratic function of $I$ to a linear function of $I$. Therefore, our approach can be used even if $I$ is extremely large.

We propose the following algorithm to compute $\mathcal{S}$ given by (4.12) or (4.13). It assumes that a massive data set has already been stored on a

hard disk. The algorithm only needs $O(q^2I + qI + 1)$ memory size in the entire computation.

**Algorithm 1** Scan Data by Rows for $\mathcal{S}$ in (4.12) By A Single Processor

**Input**: A massive data set on a hard disk

**Output**: $s_{yy}$, $\tilde{\mathbf{s}}_{1,wy}, \cdots, \tilde{\mathbf{s}}_{I,wy}$, and $\tilde{\mathbf{S}}_{1,ww}, \cdots, \tilde{\mathbf{S}}_{I,ww}$

1: **procedure** UPDATING $\mathcal{S}$ BY ROWS

2: Let all of $s_{yy}$, $\tilde{\mathbf{s}}_{1,wy}, \cdots, \tilde{\mathbf{s}}_{I,wy}$, and $\tilde{\mathbf{S}}_{1,ww}, \cdots, \tilde{\mathbf{S}}_{I,ww}$ be zero.

3: **for** the $k$th row of the data **do** update $s_{yy} = s_{yy} + y_k^2$, and if the level of the index variable is $i$ then update $\tilde{\mathbf{s}}_{i,wy} = \tilde{\mathbf{s}}_{i,wy} + y_k\tilde{\mathbf{w}}_k$ and $\tilde{\mathbf{S}}_{i,ww} = \tilde{\mathbf{S}}_{i,ww} + \tilde{\mathbf{w}}_k\tilde{\mathbf{w}}_k^\top$ until the last row is scanned

4: **end for**

5: Output

6: **end procedure**

We propose the profile maximum likelihood approach, equivalent to the profile least squares approach, to estimate model parameters in (4.1) and (4.4). The computation must be completely carried out by the index array. It is enough for us to provide those for (4.1), as the results for (4.4) can be easily modified. We assume that only $\mathcal{S}$ given by (4.13) is available. As the intercept is included, the first component of $\mathbf{s}_{i,wy}$ is $\sum_{j=1}^{n_i} y_{ij}$ and the $(1,1)$th entry of $\mathbf{S}_{i,ww}$ is $n_i$. Then, we can compute $\bar{y}_i = \sum_{i=1}^{n} y_{ij}/n_i$ and

$n = \sum_{i=1}^{I} n_i$ by $\mathcal{S}$. As $\mathbf{S}_{i,ww}$ and $\mathbf{S}_{i,zz}$ are $q_0 \times q_0$ and $(q - q_0) \times (q - q_0)$-dimensional matrices, respectively, we can also obtain $q_0$ and $q$, implying that the residual degrees of freedom of (4.1) and (4.4) are available.

Let the loglikelihood function of (4.1) be $\ell(\boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_I, \boldsymbol{\delta}, \sigma^2)$. Maximizing $\ell(\boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_I, \boldsymbol{\delta}, \sigma^2)$ with respect to $\boldsymbol{\gamma}_i$ for a given $\boldsymbol{\delta}$, we obtain the conditional the MLE (also the conditional LSE) of $\boldsymbol{\gamma}_i$ given $\boldsymbol{\delta}$ as

$$\hat{\boldsymbol{\gamma}}_{i,\boldsymbol{\delta}} = \mathbf{S}_{i,ww}^{-1}(\mathbf{s}_{i,wy} - \mathbf{S}_{i,wz}\boldsymbol{\delta}). \tag{4.14}$$

Putting $\hat{\boldsymbol{\gamma}}_{i,\boldsymbol{\delta}}$ into the loglikelihood function, we obtain the profile loglikelihood function $\ell_P(\boldsymbol{\beta}, \sigma^2)$ of (4.1). Maximizing $\ell_P(\boldsymbol{\delta}, \sigma^2)$ with respect to $\boldsymbol{\delta}$, we derive the MLE (also the LSE) of $\boldsymbol{\delta}$ as

$$\hat{\boldsymbol{\delta}} = \left[\sum_{i=1}^{I}(\mathbf{S}_{i,zz} - \mathbf{S}_{i,zw}\mathbf{S}_{i,ww}^{-1}\mathbf{S}_{i,wz})\right]^{-1}\left[\sum_{i=1}^{I}(\mathbf{s}_{i,zy} - \mathbf{S}_{i,zw}\mathbf{S}_{i,ww}^{-1}\mathbf{s}_{i,wy})\right]. \tag{4.15}$$

By (4.11), we obtain the variance-covariance matrix of $\hat{\boldsymbol{\delta}}$ as

$$\mathrm{V}(\hat{\boldsymbol{\delta}}) = \sigma^2\left[\sum_{i=1}^{I}(\mathbf{S}_{i,zz} - \mathbf{S}_{i,zw}\mathbf{S}_{i,ww}^{-1}\mathbf{S}_{i,wz})\right]^{-1}.$$

By (4.14), we have the MLE (also the LSE) of $\boldsymbol{\gamma}_i$ as

$$\hat{\boldsymbol{\gamma}}_i = \hat{\boldsymbol{\gamma}}_{i,\hat{\boldsymbol{\delta}}} = \mathbf{S}_{i,ww}^{-1}(\mathbf{s}_{i,wy} - \mathbf{S}_{i,wz}\hat{\boldsymbol{\delta}}). \tag{4.16}$$

Still by (4.11), we obtain the variance-covariance matrix of $\hat{\boldsymbol{\gamma}}_i$ as

$$\mathrm{V}(\hat{\boldsymbol{\gamma}}_i) = \sigma^2\left\{\mathbf{S}_{i,ww}^{-1} + \mathbf{S}_{i,ww}^{-1}\mathbf{S}_{i,wz}\left[\sum_{k=1}^{I}(\mathbf{S}_{k,zz} - \mathbf{S}_{k,zw}\mathbf{S}_{k,ww}^{-1}\mathbf{S}_{k,wz})\right]^{-1}\mathbf{S}_{i,zw}\mathbf{S}_{i,ww}^{-1}\right\}.$$

The covariance matrix between $\hat{\boldsymbol{\delta}}$ and $\hat{\boldsymbol{\gamma}}_i$ is

$$\mathrm{Cov}(\hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\gamma}}_i) = -\sigma^2 \mathbf{S}_{i,ww}^{-1} \mathbf{S}_{i,wz} \left[ \sum_{k=1}^{I} (\mathbf{S}_{k,zz} - \mathbf{S}_{k,zw} \mathbf{S}_{k,ww}^{-1} \mathbf{S}_{k,wz}) \right]^{-1}.$$

The covariance matrix between $\hat{\boldsymbol{\gamma}}_i$ and $\hat{\boldsymbol{\gamma}}_{i'}$ for distinct $i$ and $i'$ is

$$\mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\gamma}}_{i'}) = \sigma^2 \mathbf{S}_{i,ww}^{-1} \mathbf{S}_{i,wz} \left[ \sum_{k=1}^{I} (\mathbf{S}_{k,zz} - \mathbf{S}_{k,zw} \mathbf{S}_{k,ww}^{-1} \mathbf{S}_{k,wz}) \right]^{-1} \mathbf{S}_{i',zw} \mathbf{S}_{i',ww}.$$

The sum of squares of errors (SSE) is

$$\begin{aligned} SSE = \; & s_{yy} - \textstyle\sum_{i=1}^{I} \mathbf{s}_{i,wy}^{\top} \mathbf{S}_{i,ww}^{-1} \mathbf{s}_{i,wy} - \left[ \sum_{i=1}^{I} (\mathbf{s}_{i,zy} - \mathbf{S}_{i,zw} \mathbf{S}_{i,ww}^{-1} \mathbf{s}_{i,wy}) \right]^{\top} \\ & \left[ \sum_{i=1}^{I} (\mathbf{S}_{i,zz} - \mathbf{S}_{i,zw} \mathbf{S}_{i,ww}^{-1} \mathbf{S}_{i,wz}) \right]^{-1} \left[ \sum_{i=1}^{I} (\mathbf{s}_{i,zy} - \mathbf{S}_{i,zw} \mathbf{S}_{i,ww}^{-1} \mathbf{s}_{i,wy}) \right]. \end{aligned}$$

The UMVUE of $\sigma^2$, which is also the MSE of the model, is $\hat{\sigma}^2 = SSE/[n - q - q_0(I-1)]$. The MLE of $\sigma^2$ is $\hat{\sigma}^2_{MLE} = SSE/n$. The maximum of the loglikelihood function is

$$\ell(\hat{\boldsymbol{\gamma}}_1, \cdots, \hat{\boldsymbol{\gamma}}_I, \hat{\boldsymbol{\delta}}, \hat{\sigma}^2_{MLE}) = \ell_P(\hat{\boldsymbol{\delta}}, \hat{\sigma}^2_{MLE}) = -\frac{n}{2}(1 + \log \frac{2\pi}{n}) - \frac{n}{2} \log(\hat{\sigma}^2_{MLE}). \tag{4.17}$$

By $\hat{\sigma}^2$ or $\hat{\sigma}^2_{MLE}$, we can obtain $\hat{\mathrm{V}}(\hat{\boldsymbol{\delta}})$, $\hat{\mathrm{V}}(\hat{\boldsymbol{\gamma}}_i)$, $\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\gamma}}_i)$, and $\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\gamma}}_{i'})$ $(i \neq i')$, the estimators of $\mathrm{V}(\hat{\boldsymbol{\delta}})$, $\mathrm{V}(\hat{\boldsymbol{\gamma}}_i)$, $\mathrm{Cov}(\hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\gamma}}_i)$, and $\mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\gamma}}_{i'})$ $(i \neq i')$, respectively. We use those to derive the estimators of $\boldsymbol{\gamma}_i$, $\boldsymbol{\delta}$, and $\sigma^2$ in (4.1), which can be modified to estimators of $\tilde{\boldsymbol{\gamma}}_i$ and $\sigma^2$ in (4.4). We can also derive the variance-covariance matrix of $\tilde{\boldsymbol{\gamma}}_i$ in (4.4) by letting $q_0 = q$. Then, we can formulate an $F$-test to assess (4.5).

## 4.2 Multiple Model Analysis

One of the most important tasks in our approach is to identify the optimal model, where we need to compare multiple models. The standard way is to study hypotheses testing of a full model versus a reduced model. Both are modified from (4.1). Let the full model be

$$y_{ij} = \mathbf{w}_{Fij}^\top \boldsymbol{\gamma}_{Fi} + \mathbf{z}_{Fij}^\top \boldsymbol{\delta}_F + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma_F^2), i = 1, \cdots, I, j = 1, \cdots, n_i, \tag{4.18}$$

where $\mathbf{w}_{Fij} = (1, x_{Fij1}, \cdots, x_{Fij(q_{F0}-1)})^\top$, $\mathbf{z}_{Fij} = (x_{Fijq_{F0}}, \cdots, x_{Fij(q_F-1)})^\top$, $\boldsymbol{\gamma}_{Fi} = (\gamma_{Fi0}, \cdots, \gamma_{Fi(q_0-1)})^\top$, and $\boldsymbol{\delta}_F = (\delta_{q_{F0}}, \cdots, \delta_{F(q_F-1)})^\top$. Let the reduced model be

$$y_{ij} = \mathbf{w}_{Rij}^\top \boldsymbol{\gamma}_{R,i} + \mathbf{z}_{Rij}^\top \boldsymbol{\delta}_R + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma_R^2), i = 1, \cdots, I, j = 1, \cdots, n_i, \tag{4.19}$$

where $\mathbf{w}_{Rij} = (1, x_{Rij1}, \cdots, x_{Rij(q_{R0}-1)})^\top$, $\mathbf{z}_{Rij} = (x_{Rijq_{R0}}, \cdots, x_{Rij(q_R-1)})^\top$, $\boldsymbol{\gamma}_{Ri} = (\gamma_{Ri0}, \cdots, \gamma_{Ri(q_0-1)})^\top$, $\boldsymbol{\delta}_R = (\delta_{q_{R0}}, \cdots, \delta_{R(q_R-1)})^\top$. If both $(\mathbf{w}_{F,ij}^\top, \mathbf{z}_{F,ij}^\top)^\top$ and $(\mathbf{w}_{R,ij}^\top, \mathbf{z}_{R,ij}^\top)^\top$ are subvectors of $\tilde{\mathbf{w}}_{ij}$ such that they can be treated as reduced models of (4.4), then both models can be fitted by $\mathcal{S}$.

If we choose $\mathbf{w}_{Fij} = \tilde{\mathbf{w}}_{ij}$ without $\mathbf{z}_{Fij}$ in (4.18) and $\mathbf{w}_{Rij} = \mathbf{w}_{ij}$ and $\mathbf{z}_{Rij} = \mathbf{z}_{ij}$ in (4.19), then (4.18) and (4.19) become (4.4) and (4.1), respectively. Therefore, we can test (4.5). If we choose $\mathbf{w}_{Fij} = \mathbf{w}_{Rij}$ and $\mathbf{z}_{Rij}$ as

a subvector of $\mathbf{z}_{Fij}$, then we can test the significance of the components of $\boldsymbol{\delta}_F$ which are not contained in $\boldsymbol{\delta}_R$. This is a test for main-effects. If we choose $\mathbf{w}_{Rij}$ as a subvector of $\mathbf{w}_{Fij}$, then we can also test the significance of interaction-effects.

Let $\hat{\sigma}_F^2$ and $\hat{\sigma}_R^2$ be the MSEs of (4.18) and (4.19), respectively. Then, $(n-p_F)\hat{\sigma}_F^2$ and $(n-p_R)\hat{\sigma}_R^2$ are the SSEs of (4.18) and (4.19), respectively, where $p_F = q_{F_0}(I-1)+q_F$ and $p_R = q_{R_0}(I-1)+q_R$ are their model degrees of freedom, respectively. By the standard way, we construct an F-statistic as

$$F^* = \frac{[(n-p_R)\hat{\sigma}_R^2 - (n-p_F)\hat{\sigma}_F^2]/(p_F-p_R)}{\hat{\sigma}_F^2}. \tag{4.20}$$

We use $F^*$ to test (4.19) under (4.18). It follows the $F_{p_F-p_R,n-p_F}$-distribution. Large values of $F^*$ lead to a rejection of (4.19).

An obvious advantage of our approach is that we only need to access the data once in the entire comparison. The goal of the access is the derivation of $\mathcal{S}$. For any $\mathbf{w}_{Fij}$ and $\mathbf{z}_{Fij}$ in (4.18) and $\mathbf{w}_{Rij}$ and $\mathbf{z}_{Rij}$ in (4.19), we can obtain exact values of $\hat{\boldsymbol{\gamma}}_{F,i}$, $\hat{\boldsymbol{\delta}}_F$, $\hat{\sigma}_F^2$, $\hat{\boldsymbol{\gamma}}_{R,i}$, $\hat{\boldsymbol{\delta}}_R$, and $\hat{\sigma}_R^2$ by $\mathcal{S}$ only, implying that (4.20) can be applied. As plenty of models can be specified by (4.18) and (4.19), we have obtained an extremely efficient way to study a group of regression models together. This is different from typical ways used by many software packages (e.g., R and SAS) as they always use the entire

data set to fit statistical models.

## 4.3 Relation to Traditional Implementation

It is important to study theoretical connections between our proposed approach and the traditional implementation of the least squares approach. We assume that the traditional implementation is carried out by a supercomputer such that (2.2) can always be applied even if $\mathbf{X}$ is very large. We want to show that results given by our approach are identical to those given by the traditional implementation.

We use $(\hat{\boldsymbol{\gamma}}_1, \cdots, \hat{\boldsymbol{\gamma}}_I, \hat{\boldsymbol{\delta}})$ and $\hat{\sigma}^2$ to denote those from our approach. We use $(\hat{\boldsymbol{\gamma}}_{T1}, \cdots, \hat{\boldsymbol{\gamma}}_{TI}, \hat{\boldsymbol{\delta}}_T)$ and $\hat{\sigma}_T^2$ to denote those from the traditional implementation, derived under (4.2) with $\mathbf{X}$ given by (4.3). We use $(\hat{\boldsymbol{\alpha}}_T, \hat{\boldsymbol{\omega}}_{T2}, \cdots, \hat{\boldsymbol{\omega}}_{TI}, \hat{\boldsymbol{\delta}}_T)$ to denote those from the traditional implementation under (2.5) with $\mathbf{X}$ given by (2.6). Then, $\hat{\boldsymbol{\alpha}}_T = \hat{\boldsymbol{\gamma}}_{T1}$ and $\hat{\boldsymbol{\omega}}_{Ti} = \hat{\boldsymbol{\gamma}}_{Ti} - \hat{\boldsymbol{\gamma}}_{T1}$ for $i = 2, \cdots, I$. The variance-covariance matrices can be formulated respectively. We use $F^*$ and $F_T^*$ to represent the test statistics in the comparison between (4.18) and (4.19) from our and the traditional approaches, respectively.

**Theorem 1.** *$\hat{\boldsymbol{\gamma}}_i = \hat{\boldsymbol{\gamma}}_{Ti}$ for $i = 1, \cdots, I$, $\hat{\boldsymbol{\delta}} = \hat{\boldsymbol{\delta}}_T$, and $\hat{\sigma}^2 = \hat{\sigma}_T^2$.*

**Proof:** By the fact that $\mathbf{X}^\top\mathbf{X}$, $\mathbf{X}^\top\mathbf{y}$, and $\mathbf{y}^\top\mathbf{y}$ given by (4.3) and (4.13) are identical, respectively, we can show the conclusion by simply studying

the formulations of (4.15) and (4.16) with those given by the traditional implementation. □

**Theorem 2.** $\mathrm{V}(\hat{\boldsymbol{\gamma}}_i) = \mathrm{V}(\hat{\boldsymbol{\gamma}}_{Ti})$, $\mathrm{V}(\hat{\boldsymbol{\delta}}_i) = \mathrm{V}(\hat{\boldsymbol{\delta}}_{Ti})$, $\mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\delta}}) = \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_{Ti}, \hat{\boldsymbol{\delta}}_T)$, *and* $\mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\gamma}}_{i'}) = \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_{Ti}, \hat{\boldsymbol{\gamma}}_{Ti'})$ *for distinct* $i, i' = 1, \cdots, I$.

**Proof:** We draw the conclusion by comparing each of them based on those given by Section 4.1 with those given by the traditional implementation. □

**Corollary 1.** $\hat{\boldsymbol{\alpha}}_T = \hat{\boldsymbol{\gamma}}_1$, $\hat{\boldsymbol{\omega}}_{Ti} = \hat{\boldsymbol{\gamma}}_i - \hat{\boldsymbol{\gamma}}_1$, $\mathrm{V}(\hat{\boldsymbol{\alpha}}_T) = \mathrm{V}(\hat{\boldsymbol{\gamma}}_1)$, $\mathrm{V}(\hat{\boldsymbol{\omega}}_{Ti}) = \mathrm{V}(\hat{\boldsymbol{\gamma}}_i) + \mathrm{V}(\hat{\boldsymbol{\gamma}}_1) - \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\gamma}}_1) - \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_i)$, $\mathrm{Cov}(\hat{\boldsymbol{\alpha}}_T, \hat{\boldsymbol{\omega}}_{Ti}) = \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_i)$, $\mathrm{Cov}(\hat{\boldsymbol{\omega}}_{Ti}, \hat{\boldsymbol{\omega}}_{Ti'}) = \mathrm{V}(\hat{\boldsymbol{\gamma}}_i) + \mathrm{V}(\hat{\boldsymbol{\gamma}}_{i'}) - \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_{i'}, \hat{\boldsymbol{\gamma}}_i) - \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\gamma}}_{i'})$, $\mathrm{Cov}(\hat{\boldsymbol{\alpha}}_T, \hat{\boldsymbol{\delta}}_T) = \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\delta}})$, *and* $\mathrm{Cov}(\hat{\boldsymbol{\omega}}_{Ti}, \hat{\boldsymbol{\delta}}_T) = \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_i, \hat{\boldsymbol{\delta}}) - \mathrm{Cov}(\hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\delta}})$, *for* $i, i' = 2, \cdots, I$ *with* $i \neq i'$.

**Proof:** The conclusion is directly implied by Theorems 1 and 2. □

**Corollary 2.** $F^* = F_T^*$.

**Proof:** The conclusion is directly implied by Theorem 1. □

We classify our approach as an exact approach, as its results are identical to those given by the traditional implementation. Since fitting procedures based on the traditional implementation may need as much as $O(nqI)$

memory size but fitting procedures based on index least squares formulations only needs $O(q^2I + qI + 1)$ memory size, we successfully avoid the difficulty caused by the memory barrier in traditional fitting procedures.

## 4.4 Extension

We have two methods to extend our approach to a model with at least two factor variables. The first, called the combined factor approach, uses a combined factor to represent all of the factors. The second, called the mixed index least squares and dummy variable approach or mixed approach for short, treats one factor variable as an index and all the rest by dummy variables. We introduce the two methods based on two factor variables cases below.

Let $A$ and $B$ be the two factor variables with $I$ and $J$ levels denoted by $A_1, \cdots, A_I$ and $B_1, \cdots, B_J$, respectively. Let $F$ be a combined factor for $A$ and $B$. Then, $F$ has $IJ$ levels which can be represented by $F_{ij} = (A_i, B_j)$ for $i = 1, \cdots, I$ and $j = 1, \cdots, J$. The observed data can be expressed as

$$\mathcal{D} = \{(y_{ijk}, \mathbf{x}_{ijk}^\top, A_i, B_j) : i = 1, \cdots, I, j = 1, \cdots, J, k = 1, \cdots, n_{ij}\}, \tag{4.21}$$

where $\mathbf{x}_{ijk} = (1, x_{ijk1}, \cdots, x_{ijk(q-1)})^\top$ represents the $(i, j, k)$th observed vector of explanatory variables.

In the combined factor approach, we express (4.21) as

$$\mathcal{D} = \{(y_{ik}, \mathbf{x}_{ik}^\top, F_i) : i = 1, \cdots, IJ, k = 1, \cdots, n_i\}. \tag{4.22}$$

Then, $F_i$ with $i = 1, \cdots, IJ$ can be used to represent all of the factor levels of $F$. By the same method given by Section 4.1, we can define $\mathcal{S}$ by (4.12). It contains at most $IJq^2$ real numbers. Note that $\mathcal{S}$ is an array of sufficient statistics. We can fit a model similar to (4.1) only by $\mathcal{S}$.

In the mixed approach, we only treat $A$ as an index variable, and account for $B$ by dummy variables. Let $b_{ijk}^{j'} = 1$ if the $(i, j, k)$th level of $B$ is $j'$ or $b_{ijk}^{j'} = 0$ otherwise. Then, we obtain $J$ dummy variables for $B$. Let $\tilde{\mathbf{x}}_{ijk} = (b_{ijk}^1 \mathbf{x}_{ijk}^\top, \cdots, b_{ijk}^J \mathbf{x}_{ijk}^\top)^\top$. Then, $\tilde{\mathbf{x}}_{ijk}$ is a $qJ$-dimensional vector. We equivalently express (4.21) as

$$\mathcal{D} = \{(y_{ij}, \tilde{\mathbf{x}}_{ij}^\top, A_i) : i = 1, \cdots, I, j = 1, \cdots, \sum_{j=1}^{J} n_{ij}\}. \tag{4.23}$$

We can also define an $\mathcal{S}$ similar to (4.12), where the size of $\mathcal{S}$ is at most $IJ^2q^2$.

In the case when both $I$ and $J$ are large, the combined factor approach can significantly reduce the memory needed in the computation. However, if $J$ is small, then the mixed approach can also be used. For instance, if $I = 1000$, $J = 50$, and $q = 100$, then the memory needed in the combined factor approach is about 0.5GB but the memory needed in the mixed approach

is over 23GB. If $I$ increases 10 times, but $J$ decreases 10 times, such that $I = 10^4$ and $J = 5$, then the memory needed in the combined factor approach is still about 0.5GB but the memory needed in the mixed approach is reduced to about 2.3GB. Thus, it is important to justify the impact of $I$ and $J$ in extensions of our approach.

## 5. Parallel Computation

It is important to migrate our proposed approaches to well-known parallel or cluster computation frameworks, such as MapReduce and Spark. In MapReduce or Spark, most of the information processing tasks have similar structures. The same computation is applied over a large number of records by many processors, after that, individual results are aggregated.

Suppose that the linear model given by (2.1) is studied by parallel computation. Assume that the entire data set is partitioned into $K$ subsets. Let $\hat{\boldsymbol{\beta}}_k$ for $k = 1, \cdots, K$ be the estimates of $\boldsymbol{\beta}$ derived from the $k$th subsets. The divide-and-conquer (Bentley, 1980) or divide-and-recombine (Guha, et. al, 2012) technique attempts to combine individual final results: the final estimator of $\boldsymbol{\beta}$ is computed by a weighted average of individual final results given by $\sum_{k=1}^{K} w_k \hat{\boldsymbol{\beta}}_k$, where $w_k$ satisfying $\sum_{k=1}^{K} w_k = 1$ and $0 < w_k < 1$ is the weight of the $k$th subset. A more efficient way is to combine inter-

mediate results (Zhang and Yang, 2017a,b). Let $\mathcal{S}_1, \cdots, \mathcal{S}_K$ be regression arrays of individual subsets. Then, $\mathcal{S} = \sum_{k=1}^{K} \mathcal{S}_k$ is the regression array of the entire data. One can use $\mathcal{S}$ to compute $\hat{\boldsymbol{\beta}}$, its variance-covariance matrix, and $\hat{\sigma}^2$ in the model.

We modify the idea to linear models with factor variables. The key is the derivation of the index array by parallel computation. Let $\mathcal{S}_k = (s_{k,yy}, \tilde{\mathbf{s}}_{k1,wy}, \cdots, \tilde{\mathbf{s}}_{kI,wy}, \tilde{\mathbf{S}}_{k1,ww}, \cdots, \tilde{\mathbf{S}}_{kI,ww})$ be the index array from the $k$th subset. Then, $\mathcal{S} = \sum_{k=1}^{K} \mathcal{S}_k$ is the index array of the entire data. Therefore, we propose our parallel computation below.

**Algorithm 2** Computation of $\mathcal{S}$ in Parallel Computation

**Input**: Subsets of a massive data set on a number of hard disks

**Output**: $\mathcal{S}$ for the entire data set

1: **procedure** Updating $\mathcal{S}_k$ by Rows Individually

2: Compute $s_{k,yy}$, $\tilde{\mathbf{s}}_{k1,wy}, \cdots, \tilde{\mathbf{s}}_{kI,wy}$, and $\tilde{\mathbf{S}}_{k1,ww}, \cdots, \tilde{\mathbf{S}}_{kI,ww}$ by Step 3 of Algorithm 1 for each $k$

3: Output $\mathcal{S} = \sum_{k=1}^{K} \mathcal{S}_k$

4: **end procedure**

The main task in parallel computation is the derivation of $\mathcal{S}$ for the entire data set. Once it is available, the task of parallel computation is over. The remainder of the computations can be completely carried out by meth-

ods in Sections 4.1 and 4.2. It is not necessary to use parallel computation if the size of the data is lower than the size of the hard disk of a personal computer, however, the parallel computation can significantly enhance the speed of the computation. If the data set is too large to be stored in a single hard disk, then parallel computation must be used. Therefore, the implementation of our approach relies on the size of hard disks but not the size of memory. Since the data set is only scanned once, our approach is extremely efficient in fitting linear models with factor variables in big data.

## 6. Simulation

We evaluated the computational advantage of our approach via simulated examples. All of the computations were carried out by a third generation Intel core-i7 2.8$GHz$ processor with 16GB DDR3 memory. The algorithms of our proposed approach were written into C++ and R codes. The C++ codes were used in all of the simulated examples, while the R codes were only used in the case when the size of the simulated data was lower than the memory size of the computing system. We focused on the evaluation of the performance of our approach based on a single processor. If a parallel algorithm is used, then its performance can be reflected by the performance of algorithms carried out by individual processors. Therefore, the evaluation

based on a single processor is fundamental in the understanding of our entire approach.

## 6.1 Comparison with Traditional Implementation

We compared results of our approach with those given by the traditional implementation, to determine whether their results were identical. To carry out the traditional implementation, we assumed that the observed data were only small or moderate. We used standard fitting procedures in R and SAS as well as our proposed approach in C++ and R to analyze the data.

We assumed that the data set contained one factor variable and six continuous explanatory variables, such that it was represented by $\mathcal{D} = \{(y_{ij}, \mathbf{x}_{ij}^\top, i) : i = 1, \cdots, I, j = 1, \cdots, n_i\}$, where $\mathbf{x}_{ij} = (1, x_{ij1}, \cdots, x_{ij6})^\top$ represented the $j$th vector of the explanatory variables at the $i$th level of the factor variable. We generated $(x_{ij1}, \cdots, x_{ij6})$ identically and independently from a mean zero six-dimensional multivariate normal distribution with all variances equal to 0.25 and all correlations equal to 0.5. We generated the response from

$$y_{ij} = \gamma_{i0} + \gamma_{i1}x_{ij1} + \gamma_{i2}x_{ij2} + \delta_3 x_{ij3} + \delta_4 x_{ij4} + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, 0.25^2), \quad (6.1)$$

for $i = 1, \cdots, I$ and $j = 1, \cdots, n_i$, where $\gamma_{i0}$ was independently generated from $N(2.5, 0.25^2)$, $\gamma_{i1}$ and $\gamma_{i2}$ were independently generated from

$N(0.5, 0.125^2)$, and $\delta_3 = \delta_4 = 0.5$. The model contained the main-effects of the index variable, the interaction-effects between the index variable and the first two explanatory variables, and the main-effects of the first four explanatory variables. Finally, we obtained a data set with $n = \sum_{i=1}^{I} n_i$ rows and 8 columns. We saved the data set to the hard disk of our computer. We used a varied $I$ to study its impact.

We studied three models. The first, called the interaction-effects model, contained all of the six interaction-effects between the index and explanatory variables. It was expressed as

$$y_{ij} = \gamma_{i0} + \gamma_{i1}x_{ij1} + \cdots + \gamma_{i6}x_{ij6} + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma^2). \tag{6.2}$$

The second model, called the main-effects model, only contained the main-effects of the index and explanatory variables. It was expressed as

$$y_{ij} = \gamma_{i0} + \delta_1 x_{ij1} + \cdots + \delta_6 x_{ij6} + \epsilon_{ij}, \epsilon_{ij} \sim^{iid} N(0, \sigma^2). \tag{6.3}$$

The third model was the true model given by (6.1).

Our proposed approach was carried out by our C++ and R code. The traditional implementation was carried by the standard packages given by the `lm` function in R and the `proc glm` procedure in SAS. We compared their computational time (Table 1). The time taken in our proposed approach based on our C++ codes for the three models together was all less than 1

Table 1: Time taken (seconds) in the traditional implementation carried out by `lm` in R and `proc glm` in SAS when $n_i = 100$ for all $i$, where $\times$ means out of memory. It includes the loading of data from the hard disk to memory (Load), the fitting of the interaction-effects (Inter), main-effects (Main), and true (True) models.

| | Size | `lm` in **R** | | | | `proc glm` in **SAS** | | | |
|---|---|---|---|---|---|---|---|---|---|
| $I$ | (MB) | Load | Inter | Main | True | Load | Inter | Main | True |
| 100 | 0.5 | 0.32 | 3.51 | 0.12 | 0.71 | 0.09 | 1.05 | 0.10 | 0.26 |
| 200 | 1.0 | 0.39 | 28.28 | 0.70 | 5.47 | 0.17 | 8.12 | 0.26 | 1.35 |
| 500 | 2.5 | 0.65 | 446.12 | 9.97 | 84.24 | 0.25 | 167.45 | 2.38 | 23.25 |
| 1,000 | 5.1 | 1.06 | 3628.42 | 78.33 | 683.29 | 0.27 | 1367.27 | 16.76 | 228.78 |
| 2,000 | 10.2 | 1.36 | $\times$ | 635.27 | $\times$ | 0.27 | $\times$ | 187.52 | 1873.29 |

second (C++ showed that it was 0 seconds). The time taken based on our R code when $I = 2{,}000$ was a slightly lower than 2.5 seconds. The time taken in the rest of the cases were all lower than 1.2 seconds. The time taken in the traditional implementation might be as high as an hour. The algorithms carried out by SAS were generally four times faster than those carried out by R.

We examined the EMA algorithm by SAS. At the beginning, SAS opened a temporary file in the hard disk. It was about two to three times as large as the original data file. When a fitting procedure was used by `proc glm`, the temporary file grew dramatically to over a thousand times as large as the original data file. It was reduced to two to three times again after the fitting procedure was over. The `proc glm` stored the information used by the computation to the temporary file. The computation was still based on the traditional implementation.

We also compared numerical results. We found that all of them were identical when they were available, as expected by Corollary 1. We studied two scenarios in the implementation of $F^*$ given by Section 4.2. In the first, we used $F^*$ to assess (6.1) under (6.2). In the second, we used $F^*$ to assess (6.3) under (6.2). We treated (6.2) as the full model and (6.1) or (6.3) as a reduced model. We found that the results were also identical,

as expected by Corollary 2. Therefore, we concluded that the index least squares approach provided exact solutions to big data regression with factor variables.

## 6.2 Comparison with Sparse Matrix Approach

Since the design matrix was sparse, we compared our approach with the sparse matrix approach given by the `proc hpmixed` procedure in SAS (Table 2). The `proc hpmixed` procedure is developed for the linear mixed-effects model, but it can also be used to analyze fixed-effects models if the random-effects component is not specified. The `proc hpmixed` procedure can handle factor variables with large numbers of factor levels. We generated data from (6.1) with all $n_i = 10^4$ for selected $I$, and studied the fitting of the three models. After the data was loaded, the `proc hpmixed` procedure was found to be able to only be used individually, and accessed the data set three times, while our approach was able to be used simultaneously, with the `C++` code only scanning the entire data set once. In our R codes, we loaded the data from the hard disk to the memory at the beginning. Then, we calculated the index array in memory. After that, we removed the data from the memory. The rest of the computations were completely carried out based on the index array. Since the size of the index

Table 2: Time taken (in minutes) in the sparse matrix approach carried out by `proc hpmixed` in SAS when $n_i = 10^4$ for all $i$ and the proposed index least square approach by C++ and R, where $\times$ means out of memory. The results of `proc hpmixed` include the loading of data (Load) and the fitting of the interaction-effects (Inter), main-effects (Main), and true (True) models. The results of our approach include scanning data by rows (Scan) in our C++ code, loading the data from the hard disk to memory (Load) and scanning data by rows in memory (Scan) in our R code, and the fitting of all the three models together (Fitting).

| | | | | | | Index Least Squares | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Size | `proc hpmixed` in **SAS** | | | | **C++** | | **R** | | |
| $I$ | (GB) | Load | Inter | Main | True | Scan | Fitting | Load | Scan | Fitting |
| 1,000 | 0.5 | 0.12 | 1.88 | 0.66 | 0.80 | 1.15 | 0.00 | 0.85 | 0.37 | 0.02 |
| 2,000 | 1.0 | 0.29 | 4.05 | 1.40 | 1.42 | 2.32 | 0.00 | 1.71 | 1.23 | 0.04 |
| 5,000 | 2.5 | 0.83 | 15.85 | 4.22 | 6.13 | 5.82 | 0.01 | 4.17 | 6.78 | 0.09 |
| 10,000 | 5.0 | 2.21 | 39.53 | 12.52 | 17.04 | 11.67 | 0.02 | 8.58 | 26.34 | 0.18 |
| 20,000 | 10.2 | 4.53 | 92.38 | 34.78 | 42.20 | 23.77 | 0.04 | $\times$ | $\times$ | $\times$ |
| 50,000 | 26.6 | 13.69 | $\times$ | 96.63 | 125.75 | 61.82 | 0.09 | $\times$ | $\times$ | $\times$ |

array was not large, simultaneous fitting of the three models was extremely efficient. We examined the EMA algorithm carried out by SAS. A temporary file was created in the hard disk by the `proc hpmixed`. The size of the temporary file was about four times as large as the size of the observed data set. The `proc hpmixed` procedure was able to handle a data set higher than the memory size of the computer (i.e., $I = 50{,}000$), but it still had an out-of-memory problem in fitting the interaction-effects model. It could not provide significance of the interaction-effects when $I = 50{,}000$.

### 6.3 Implementation to Big Data Regression

We generated data from (6.1) with $I = 5 \times 10^5$ and all $n_i = 10^4$. The data set contained $5 \times 10^9$ rows and 8 columns. The size was about 260.8GB. Using our C++ code, we calculated $\mathcal{S}$ given by (4.13), by scanning the entire data set on the hard disk, which took about 618.1 minutes. After that, we fitted the interaction-effects model given by (6.2), the main-effects model given by (6.3), and the true model given by (6.1) by the index least squares approach. Their R-square values were 0.929232, 0.920363, and 0.92918, respectively. Their MSE ($\hat{\sigma}^2$) values were 0.0624971, 0.0703017, and 0.0624971, respectively. We calculated $\hat{\boldsymbol{\gamma}}_i$ and $\hat{\boldsymbol{\delta}}$ in all of the three models. In (6.1), we obtained $\hat{\delta}_3 = 0.500004$ and $\hat{\delta}_4 = 0.500246$ with

standard errors $s(\hat{\delta}_3) = 9.82(10^{-6})$ and $s(\hat{\delta}_4) = 9.82(10^{-6})$, respectively, which were close to their true values. In testing (6.3) under (6.2), we obtained $F^* = 209.3$ with $p$-value 0, indicating that at least one interaction-effect was significant. In testing (6.1) under (6.2), we obtained $F^* = 1.00044$ with $p$-value 0.3299, indicating that the interaction-effects model could be reduced to the true model. We also studied many other models. Our results showed that only effects contained in the true model were significant, implying that the true model was the optimal model. After the index array was derived, the fitting of all of the statistical models was very fast (C++ showed that it was less than 0.1 minutes for all of them together). Therefore, our approach was extremely efficient after the index array of sufficient statistics was derived.

## 7. Application

We applied our approach to the airline data set. It can be freely downloaded from the ASA (American Statistical Association) website. The airline data contained flight delay information from 1987 to 2008, over hundreds of airports in the United States. The entire data set was over 40GB. We initially analyzed data of individual years separately by the standard R package. We fitted a few basic regression models for flight delays, but we

could not put the airport code into any models that we wanted to study. The primary reason was that the airport code was a factor variable with over hundreds of levels. The standard fitting procedure in R needed to define hundreds of dummy variables, leading to an algorithm with over a few hundred GB in the computation.

We wanted to use the indexed least squares approach to overcome the difficulty. As a few important variables were lost in earlier years, we decided to analyze the data from 1995 to 2008. We chose minutes of flight delay as the response variable and departure airport code as the index variable. We had additional seven continuous explanatory variables. They were *actual elapsed time*, *CRS elapsed time*, *air time*, *arrival delay*, *distance*, *taxi in*, and *taxi out*. After cleaning missing variables and airports with only limited numbers of flights, the final data set contained about 73 million rows and 29 columns, where the variable for airport codes had 287 airports. We used 1 to 287 to denote these airports.

We used the information of the response, the airport code, and the seven continuous explanatory variables in our approach. The time taken of derivation of the index array given by scanning data by rows was 51.36 minutes. After $\mathcal{S}$ given by (4.13) was available, we fitted various models, including the model with only the main-effects of all of the explanatory

Table 3: Type III sum of squares of the interaction-effect between the airport code and the seven continuous variables under the interaction-effect model, respectively, where the MSE of the model is 49.14.

| Variable in the interaction effect | DF | Mean of Squares | $F$-value | $p$-value |
|---|---|---|---|---|
| actual elapsed time | 286 | 10,403 | 211.7 | 0 |
| CRS elapsed time | 286 | 6,652 | 135.4 | 0 |
| air time | 286 | 10,229 | 208.2 | 0 |
| arrival delay | 286 | 2,599 | 52.9 | 0 |
| distance | 286 | 3,287 | 66.9 | 0 |
| taxi in | 286 | 13,057 | 265.7 | 0 |
| taxi out | 286 | 11,482 | 233.7 | 0 |

and the factor variables (i.e., the main-effects model), and the model with all of the interaction-effects between the factor and the seven continuous explanatory variables (i.e., the interaction-effects model). The entire computation of all of the models under $\mathcal{S}$ only took 2.44 seconds. We calculated type III means of squares of the interaction-effects between the factor and the seven continuous variables (Table 3). Since all of them were significant, we concluded that the interaction-effects model was the most appropriate model. We also calculated the R-squares of all of the models. Among those, the R-squares of the interaction-effects and the main-effects models were 0.94817 and 0.94802, respectively. Their root MSEs were 7.01007 and 7.01958, respectively. After that, we fitted the model with the seven continuous variables only. We calculated the regression array of sufficient statistics, which was obtained by ignoring the airport code in the index array of sufficient statistics. Then, we computed the estimates of regression coefficients. We obtained its R-square and root MSE values, which were 0.94800 and 7.02083, respectively. The computation only took 0.09 seconds.

As the interaction-effects model was the most appropriate model, we investigated its properties. We studied estimates of coefficients of the seven continuous variables. We found that they varied significantly among air-

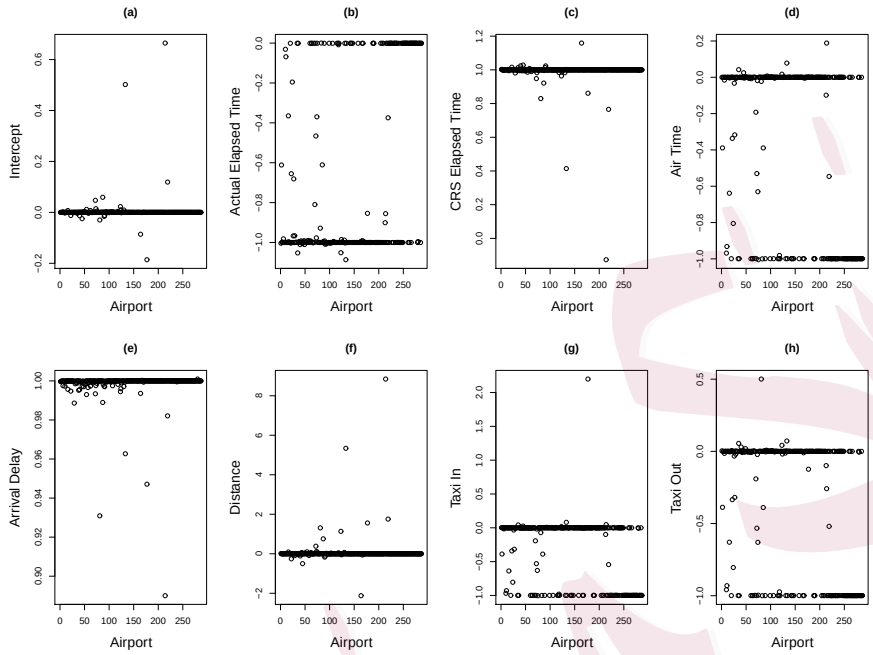


Figure 1: Estimates of coefficients with respect to airports.

ports (Figure 1). An interesting issue was that the estimates of coefficients of *actual elapsed time*, *air time*, *taxi in*, and *taxi out* with respect to airport codes could be basically partitioned into two groups. They were close to 0 in one group or close to $-1$ in another group. There were only a few airports outside of the two groups. The airports contained by the two groups were highly consistent according to results given by the four variables. We studied this issue and found that they could be basically interpreted as a large airport group and a small airport group, respectively. The estimates of coefficients of *air time*, *taxi in*, and *taxi out* with respect to airport codes

in the large airport group were almost equal to 0, while those in the small airport groups were also most equal to $-1$. The estimates of coefficients of *actual elapsed time* with respect to airport codes in the large airport group were almost equal to $-1$, while those in the small airport group were also most equal to 0. Based on our findings, we restricted the interaction-effects model to the two groups, respectively. Although they were still significant, the F-values of the interaction-effects between the factor variable and the seven continuous variables were significantly reduced. For instance, the F-values of the interaction-effects related to *actual elapsed time*, *air time*, *taxi in*, and *taxi out* in the large airport group were 5.99, 2.83, 2.00, and 5.23, respectively. Comparing those with Table 3, we concluded that the performance of the continuous variables on flight delay highly depended on sizes of airports.

To compare, we also investigated two previous approaches: the sampling, and the divide-and-recombine approaches. The sampling approach attempts to sample a small number of observations from the data. Statistical models are only fitted based on the sampled data. The procedure is often repeated many times to make the results reliable. We found two difficulties in the implementation of the sampling approach. One difficulty was that the number of flights among airports were extremely unbalanced.

For instance, each of the largest five airports (Chicago ORD, Atlanta ATL, Dallas/Fort DFW, Los Angeles LAX, and Phoenix PHX) had over 2.47 million flights in the entire period, but each of the smallest five airports (Muskegon County MKG, Rhinelander/Oneida County RHI, Crater Lake-Klamath Regional LMT, Southwest Oregon Regional OTH, and Houghton County CMX) had less than 600 flights. It was hard to make the sample to contain all of the airports. Another difficulty was caused by the airport codes contained in the sampled data, which were mostly inconsistent among replications, and made it difficult to summarize the results. The divide-and-recombine approach attempts to partition a massive data set into many small or moderate subsets. After that, parallel computation is applied. Each subset is independently analyzed by an individual process with no communications between processes, and outputs of subsets are recombined at the end. We identified two difficulties in the implementation of the divide-and-recombine approach. The first was caused by airports. We found that airport codes in data sets for individual calendar years were not consistent. Many airports appeared in one calendar year but not in another, which made it hard to keep airport codes consistent among subsets. The second was caused by the memory needed in the analysis of partitioned subsets. Because of the factor variable, the memory needed in the computa-

tion of individual processes could still be large, even if the size of the subset was not. It was hard to determine the size of subsets to be partitioned. In the comparison, we found that our approach successfully overcame the difficulties in the sampling and the divide-and-recombine approaches.

## 8. Discussion

In this article, we propose the index least squares approach to linear regression with factor variables for big data. It successfully avoids the memory barrier caused by factor variables. If factor variables are involved, the main difficulty in the traditional implementation is the inflation of the size of the design matrix given by the dummy variable approach. This is not important if the size of data is small, but it is a serious concern in big data regression.

We have several findings. First, the concept of sufficient statistics is important in classical statistical theories, but it has not been paid much attention to in statistical applications. In most popular statistical packages, such as R and SAS, the first step is always the loading of the entire data set to memory. If the first step fails, then it is impossible to do any further analysis. If sufficient statistics are available, then we can also provide precise solutions to statistical models. Therefore, it is not necessary to

load the entire data set to memory. Second, as the access of entire data set on hard disks is time-consuming, the consideration of sufficient statistics must be based on multiple models instead of a specific model. We should modify the traditional concept of sufficient statistics such that the modified version can be used to a group of statistical models together. This kind of sufficient statistics can provide exact solutions to a group of models simultaneously. Third, the identification of optimal models is important not only in traditional statistics, but also in big data regression. To identify an optimal model, one must study many candidate models. If the size of data is small or moderate, it is enough to evaluate a fitting procedure based on individual models. However, if the size of the data is big, then the evaluation of a fitting procedure based on a group of models is more appropriate. Therefore, new criteria are needed.

Although only regression problems are studied, it does not mean that linear regression can always capture the complicated relationship between the response and explanatory variables. We only treat linear regression as the first step in the development of statistical approaches to big data. Other methods beyond linear regression are also important. An obvious example is the case when the response is count, where generalized linear models (GLMs) are often used. Since the standard fitting procedure for GLMs

is the iterative reweighted least squares (IRWLS) algorithm, it is likely to extend our idea to the IRWLS for GLMs with factor variables. In addition, if a semiparametric model is studied and its parametric component has factor variables, then we may also define an index variable to reduce the memory needed in the computation. Note that there are plenty of statistical approaches to small or moderate data. It is important to investigate all of them under the framework of big data. The idea of the article provides a way to study these problems.

Basically, statistical approaches can be as important as computer science approaches. It is well-known that parallel or cluster computation is powerful in the analysis of big data, but statistical approaches and algorithms used in the parallel or cluster computation can significantly affect its efficiency and feasibility. Previous approaches under many popular parallel or cluster computation frameworks recommend combining final results obtained from individual subsets. We point out that the combination of intermediate results can be more efficient and precise. Therefore, the problem about what and where to be combined must be investigated. Statistical approaches are important in the development of the corresponding methodology. We believe that this should be an important future research topic in big data analysis.

## Acknowledgements

The authors appreciate comments from an associate editor and three anonymous referees that significantly improved the quality of the article.

## References

Battey, H., Fan, J., Liu, H., Lu, J. and Zhu, Z. (2018). Distributed testing and estimation under sparse high-dimensional models. *Ann. Stat.*, **46**, 1352-1382.

Bentley, J.L. (1980). Multidimensional divide-and-conquer. *Commun. ACM*, **23**, 214-229.

Chen, Y. and Dong, G. (2006). Regression cubes with lossless compression and aggregation. *IEEE Trans. Knowl. Data Eng.*, **18**, 1585-1599.

Dean, J. and Ghamawat, S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM*, **51**, 137-150.

Dhillon, P.S., Lu, Y., Foster, D. and Ungar, L. (2013). New subsampling algorithms for fast least squares regression. *Adv. Neural Inf. Process. Syst.*, **26**, 360-368.

Emerson, J.W. and Kane, M.J. (2012). Don't drown in the data. *Significance*, **9**, 38-39.

Enea, M. (2009). Fitting linear models and generalized linear models with large data sets in R. *Statistical Methods for the Analysis of Large Datasets: book of short papers*, 411-414.

Fang, H., Yang, B. and Zhang, T. (2017). Finding the best Box-Cox transformation from massive datasets on Spark. *2017 IEEE International Conference on Big Data(Big Data)*, 2951-2960, DOI: 10.1109/BigData.2017.8258263

# REFERENCES

Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M., Bentez, J.M. and Herrera, F. (2014). Big data with cloud computing: an insight on the computing environment, MapReduce, and programming frameworks. *WIREs Data Mining Knowledge Discovery*, **4**, 380-409.

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B. and Cleveland, W.S. (2012). Large complex data: divide and recombine (D&R) with Rhipe. *Stat*, **1**, 53-67.

Klotz, J.H. (1995). Updating simple linear regression. *Statistica Sinica*, **5**, 399-403.

Lin, N. and Xi, R. (2011). Aggregated estimating equation estimation. *Statist. Its Interface*, **4**, 73-83.

Ma, P, and Sun, X. (2015). Leveraging for big data regression. *WIREs Computational Statistics*, **7**, 70-76.

Meeker, W.Q. and Hong, Y. (2014). Reliability meets big data: opportunities and challenges. *Qualify Engineering*, **26**, 102-116.

Miner, D. and Shook, A. (2012). *MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems. O'Reilly Media Inc*, Sebastpool, California.

Pinar, A. and Heath, M.T. (1999). Improving performance of sparse matrix-vector multiplication. *1999 ACM/IEEE Conference on Supercomputing*, Article 30.

Schifano, E.D., Wu, J., Wang, C., Yan, J.,and Chen, M. (2016). Online updating of statistical inference in the big data setting. *Technometrics*, **58**, 393-403.

# REFERENCES

Vitter, J.S. (2008). *Algorithms and Data Structures for External Memory*. Now Publication Inc, Hanover, MA, USA.

Wang, H., Yang, M., and Stufken, J. (2018). Information-based optimal subdata selection for big data regression. *J. Am. Stat. Assoc*, to appear, DOI: 10.1080/01621459.2017.1408468.

Yang, B. and Zhang, T. (2016a). A scalable meta-model for big data security analyses. *IEEE Big Data Security on Cloud (BigDataSecurity)*, 55-60, DOI: 10.1109/BigDataSecurity-HPSC-IDS.2016.71

Yang, B. and Zhang, T. (2016b) A scalable feature selection and model updating approach for big data machine learning. *IEEE International Conference on Smart Cloud (SmartCloud)*, 146-151, DOI: 10.1109/SmartCloud.2016.32

Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I. (2010). Spark: cluster computing with working sets. *Proceedings of the second USENIX Conference on Hot Topics in Cloud Computing*, 2010.

Zhang, T. and Yang, B.(2016). Big data dimension reduction using PCA. *IEEE International Conference on Smart Cloud (SmartCloud),* 152-157, DOI: 10.1109/SmartCloud.2016.33

Zhang, T. and Yang, B. (2017a). Box-Cox transformation in big data. *Technometrics*, **59**, 189-201.

Zhang, T. and Yang, B. (2017b). An exact approach to ridge regression for big data. *Comput. Stat.*, **32**, 909-928.

## REFERENCES

Zhang, T. and Yang, B. (2018). Dimension reduction for big data. *Stat. Its Interface*, **11**, 295-306.

Department of Statistics, Purdue University, 250 North University Street, West Lafayette, IN 47907-2066, USA

E-mail: tlzhang@purdue.edu

Department of Computer and Information Technology, Purdue University, 401 North Grant Street, West Lafayette, IN 47907, USA

E-mail: byang@purdue.edu