**Transforming Freight Flow Data Collection**

Final Report

Proposed Sampling Approach and Results

Saeed Ghanbartehrani, Sara Akbar Ghanadian, Mohammad Al Adailah, Hoda Rahmani

Ohio University

Athens, OH

Jan 2020

Contents

[1 Introduction 5](#_Toc29161445)

[1.1 Sampling 5](#_Toc29161446)

[1.1.1 Simple Random Sampling 5](#_Toc29161447)

[1.1.2 Stratified Sampling 5](#_Toc29161448)

[1.1.3 Cluster Sampling 6](#_Toc29161449)

[1.1.4 Adaptive Sampling 7](#_Toc29161450)

[1.2 Optimal Stratification 7](#_Toc29161451)

[1.2.1 Auxiliary Variables 7](#_Toc29161452)

[1.2.2 Atomic Strata 8](#_Toc29161453)

[1.2.3 Objective Function 8](#_Toc29161454)

[1.2.4 Optimal Solution 9](#_Toc29161455)

[2 Related Literature 9](#_Toc29161456)

[2.1 Optimal Stratification 9](#_Toc29161457)

[2.2 Optimal Allocation 13](#_Toc29161458)

[2.3 Joint Stratification-Allocation 14](#_Toc29161459)

[2.4 Genetic Algorithm 14](#_Toc29161460)

[2.5 Simulated Annealing 16](#_Toc29161461)

[3 Proposed Methodology 17](#_Toc29161462)

[3.1 Current CFS Sample Design 18](#_Toc29161463)

[3.2 Candidate Methods 19](#_Toc29161464)

[3.2.1 Comparison of the Candidate Methodologies 21](#_Toc29161465)

[4 Case Study 22](#_Toc29161466)

[4.1 Data Sources 23](#_Toc29161467)

[4.2 Pre-processing of the Data 23](#_Toc29161468)

[4.3 Frame Data Generation 24](#_Toc29161469)

[4.4 Method 1: Optimal Stratification and Allocation Based on Genetic Algorithm 26](#_Toc29161470)

[4.4.1 Method 1 Procedure 27](#_Toc29161471)

[4.4.2 Method 1 Results 34](#_Toc29161472)

[4.5 Method 2: Optimal Stratification and Allocation Using Genetic Algorithm and Simulated Annealing 40](#_Toc29161473)

[4.5.1 Method 2 Procedure 41](#_Toc29161474)

[4.5.2 Method 2 Results 49](#_Toc29161475)

[4.6 Comparison of GA and GA/SA Methods 53](#_Toc29161476)

[4.7 Method 3: Generalized Lavallée-Hidiroglou Method for Strata Construction 54](#_Toc29161477)

[4.7.1 Method 3 results 57](#_Toc29161478)

[4.7.2 Comparing Results from Method 2 (SA\_GA) and Generalized LH Method 59](#_Toc29161479)

[5 Appendix 61](#_Toc29161480)

[5.1 Method 1 Scripts 61](#_Toc29161481)

[5.2 Method 1 Summary of the Results With Random Initial Solution (no K-means) 63](#_Toc29161482)

[5.3 Method 1 Summary of Results With K-means Initial Solution 65](#_Toc29161483)

[5.4 Method 2 Scripts 69](#_Toc29161484)

[5.5 Method 2 Results Based on Sample Size From GA Without K-means Initial Solution 73](#_Toc29161485)

[5.6 Method 2 Results Based on GA With Random Initial Solution (No K-means) 83](#_Toc29161486)

[5.7 Method 2 Results Based on Sample Size Vector From GA With K-means Initial Solution 95](#_Toc29161487)

[5.8 Method 2 Results Based on Total Sample Size From GA with K-means Initial Solution 99](#_Toc29161488)

[5.9 Method 3 Scripts 105](#_Toc29161489)

[5.10 Method 3 Results 106](#_Toc29161490)

[5.11 Related formulae 109](#_Toc29161491)

[6 References 111](#_Toc29161492)

# Introduction

## Sampling

The term ‘sampling’ has been used to refer to situations in which a fraction of a population is taken to be representative of the entire population. In other words, sampling is the process of observing selected members to approximate characteristics of the whole population from which they are drawn. By the same token, sampling design is defined as the methodology by which the sample units are chosen (Thompson, 2012). Some common sampling methods are simple random sampling, stratified sampling, and cluster sampling. Amongst those techniques, stratified sampling is the primary focus of this document.

### Simple Random Sampling

Simple random sampling (SRS) is a method in which a given number of distinct units are selected from a population so that each unit in the population is equally likely to be selected (Thompson, 2012).

### Stratified Sampling

Stratified sampling is a strategy in which the population is divided into several non-overlapping sub groups referred to as stratum from which the samples are selected (Al-Kateb & Lee, 2014). The main objective of this kind of sampling design is to acquire a sample with the desired level of accuracy while decreasing the sampling error. Stratified sampling can also be used to obtain a smaller sample with maintaining the desired level of accuracy. The variable that the study seeks to measure is assumed to be homogeneous among different strata in stratified sampling (Manly, 2004). Despite the fact that the variable of interest is anticipated to be similar, there might be some disparities across different strata. These potential inconsistencies are rooted in the nature of the sampling design since samples are selected independently in each stratum. To find the variances of estimators for the whole population, the variances of estimators for each stratum can be calculated and added up (Thompson, 2012).

There are two questions that need to be answered in a stratified sampling design (Basoglu, 2014): (i) How the population should be partitioned into a specific number of strata (stratification), (ii) How the sample size for each strata is determined (allocation).

There are two approaches to stratification and allocation in the literature. The first approach solves the problems of stratification and allocation in two separate phases. Stratification can be performed based on various variance reduction techniques such as Naive Monte Carlo Simulation, importance sampling, and stratified sampling, which will be discussed in ‎4.1. When the strata are created, the allocation can be determined using an allocation fractions technique. Allocation techniques are reviewed in ‎4.2.

The second approach employs techniques to solve the problem of stratification and allocation in one phase as a joint stratification-allocation method. Joint stratification-allocation techniques are reviewed in ‎4.3.

### Cluster Sampling

Cluster sampling is a sampling method in which the population is divided into separate groups known as clusters and then a simple random sample of clusters are selected from the population. When a cluster is selected, all units in the cluster are included in the sample.

### Adaptive Sampling

One significant concept in sampling design is the procedure that alters as the experiment progresses, referred to as ‘adaptive sampling’. Adaptive sampling enables the model to learn from the gathered data during the survey and to be modified accordingly (Lermusiaux, 2007). The main advantage of adaptive designs compared to non-adaptive design of the same size is the improve in precision and efficiency by taking advantage of the observed characteristics of the population (Thompson, 2012). As a result, adaptive sampling approach can contribute to a more comprehensive understanding of the target population.

One of the most remarkable current discussions in adaptive design is adaptive stratification. This strategy seeks to find the strata in which the variability in the variables of interest is abundant. Hence, by collecting more samples from such subgroups in which the variations are inconsistent with the rest of the subgroups (i.e. have significantly larger variability compared to the others), one can have samples that better reflect the larger population (Carpentier & Munos, 2013).

## Optimal Stratification

### Auxiliary Variables

The main objective in sampling is to gain information (i.e. estimate) regarding variable or variables of interests. Auxiliary variables are variables that can help provide superior estimates of the variable(s) of interest. Therefore, auxiliary variables are variables which their values are available for every unit of population, however, they are not among the variables of interest (“Auxiliary Variable,” 2008). For example, in a survey on the annual income of people in a country, age, gender, and education can be considered auxiliary variables.

### Atomic Strata

In stratified sample designs, the stratification is performed based on the values of auxiliary variables. In order to do that, the auxiliary variables need to be either categorical or converted to categorical variables. Variables such as education (degree), ethnicity, and gender are categorical, while variables such as weight or height can be converted to categorical variables by breaking down their domain of values to several intervals (50-60, 60-70, etc.).

Atomic strata are the strata constructed by the cartesian product of the values of all auxiliary variables. In other words, atomic strata are the strata formed by combining all possible values of the auxiliary variables. For example, if variable X1 takes values of 1, 2, and 3 and variable X2 takes values of 4, 5, and 6, the atomic strata formed by X1 and X2 variable will be as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | X1 | X2 | Stratum |
| 1 | 1 | 4 | 1\*4 |
| 2 | 1 | 5 | 1\*5 |
| 3 | 1 | 6 | 1\*6 |
| 4 | 2 | 4 | 2\*4 |
| 5 | 2 | 5 | 2\*5 |
| 6 | 2 | 6 | 2\*6 |
| 7 | 3 | 4 | 3\*4 |
| 8 | 3 | 5 | 3\*5 |
| 9 | 3 | 6 | 3\*6 |

### Objective Function

In mathematical programming, optimization is a technique that helps finding the best alternative (i.e. course of action) from a set of available alternatives. A mathematical model is an abstract representation of an existing problem using variables and mathematical and logical operators. A mathematical model typically includes one or more objectives functions as swell as a set of constraints. The objective function is a function that defines what needs to be optimized while the constraints define how the variables in the model can change. An objective function is typically formulated as maximizing profit (revenue) or minimizing loss (cost).

Below is an example of a mathematical model aiming at maximizing the volume of a cylindrical container (can) made from a L by W sheet of material.

Objective function

Max Z = r2.π.h

Subject to (constraints)

2.π.r (h + r) ≤ L.W

r,h >0

It is worth mentioning that the model presented above assumes that no material is wasted in the cutting process for simplicity.

### Optimal Solution

Optimal solution is referred to the specific values for the variables in a mathematical model that optimize the value of the objective function while satisfying all constraints.

When the mathematical model in the previous section is solved for a letter sized sheet of paper (8.5” by 11”) the optimal values for r (radius) and h (height) of the cylinder container would be 2.23” and 4.45” respectively resulting in a 69.41 in3 container.

# Related Literature

In this section, related literature in the areas of stratification, allocation, and joint stratification-allocation is reviewed.

## Optimal Stratification

A number of techniques have been developed to determine the optimal number of strata along with an efficient allocation approach. Carpentier and Munos (2013) proposed an algorithm referred to as Monte-Carlo Upper Lower Confidence Band (MC-ULCB) which adapts the Monte-Carlo integration algorithm in the stratification methodology. The objective function of this algorithm is based on a ‘noisy function’ which refers to a function involving a random component (i.e. noise). Therefore, the adaptive strategy aims at discovering the areas where the variation of the noisy function is bountiful besides finding the best partitioning method for the sampling procedure. Once the samples are extracted, the algorithm proceeds to get some parameters related to the upper and lower confidence bounds on the variability of the noisy function in each stratum since the function is assumed to be bounded (Carpentier & Munos, 2013). The variability of the variable of interest in each partition is measured in the form of standard deviation. Following this process, the samples are allocated to each stratum proportional to the variability of each stratum multiplied by its upper confidence band. In this study, the authors concluded that the proposed model can obtain more homogenous samples from each stratum as well as maintaining the number of strata it contains as small as possible (Carpentier & Munos, 2013).

In another study, Glasserman et al. (1999) proposed a procedure aiming to reduce the variability in Monte-Carlo simulation using stratification technique. In this scheme, the authors developed a methodology in which some samples are taken from a population that is considered to have a standard normal distribution. In this reserach, the Monte-Carlo technique is performed as well as stratification with a vector along different directions which are defined by the conditional covariance matrix (Glasserman et al., 1999). In order to stratify in different dimensions of the matrix, each dimension is split into multiple intervals with equal width. After obtaining the intervals, a point is uniformly selected from each interval. Finally, samples are selected by calculating the inverse of the cumulative normal distribution for each dimension (Glasserman et al., 1999).

In a study conducted by Yong, et al. (2016), the authors attempted to establish an optimal stratification method in the field of medical science using the stratum specific mean. The researchers highlight the importance of employing the baseline information of previous patints in order to predict a variable of interest which is the probability of subject’s response to a certain medicine or procedure. To begin this process, a dataset consisting of the subjects’ predicted values is created. This dataset can provide the researchers with a scoring system. Once the dataset is generated, a regression model is used to associate predicted values to their corresponding actual results. Stratification process is incorporated into this approach to predict future outcomes for the patients. To put it simply, scores are classified into several strata and the average score is calculated as a mean for anticipating the results of future individuals. The authors propose a scheme that can find the best stratifying method which minimizes the prediction error using a loss function.

Etore and Jourdain (2010) presented an adaptive stratified sampling algorithm in which the randomly selected samples from strata converge to an optimal allocation. In their study, the authors claim that the Monte-Carlo estimator in stratified sampling can be used to find the expectation of a target function (of the drawings). It is assumed in the model that the probability of each random variable under study is known and follows the normal distribution. As a result, the expectation of the target function can be computed through multiplying the expectation of each stratum by its probability. In fact, the number of total samples drawn from all strata and the proportion of extracted samples from each stratum are formulated in terms of the expectation of interest. In the same way, the variance can be calculated given the conditional expectations. This framework can contribute to variance reduction if the proportion of samples taken from each stratum is computed appropriately.

Basoglu (2014) applied variance reduction techniques to reduce the size of confidence intervals generated by Monte Carlo simulation used in computing financial risk involved in realistic and complex portfolio models. In this study, an efficient implementation of stratified sampling technique for Monte Carlo simulation problems referred to as Optimal Allocation Stratification and Importance Sampling (OASIS) is proposed. The proposed approach involves an efficient simulation algorithm that combines optimal stratification and importance sampling to estimate multiple conditional loss and gain probabilities for asset portfolios.

Two classes of objective functions are proposed to represent the overall error. The first class of error function minimizes a linear function of the variance-covariance matrix of the stratified estimates. The second class minimizes the maximum of variances weighted with non-negative coefficients. Both objective functions are used in nonlinear optimization models with allocation fractions as decision variables. A closed-form solution is developed for the first class of objective functions. For the second class, an optimal allocation heuristic is utilized to find a near optimal solution. Solutions from these models are used in the sampling phase to minimize quantities such as the mean-squared (relative) error or the maximum absolute (relative) error that represent the overall error of the simulation. The idea of the OASIS algorithm can be used to minimize the overall error of an arbitrary simulation associated with multiple estimates. The numerical results show that the OASIS algorithm is an efficient and flexible method for simulation problems for which we can find efficient stratification functions.

## Optimal Allocation

The optimal allocation is a sample allocation method used with stratified sampling, which is designed to provide the best precision (lowest variation) for the least cost (least sample size) (Statistics Dictionary, 2019b). Neyman allocation is a special case of optimal allocation when the total sample size is fixed (Statistics Dictionary, 2019a).

Applying Neyman allocation has been investigated in researches including (Lavallee & Hidirogloui, 1988), (Benedetti et al. 2010) and (Benedetti & Piersimoni, 2012a). Collectively, these methods restrict the number of strata into two or three. In Lavallee & Hidirogloui (1988) method, population is divided into two strata, one of which was used as take-all stratum while the other one was sampled stratum.

In conventional stratified sampling, the fraction of samples to be allocated to strata is typically decided after the stratification is determined, and the focus is on the minimization of variance of the final stratified estimators. However, the optimal allocation in adaptive stratified sampling can be carried out without variance reduction within each stratum (Kawai, 2010). Etore, et al. (2011) proposed an iterative adaptive optimal allocation algorithm. In each iteration, the algorithm adjusts the proportion of further drawings by applying conditional standard deviation estimates. These proportions converge to the optimal allocation fractions. In this method, at least one drawing is allocated to each stratum which is similar to the method proposed in (Etore & Jourdain, 2010) discussed in ‎4.1 which leads to suboptimal allocations in initial iterations.

## Joint Stratification-Allocation

Benedetti and Piersimoni (2012b) proposed a multivariate framework as the extension of Hidiroglou (1986) univariate method discussed in ‎4.2. The size of each strata is defined by a set of univariate thresholds for each auxiliary variable present in the sampling frame. Univariate thresholds make the strata to have “box-shaped” boundaries. However, The multivariate framework lifts the limitation of box-shaped partition boundaries by applying a random search algorithm and using simulated annealing to solve a general combinatorial optimization problem (Lisic et al., 2018).

Barcaroli (2014) proposed an optimal stratification and allocation method to minimize the cost while all precision constraints are satisfied. Their method is used for multivariate cases in which the estimate of target variables in strata are available. Since the number of possible alternative stratifications is high, Genetic algorithm is applied to find near optimal stratification in specific iteration.

Lisic et al. (2018) proposed another optimal stratification and allocation method and used simulated annealing to solve the optimization model.

## Genetic Algorithm

Genetic Algorithm (GA) is a heuristic search technique inspired by evolutionary biology. It is a stochastic algorithm which means that it makes random changes to existing solutions to generate new solutions.

GA starts with an initial population which is a set of potential solutions. Each solution is referred to as an individual and is represented with a chromosome. Each chromosome includes a set of genes that reflect the important characteristics of an individual. These characteristics (genes) are typically represented with 0 or 1 values.

To evaluate the quality of each individual, GA uses a function referred to as the fitness function to assign a fitness value to the individuals. This way, GA is able to select higher quality solutions from the population. GA combines (mates) high quality individuals (referred to as parents) to generate higher quality offspring (children). This procedure is expected to keep the good properties of the high quality individuals in the next generations and eliminate the lower quality solutions. One drawback with this approach is that in the new generations, no new characteristics are explored or added to the population which limits the quality of the final solution.

To address this problem, some changes are applied to each offspring to create new individuals which is referred to as mutation.

Therefore, GA iteratively generates a new population based on the existing population using inheritance, mutation, selection, and crossover operators. The application of these operators along with randomly introduced changes via mutation results in exploring a variety of solutions and finding higher quality solution while there is no guarantee to find the optimal solution.

Several criteria are typically used to terminate GA. It can terminate after certain number of iterations or when the difference between the two best solutions is less than a specified threshold or in case no better solution is discovered within a given number of iterations.

Initial Population

Fitness Calculation

Selection

Terminate?

Yes

Final Solution

Crossover

Mutation

No

Figure 1. Genetic Algorithm Procedure

The performance of GA depends on many factors such as the design of chromosomes and genes, operators, and parameters such as the probability of mutation and number of iterations.

## Simulated Annealing

Simulated Annealing (SA) is a stochastic optimization process that can potentially find the global optima. “Annealing” is an anlogy with the annealing of metals when they cool down.

Simulated Annealing method was proposed in Kirkpatrick et al. (1983) and Černý (1985) to find the minimum of a cost function that has several local minima.

SA algorithm makes a random move at each iteration to explore a different solution. If the move improves the solution, then it is always accepted. Otherwise, the algorithm may or may not accept the move with some probability less than 1. The probability of accepting a non-improving move exponentially decreases with how bad the move is (i.e. the amount by which the solution is worsened). The other factor affecting the probability of accepting a non-improving move is the temperature (T) which is analogous to temperature in the annealing process. At higher temperatures, non-improving moves are more likely to be accepted compared to the lower temperatures. In a typical SA optimization, temperature is high at the beginning and it gradually decreases according to the annealing schedule.

Similar to GA, the quality of SA solutions depend on factors such as the probability and temperature functions as well as the number of iterations. Simulated annealing is used in discrete complex problems such as the set of possible orders of cities to be visited in the Traveling Salesman Problem (TSP). TSP is an optimization problem dealing with finding the shortest route for a salesman to visit a set of cities.

# Proposed Methodology

Two of the methodologies reviewed in the literature were selected to be considered as candidate proposed methodologies. The selected methodologies use an optimization model along with constraints to optimize both stratification and allocation with respect to budget and precision constraints.

The comprehensive literature review performed on sampling methods revealed that adaptive sampling is typically used in surveys of rare, clustered populations (Thompson, 2012). Although adaptive designs are advantageous in terms of efficiency and precision compared to conventional designs, the proposed adaptive methods in the literature seem to be limited to surveys involving small and rare populations of animals, plants, minerals, and fossil-fuel resources. The two identified candidate methodologies have been applied to real surveys on large populations and demonstrated significant improvement compared to conventional non-optimal stratification and allocation methods.

For the purpose of this exploratory research effort, after consulting with the project stakeholders, the research team decided to implement the data processing and sampling procedures in SQL and R which are popular, open, and platform independent languages. Using this platform makes experimentation and customization of the proposed procedures and scripts more convenient. On the other hand, both identified candidate methods have readily available libraries implemented in R. This would allow the project to benefit from the future updates and improvements made by the original authors of the method.

## Current CFS Sample Design

According to the CFS survey methodology the goal of the Commodity Flow Survey (CFS) is to make reliable estimates of the annual total value of shipments, as well as the annual total tonnage and the annual total ton-miles of shipments (Bureau of Transportation Statistics, 2019a). The estimates are needed for areas of interest specified by combination of the auxiliary variables including the origin of the shipment and industry of the establishment making the shipment.

The sample design consists of three stages. In the first stage, the establishments are sampled, while in the second and third stages, weeks and shipments are sampled.

The sample design for the first stage is stratified simple random sampling without replacement. The stratification is based on (establishment) origin by industry by Measure Of Size (MOS) class. MOS needs to be converted to a categorical variable for stratification, and the boundaries of the MOS size classes are determined by Lavallée-Hidiroglou (LH) algorithm so that the sample size needed to achieve the target CV is minimized. The sample size for each stratum is them allocated according to the Neyman allocation.

In the second stage, the 13 weeks in each quarter (referred to as panels) are sequentially allocated to the list of establishments selected in the first stage sorted by origin, industry, and MOS class.

In the third stage, survey respondents are asked to systematically sample their shipments in the weeks determined in the second stage. The sampling interval (i.e. frequency of sampling) is determined by the number of shipments. Smaller intervals (more frequent sampling) is assigned to establishments with fewer shipments while wider sample intervals (less frequent sampling) is used for establishments with more shipments in the selected reporting weeks.

## Candidate Methods

The two candidate methodologies are proposed for the first stage of the CFS survey. The second and third stages of the sample design involve the selection of reporting weeks and shipments. Since there is no auxiliary data available on the variability of the establishments’ shipping activities over time as well as the variability of shipments in terms of value, weight, and milage, imporoving the sample design with the existing information is not possible.

The first candidate methodology is proposed by Lisic et al. (2018) which is an optimal stratification and allocation method based on simulated annealing that considers coefficient of variance and fixed sample size constraints. This methodology was developed to create an optimal sample design for the June Area Survey (JAS) under quality (coefficient of variance) and sample size constraints. The JAS is one of the largest annual National Agricultural Statistics Service (NASS) agricultural area survey projects over the contiguous 48 states designed to account for every acre of land, all agricultural activities, and land uses within segment boundaries (National Agricultural Statistics Service, 2018).

The proposed methodology uses an objective function composed of the sum of penalties of deviations from the target CVs as a proxy for quality constraints. This is a soft constraint since it does not prohibit the model from deviating from the target values. On the other hand, hard constraints are introduced to the model through defining nonlinear constraints.

In order to minimize the objective function, simulated annealing heuristic is used. Simulated annealing is a stochastic optimization process that allows the objective function to explore some nonoptimal states with nonzero probabilities. The iterative process starts with a feasible initial stratification and allocation. A primary sampling unit (PSU) is exchanged in each state. In the same way, allocation is performed by choosing a stratum to accept the PSU. The sample size of the stratum which accepts the PSU is increased by 1 and the sample size of the one that loses the PSU is decreased by 1. The algorithm stops after a specified number of iterations or when the threshold is met. In each iteration, a candidate state is randomly generated. Then, a candidate allocation with regards to the new state is created. The inner loop of the algorithm checks whether the new combination of candidate state and allocation improves the objective function.

The next candidate methodology proposed by Barcaroli (2014), is an optimal stratification and allocation method aiming at minimizing the sample cost while satisfying a set of precision constraints. Also, the value of target variables is assumed to be either available in the frame or it is possible to estimate their standard deviation and mean from the same or a previous round of the same survey.

The process initiates with the analysis of the frame data. First, auxiliary variables are identified from current variables. In case the values of auxiliary variables are continuous, they must be converted into categorical variables using the k-means clustering technique. Atomic strata are constructed and characterized based on the categorical auxiliary variables and distributions of the target variables inside the different strata. Then, precision constrains on the target estimates are constructed. These precision constraints are differentiated by domain values. Bethel algorithm is exploited to determine the required number of units to be selected which needs to be reduced in optimization of stratification later. Once the strata and constraints data frames have been prepared, the frame stratification is optimized, and the required sample size and allocation to satisfy the precision constraints are determined. The resulting optimized strata are then analyzed and new labels are assigned to the sampling frame units. Each label reflects the new strata resulting from the optimal aggregation of the atomic strata. Finally, units are selected from the sampling frame based on stratified random sample selection scheme and the optimal solution is evaluated in terms of expected precision and bias.

### Comparison of the Candidate Methodologies

In the model proposed by Lisic et al. (2018) (referred to as the SA based method in the rest of the document) the number of strata and also the total sample size are assumed to be fixed. In Barcaroli's (2014) model (referred to as the GA based method in the rest of the document), the number of units in each stratum as well as the maximum of the coefficient of variation (CV) for each variable are required. Based on the provided input variables, unknown population characteristics are estimated for both models using Horvitz-Thompson estimator. Moreover, the continuous administrative data is transformed into the categorical data in both sampling methods.

The objective function for the SA based method consists of two terms; the first term is the weighted norm vector of modeled CVs and the second term is the penalty function for violating the constraints. On the other hand, in the GA based method objective function, a fixed cost is added to the summation of the products of cost of interviewing each unit and the cost of allocation. Both methods include CV constraints.

In the SA based method, simulated annealing algorithm is used for optimizing the objective function. The algorithm stops after a given number of iteration or when the CV constraints are met. The GA based method uses Genetic Algorithm as the solution method and the algorithm terminates after a specific number of iterations or when the value of the objective function reaches a given minimum.

# Case Study

To study and evaluate the candidate methodologies, a case study is presented to demonstrate the several proposed methodologies for optimal stratification and allocation in a CFS like scenario. Data sources, pre-processing, frame data, and the optimal solution are discussed in the case study section. A high level discussion of each method and results are discussed for each method separately.

The optimal stratification and allocation method based on Genetic algorithm is evaluated on a case study involving a sampling frame with 100,000 units. The units in the case study are designed to replicate the establishments in CFS. The frame size chosen for the case study is significantly smaller than the actual CFS frame (about 710,000) to reduce the processing time required for the experiments. Larger frames can be generated using the SQL function discussed in ‎4.3.

## Data Sources

The state level freight transportation value and weight were used from FAF 2016 (Bureau of Transportation Statistics, 2019b) estimates Access database. The selected FAF dataset with over 1.5 million records is about 50 MBs in size in CSV format. The complete county file for 2016 County Business Patterns (CBP) (US Census Bureau, 2018) was used to estimate average freight value and weights for each industry at the county level. More details on the steps involved in the data processing are presented in the next section. The CBP dataset with over 2 million records is about 12 MBs in CSV format. A mapping between NAICS and SCTG codes was created based on “NAICS Industries In-scope to the 2017 CFS” list from 2017 CFS methodology (Bureau of Transportation Statistics, 2019a). The excel sheet containing the list of counties making up the CFS areas in 2012 was retrieved from the US Census Bureau website (US Census Bureau, 2018). The raw data sources are available in CSV format in “Raw\_Data” folder on the GitHub repository (Ghanbartehrani, 2019).

## Pre-processing of the Data

All data files were imported in a PostgreSQL (The PostgreSQL Global Development Group, 2019b) relational database to facilitate the pre-processing stage. Total value and weight for each SCTG code were aggregated at the state level in the FAF table. In CBP table, total number of establishments was calculated for each state, county, and NAICS category based on the list of industries in-scope to the 2017 CFS. Then, CBP and FAF tables were joined based on the NAICS/SCTG mapping mentioned in the previous section to add total number of establishments in each industry-state combination in the FAF table. County level value and weights for each industry were estimated by multiplying state level numbers by the ratio of the number of establishments in each county (and industry) over the total number of establishments in each state (and industry). The list of counties in 2012 CFS areas was used to determine the CFS area associated with each establishment based on state and county.

All the SQL scripts used to perform the steps involved in the pre-processing stage are available in “SQL\_Scripts.sql” file available in “SQL” folder on the GitHub repository (Ghanbartehrani, 2019).

## Frame Data Generation

A function in PostgreSQL procedural language (The PostgreSQL Global Development Group, 2019a) was developed for generating sampling frames with user defined size based on the pre-processed data described in the previous section. The function signature is as follows.

generate\_est(frame\_size, source\_table, value\_CV, wgt\_CV, mile\_CV)

“Frame\_size” is the desired number of units in the frame, “source\_table” is the name of the table in which the pre\_processed data is stored, while “value\_CV”, “wgt\_CV”, and “mile\_CV” parameters are the desired Coefficient of Variations for generated values, weights, and mileages for each establishment. Value will be used in this study as a substitute for the Measure of Size (MOS) in the current CFS sample design. Weight and mileage are estimates from FAF and are included in the function for experimental purposes. It is worth mentioning that mileage is a modeled number and therefore is not included in the current CFS frame.

The following is an example call to the function to generate a frame with 100,000 establishments based on the data stored in “fafcbp” table with 0.1 CVs for value, weight, and mileage.

SELECT \* FROM generate\_est(100000, 'fafcbp', 0.1, 0.1, 0.1);

The function distributes the number of units proportional to number of establishments in each county, CFS area, and industry combination. Value, weight, and mileage values for each establishment are generated from the normal distribution using the estimated average values stored in the input tables (discussed in ‎4.2) and standard deviations calculated based on the user provided CVs (0.1 in this case). In other words, the standard deviation of the normal distribution is calculated as σ = µ × CV. Log-normal distribution was suggested to the project team as a better choice to simulate the skewness of the actual population. However, it resulted in the generation of extreme values for value, weight, and mileage in some cases. A Truncated Normal Log-normal distribution (Steven P. Millard, 2019) is recommended to be used with the limits calculated based on the real data to overcome this issue. The experiments in this study was performed based on the frame generated with normal distribution.

The actual size of the generated frame is typically less than the user provided number due to rounding errors. The function does not generate an establishment for a state-county-industry combination if the number of allocated units is less than one (after rounding to the closest integer). The example code provided above resulted in a frame with 98,388 establishments which is available in “100K\_Frame\_newCFS.csv” file available in “R\_Scripts” folder on the GitHub repository (Ghanbartehrani, 2019).

The source code for the function is available in “Generate\_est.sql” file available in “SQL” folder on the GitHub repository (Ghanbartehrani, 2019).

## Method 1: Optimal Stratification and Allocation Based on Genetic Algorithm

The optimal stratification and allocation method proposed by Ballin and Barcaroli (2013) aims at minimizing the total sample cost while satisfying the precision (CV) constraints. This method explores the set of all possible stratifications (referred to as the universe of stratifications) based on atomic strata which is the most detailed stratification derived from the Cartesian product of all auxiliary variables as the solution space. Since the set of all possible stratifications based on the atomic strata is quite large even for cases with a few auxiliary variables (e.g. 4 auxiliary variables each with 3 levels, result in an atomic strata of size 12 with 4,213,597 possible stratifications), full enumeration of the solution space is not possible in reasonable time. To address that, Ballin and Barcaroli (2013) used Genetic Algorithm (GA) which is a heuristic search technique inspired by evolutionary biology. Therefore, this method starts with an initial set of potential solutions and evolve them using inheritance, mutation, selection, and crossover operators at each iteration to improve the solution in future iterations and finally reach a good solution while there is no guarantee to find the optimal solution. This means that only a fraction of possible stratifications is explored in the process. For each stratification, the optimal allocation is determined by Bethel's (1989) multivariate method. The auxiliary variables need to be categorical. Continuous variables are therefore converted to categorical ones using the k-means clustering method proposed by Hartigan and Wong (1979).

The objective function minimizes the total sampling cost. Cost of sampling per unit can be set according to the effort associated with collecting and processing each unit. For simplicity, relative sampling costs (i.e. cost of 2 for units requiring twice as much effort compared to the regular units with cost of 1) can be used in the model. A possible approach to determine the sampling costs is to rank the establishments based on their response rate, quality, or difficulty of the data processing and assign the sampling costs accordingly. If all sampling costs are set to 1, the model minimizes the total sample size.

The quality of the solutions generated by the GA method depends on the quality of the initial solution and therefore, having a higher quality initial solution leads to higher quality final solutions. To improve the quality of the initial solution, K-means algorithm (J. Hartigan & Wong, 1979) can be used instead of a randomly generated initial solution. K-means is a clustering algorithm that aims at dividing a m×n matrix into K clusters such that the sum of the squares of each matrix is minimized. In section ‎9.3 the comparison of the results between a K-means based and randomly generated initial solution is presented.

Barcaroli (2014) implemented their proposed method in an R package titled “SamplingStrata”. The R package “SamplingStrata” (Barcaroli, 2014a) is available on the Comprehensive R Archive Network (CRAN) (*The Comprehensive R Archive Network*, 2019).

### Method 1 Procedure

“SamplingStrata” package needs to be installed prior to running which requires R (The R Foundation, 2019) version 2.15 or newer. The package can be installed by clicking on “Install Packages” from “Tools” menu in R Studio (R Consortium, 2014) and Typing “SamplingStrata” in the search box. “Repository (CRAN)” needs to be chosen as source. The source code, documentation, and samples are available on the package’s GitHub repository (Barcaroli, 2019a).

In the first step, the input data is read and loaded in “CFSFrameData” matrix.

CFSFrameData <- read.csv(file="./100K\_Frame.csv", header=TRUE, sep=",")

Then, the frame based on the loaded data is created as follows.

CFSFrame <- buildFrameDF(df = CFSFrameData,

id = "estno",

X = c("county","naics"),

Y = c("value"),

domainvalue = "state")

df is the matrix in which the input data is stored, id is the column used to uniquely identify the units, auxiliary variables are listed in X, and Y is the list of target variables. The column corresponding with the domain variable is specified in “domainvalue “. “domainvalue” is set to state that needs to be sequential numbers starting from 1 (i.e. 1, 2, 3,..). County, and NAICS are categorical used as auxiliary variables. Value is converted to 15 categories and used as the fourth auxiliary variable in the frame in the next step.

CFSFrame$X4 <- var.bin(CFSFrameData$value, bins=15)

Atomic strata which is the most detailed strata resulting from the cartesian product of all auxiliary variables is then constructed and stored in “AtomicStrata” matrix.

AtomicStrata <- buildStrataDF(CFSFrame, progress = TRUE)

The size of the atomic strata in this example is 1,818 and the first few rows are displayed below.

Table 1. Atomic strata for the case study

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STRATO | N | M1 | S1 | COST | CENS | DOM1 | X1 | X2 |
| 1\*4238 | 1 | 73.5981 | 0 | 1 | 0 | 1 | 1 | 4238 |
| 1\*4543 | 2 | 106.8469 | 16.6881 | 1 | 0 | 1 | 1 | 4543 |
| 101\*311 | 2 | 342.748 | 4.947 | 1 | 0 | 1 | 101 | 311 |
| 101\*321 | 2 | 135.187 | 6.462 | 1 | 0 | 1 | 101 | 321 |
| 101\*323 | 3 | 72.74363 | 3.116371 | 1 | 0 | 1 | 101 | 323 |
| 101\*325 | 3 | 107.4553 | 65.53779 | 1 | 0 | 1 | 101 | 325 |
| 101\*326 | 1 | 425.939 | 0 | 1 | 0 | 1 | 101 | 326 |
| 101\*327 | 2 | 1.718205 | 0.088995 | 1 | 0 | 1 | 101 | 327 |
| 101\*331 | 1 | 410.253 | 0 | 1 | 0 | 1 | 101 | 331 |
| 101\*336 | 3 | 63.59713 | 1.914006 | 1 | 0 | 1 | 101 | 336 |
| 101\*337 | 2 | 251.5355 | 28.5455 | 1 | 0 | 1 | 101 | 337 |
| 101\*339 | 4 | 129.2472 | 89.72297 | 1 | 0 | 1 | 101 | 339 |
| 101\*4231 | 6 | 4661.042 | 279.9693 | 1 | 0 | 1 | 101 | 4231 |
| 101\*4233 | 9 | 21.75234 | 11.32401 | 1 | 0 | 1 | 101 | 4233 |
| 101\*4237 | 3 | 701.8043 | 55.90901 | 1 | 0 | 1 | 101 | 4237 |
| 101\*4238 | 8 | 661.1641 | 33.56814 | 1 | 0 | 1 | 101 | 4238 |
| 101\*4244 | 6 | 454.4305 | 302.5896 | 1 | 0 | 1 | 101 | 4244 |
| 101\*4245 | 3 | 125.8065 | 116.7259 | 1 | 0 | 1 | 101 | 4245 |
| 101\*4249 | 3 | 123.6077 | 9.917914 | 1 | 0 | 1 | 101 | 4249 |
| 101\*4543 | 4 | 235.6593 | 16.14359 | 1 | 0 | 1 | 101 | 4543 |
| 101\*5111 | 3 | 263.265 | 8.865444 | 1 | 0 | 1 | 101 | 5111 |

The first column shows the combination of the values for the auxiliary variables (X1and X2) identifying each stratum, N is the number of units (i.e. establishments) in each stratum. M1 and S1 are mean and standard deviation of the value for each stratum. Cost is the assigned sampling costs (all one). CENS column allows defining take-all (certainty) strata (strata from which all units must be included in the sample) when set to one. In this case study, no take-all strata are defined. X1 and X2 columns are the values of the two auxiliary variables county, and NAICS code.

Take-all (certainty) strata can be specified through the parameter “CENS” in “optimizestrata” function discussed later in this section.

Next, “CV.csv” which contains the CV constraints for each domain (State in this case) is imported.

CVConst <- read.csv("./CV.csv", header=TRUE, sep=",")

Following shows selected rows from “CV.csv”. Each row in the file corresponds with a CV constraint that corresponds with each value of the domain variable. In this case, CV constraints of are defined for each state (presented in “domainvalue” column). Note that the values in “domainvalue” column in “CV.csv” file and the variable assigned to “domainvalue” in “buildFrameDF” function discussed earlier need to be consistent.

Table 2. Selected rows from CV.csv

|  |  |  |
| --- | --- | --- |
| DOM1 | CV1 | Domainvalue |
| DOM1 | 0.02 | 1 |
| DOM2 | 0.03 | 2 |
| DOM3 | 0.02 | 3 |
| DOM4 | 0.04 | 4 |
| DOM5 | 0.02 | 5 |

Then, all input data needs to be checked to ensure consistency using “checkInput” function. The message “Input data have been checked and are compliant with requirements” will be displayed indicating that the input data is valid and ready to be processed by “optimizeStrata” function. Otherwise, an error will be displayed in case an error is detected in the input data.

checkInput(errors = CVConst,

strata = AtomicStrata,

sampframe = CFSFrame)

To use the K-means based initial solution, the solution generated by “KmeansSolution” function is fed as an initial solution (in place of the “suggestions” parameter) for the “optimizeStrata” function. “KmeansSolution” function takes the atomic strata stored in “AtomicStrata” and develops an initial stratification by clustering it. “CVConst” contains the precision constraints on the target variables (CV). “nstrata” is the total number of strata, which is set it to “NA” here, which enables the function to explore and find the best number of clusters by changing the number of clusters from 2 to half of the number of the atomic strata. “Minnumstrata” represents the minimum number of atomic strata to be included in each stratum. “maxclusters” is the maximum number of clusters that the algorithm explores which by default is set to half of the number of atomic strata (to be consistent with having at least 2 atomic strata in each cluster). “showPlot” parameter enables visualizing the clustering results.

solutionKmeans1 <- KmeansSolution(AtomicStrata,

CVConst,

nstrata=NA,

minnumstrat=2,

maxclusters=NA,

showPlot=FALSE)

The next step is to call “optimizeStrata” function which performs the optimal stratification and allocation based on Genetic Algorithm. In this case, a few parameters such as parallel processing, number of iterations (i.e. generations in the genetic algorithm), initialStrata, output files, and plots are specified along with the two required input matrices which are errors (CV constraints) and atomic strata. Here, the initial solution for the genetic algorithm is set to null using “suggestions” parameter. However, in order to speed up the convergence to the optimal solution or improve the quality of the final solution, an initial solution can be provided.

The number of samples the algorithm generates are real numbers (i.e. numbers with decima points) which need to be converted to integer before using the results. ‘realAllocation’ is the parameter that converts the generated sample size values to integer when set to FALSE.

To specify take-all (certainty) strata (i.e. strata from which all units will be included in the sample), a binary vector with the same length as the number of atomic strata needs to be fed to the parameter “CENS” and the parameter “strcens” set to “TRUE”. An element with value set to ‘1’ in “CENS” vector means that the atomic strata corresponding with that row is marked as take-all while a ‘0’ value means a take-some (regular) strata. Following is an example of the CENS binary vector with three take-all atomic strata.

Table 3. An example of CENS binary vector

|  |
| --- |
| CENS |
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |
| 0 |
| 0 |

The full list of input arguments along with their descriptions and an example are provided on the package page on rdrr.io (Barcaroli, 2019b).

The following code is used to generate the GA solution with a randomly generated initial solution.

GA\_solution <- optimizeStrata(errors = CVConst,

strata = AtomicStrata,

parallel = TRUE,

suggestions = NULL,

initialStrata= NA,

iter = 100,

realAllocation = False,

writeFiles = FALSE,

showPlot = FALSE)

To use the previously discussed K-means initial solution, “suggestions” parameter needs to be set to “solutionKmeans1” as depicted in the following code snippet.

Ga\_solution <- optimizeStrata(errors = CVConst,

strata = AtomicStrata,

suggestions = solutionKmeans1,

parallel = TRUE,

initialStrata= NA,

iter = 100,

realAllocation = False,

writeFiles = FALSE,

showPlot = FALSE)

Although the entire results are stored in “GA\_solution”, selected elements can be stored in separate csv files for convenience and further analysis. The two major outputs are “aggr\_strata” and “indices” columns. The GA method reports the final CV based on the formula presented in the appendix (Related formulae) section.

The results including aggregated strata, indices, and final CV can be stored in CSV files.

write.table(Ga\_solution$aggr\_strata,file="./aggr\_strata.csv", sep=",")

write.table(Ga\_solution$indices,file="./indices.csv", sep=",")

expected\_CV(ga\_solution$aggr\_strata)

The stratification is presented in “indices” which is a one column vector listing labels (i.e. strata number) for each atomic stratum. “aggr\_strata” shows the number of samples allocated to each stratum. After finding the optimal stratification, the function ‘updateStrata’ is used to combine the new labels from “indices” with the atomic strata generated earlier.

newstrata <- updateStrata(AtomicStrata,

GA\_solution,

writeFiles = TRUE)

The next step involves ‘updateFrame’ and ‘selectSample’ functions. ‘updateFrame’ updates the frame by incorporating the new labels created by “updateStrata” to the original “CFSFrame” that was created using “buildFrameDF”. ‘selectSample’ function selects samples using simple random sampling without replacement method. In this case, CFSFrame and newstrata are the inputs and the results which include selected units, along with their weights are stored in the variable “sample\_CFS”. Weight is the inverse of the probability of inclusion of each unit in the sample which is calculated by dividing the sample size (values in the SOLUZ column) by the population of each stratum (N). The final step is to evaluate the solution using ‘myEval’ function. ‘myeval’ function selects a random sample from the frame based on the stratification and allocation specified by the solution and reports the actual CV in each domain based on the selected sample. The sample size (‘nsampl’ parameter value) was set to the sample size determined by the GA method. ‘Framenew\_CFS’ and ‘ga\_solution$aggr\_strata’ are other inputs the function needs to operate. The CVs for all domains are reported and compared in eval$coeff\_var.

framenew\_CFS <- updateFrame(CFSFrame,

newstrata,

writeFiles=FALSE)

sample\_CFS <- selectSample(framenew\_CFS,

ga\_solution$aggr\_strata,

nsampl= sum(ga\_solution$aggr\_strata$SOLUZ)

writeFiles=TRUE,

verbatim = TRUE)

source("<path to the source directory>/myeval.R")

eval<- myEval(framenew\_CFS,

ga\_solution$aggr\_strata,

nsampl=sum(ga\_solution$aggr\_strata$SOLUZ),

writeFiles=TRUE,

progress=FALSE)

eval$coeff\_var

### Method 1 Results

In this section, first the results from the GA method with a randomly generated initial solution are presented. The optimal stratification and allocation produced by ‘optimizeStrata’ function are presented in the two tables “aggr\_strata”, and “indices”. The selected rows from these tables are shown in Table 4 and Table 5. “aggr\_strata.csv” contains 8 columns. The first column ‘STRATO’ indicates the label associated with each stratum in each domain. The algorithm assigns labels serially (starting from 1) to each stratum in each domain. Therefore, there can be strata with the same labels in different domains.

M1 and S1 are mean and standard deviation of the value (i.e. the target variable) in each stratum, N is the number of units (i.e. establishments) in each stratum, CENS column shows take-all (certainty) strata (i.e. one if selected as take-all or certainty and zero otherwise), and SOLUZ is the total number of sampling units to be selected from the stratum. Also, the number of rows in “aggr\_strata” is equal to the number of strata in the optimal solution.

Table 4. Selected rows from “aggr\_strata.csv” file

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STRATO | M1 | S1 | N | DOM1 | COST | CENS | SOLUZ |
| 1 | 73.5981 | 0 | 1 | 1 | 1 | 0 | 1 |
| 2 | 193.0291 | 180.4237 | 4 | 1 | 1 | 0 | 2 |
| 3 | 499.7196 | 403.1929 | 16 | 1 | 1 | 0 | 6 |
| 4 | 135.187 | 4.569324 | 2 | 1 | 1 | 0 | 2 |
| 5 | 92.38961 | 65.48738 | 17 | 1 | 1 | 0 | 2 |
| 6 | 264.9097 | 231.8558 | 20 | 1 | 1 | 0 | 4 |
| 7 | 272.34 | 153.599 | 2 | 1 | 1 | 0 | 2 |
| 8 | 29.67268 | 48.99352 | 4 | 1 | 1 | 0 | 2 |

The table of “indices” contains column x that shows the vector of labels for generated strata in the optimal result. The selected rows of “newStrata” is shown in Table 5.

Table 5. Selected rows from “indices.csv” file

|  |
| --- |
| x |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |

“newStrata” table stores the full list of atomic strata along with the values for all auxiliary variables (i.e. X1, X2, …) as well as the labels (i.e. LABEL column) reflecting the allocation of each atomic stratum to the optimal strata. In other words, the atomic strata labeled 1 form the first stratum and so on. Selected rows from “newStrata” is shown in Table 6.

Table 6. Selected rows from “newStrata.csv” file

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STRATO | N | M1 | S1 | COST | CENS | DOM1 | X1 | X2 | LABEL | STRATUM |
| 1\*4238 | 1 | 73.5981 | 0 | 1 | 0 | 1 | 1 | 4238 | 1 | 1\*4238 |
| 1\*4543 | 2 | 106.8469 | 16.6881 | 1 | 0 | 1 | 1 | 4543 | 2 | 1\*4543 |
| 101\*311 | 2 | 342.748 | 4.947 | 1 | 0 | 1 | 101 | 311 | 3 | 101\*311 |
| 101\*321 | 2 | 135.187 | 6.462 | 1 | 0 | 1 | 101 | 321 | 4 | 101\*321 |
| 101\*323 | 3 | 72.74363 | 3.116371 | 1 | 0 | 1 | 101 | 323 | 5 | 101\*323 |
| 101\*325 | 3 | 107.4553 | 65.53779 | 1 | 0 | 1 | 101 | 325 | 6 | 101\*325 |
| 101\*326 | 1 | 425.939 | 0 | 1 | 0 | 1 | 101 | 326 | 7 | 101\*326 |
| 101\*327 | 2 | 1.718205 | 0.088995 | 1 | 0 | 1 | 101 | 327 | 8 | 101\*327 |
| 101\*331 | 1 | 410.253 | 0 | 1 | 0 | 1 | 101 | 331 | 9 | 101\*331 |

Randomly selected units are listed in “sample\_CFS”. The column FPC indicates the total number of sampling units from the stratum divided by the total number of units (population) in the stratum. Weight column is the inverse of the probability of inclusion for each unit in the sample (higher weight means lower probability and vice versa). Selected rows from “sample\_CFS” are shown in Table 7.

Table 7. Selected rows from “sample\_CFS.csv” file

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DOMAIN  VALUE | STRATO | STRATUM | ID | X1 | X2 | Y1 | LABEL | WEIGHTS | FPC |
| 1 | 1 | 1\*4238 | 1 | 1 | 4238 | 73.5981 | 1 | 1 | 1 |
| 1 | 10 | 101\*336 | 841 | 101 | 336 | 61.0069 | 10 | 7.5 | 0.133333 |
| 1 | 10 | 17\*4543 | 94 | 17 | 4543 | 70.6568 | 10 | 7.5 | 0.133333 |
| 1 | 11 | 101\*337 | 844 | 101 | 337 | 280.081 | 11 | 2 | 0.5 |
| 1 | 11 | 83\*339 | 571 | 83 | 339 | 103.51 | 11 | 2 | 0.5 |
| 1 | 12 | 101\*339 | 848 | 101 | 339 | 222.551 | 12 | 3.2 | 0.3125 |
| 1 | 12 | 59\*4543 | 245 | 59 | 4543 | 59.1239 | 12 | 3.2 | 0.3125 |
| 1 | 12 | 51\*339 | 210 | 51 | 339 | 99.5357 | 12 | 3.2 | 0.3125 |

The summary of result from optimizeStrata function is assigned to Ga-solution variable and is displayed in R console once the process completes. The summary of the results for two domains is presented in Figure 2. Results for all domains are reported in section ‎5.2. The last two lines in each domain’s result specify the sample cost (which is the same as sample size since the cost is set to 1) and number of strata.

Domain: 1

Maximum number of strata: 426

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 426

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 100

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 386

\*\*\* Number of strata: 76

---------------------------

Domain: 2

Maximum number of strata: 80

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 80

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 100

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 76

\*\*\* Number of strata: 13

Figure 2. Summary of GA results for 2 domains

The summary of the results for two domains based on the K-means initial solution is presented in Figure 3. Results for all domains are reported in the in section ‎5.3.

Domain: 1

Maximum number of strata: 426

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 426

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 2 2 8 5 2 2 6 11 6 4 8 5 10 11 13 13 6 5 5 8 8 9 6 9 4 8 8 11 13 11 2 5 12 11 9 6 4 5 5 9 9 2 6 4 6 11 11 4 11 11 5 12 5 5 4 2 8 5 5 4 2 11 8 9 3 11 13 7 5 6 2 6 8 4 2 11 13 11 2 5 5 11 5 8 8 9 4 3 2 8 11 4 9 5 3 11 8 8 6 4 8 4 8 5 11 11 4 5 5 4 2 9 8 11 4 2 5 8 2 4 8 11 13 4 8 4 12 11 5 8 8 5 11 2 4 2 11 4 2 9 4 4 2 8 6 9 2 4 5 8 11 4 6 5 7 11 9 6 6 2 9 2 9 11 9 11 4 4 8 5 11 5 13 11 11 5 2 7 9 5 5 11 2 11 13 4 5 4 8 4 8 11 9 11 11 8 3 11 2 5 4 8 2 2 11 5 5 2 13 5 4 2 9 13 12 4 11 8 11 9 5 5 5 5 4 11 8 11 11 2 4 2 5 5 2 2 11 4 9 11 6 11 5 13 11 9 5 2 5 2 4 5 4 11 5 4 4 5 4 4 4 2 4 9 4 6 11 11 8 5 3 11 6 8 6 5 9 4 2 9 2 9 11 6 4 5 2 8 6 7 8 6 5 8 8 12 8 13 11 3 4 6 6 1 4 12 3 9 3 8 7 6 2 2 5 2 11 2 9 11 6 9 2 7 11 5 5 9 4 4 4 4 5 8 4 13 11 4 5 2 6 11 5 5 4 5 2 5 4 11 9 4 7 11 5 4 11 9 8 9 9 8 2 9 6 11 7 4 9 9 3 11 13 13 6 5 8 5 11 13 5 2 2 11 4 5 5 11 13 2 11 2 13 11 4 5 3 9 5 2 2 7 13 8 2 3 6 8 11 13 5 8 9 10 4 12 12 12 9 8 5 2 4 2 8 4

\*\*\* Sample cost: 122

\*\*\* Number of strata: 13

---------------------------

Domain: 2

Maximum number of strata: 80

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 80

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 2 2 8 7 7 2 5 7 7 7 7 7 2 7 5 2 5 2 2 2 7 6 2 2 6 2 7 7 6 2 4 7 1 2 7 2 6 2 7 6 6 7 7 7 7 8 6 2 7 5 1 5 2 3 1 1 7 1 2 2 6 2 6 5 2 5 7 2 7 7 2 3 2 6 7 2 4 7 6 1

\*\*\* Sample cost: 50

\*\*\* Number of strata: 9

Figure 3. Summary of GA results based on K-means initial solution

The results for the first two domains from GA with random and K-means initial solutions are compared in Table 8. The sample size for GA with K-means solution is significantly lower by 68% and 34% in domains 1 and 2 respectively which reflects a lower cost solution that satisfies the same CV constraints.

Table 8. The comparison between GA with random and K-means initial solutions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | GA, random initial solution | GA, K-means initial solution | Difference |
| Domain 1 | Sample size | 386 | 122 | 68% |
| Strata | 76 | 13 | 83% |
| Domain 2 | Sample size | 76 | 50 | 34% |
| Strata | 13 | 9 | 31% |

The total sample size (all domains) for the GA with K-means initial solution is 1,018 while the random initial solution resulted in the sample size of 2,015. Therefore, the K-means initial solution results in 50% reduction in sample size while satisfying the same CV constraints.

To evaluate the quality of the results from GA with K-means initial solution, CVs based on a sample selected from the frame using myEval function are reported in Figure 4.

domain cv(Y1)

[1,] 1 0.0209

[2,] 2 0.0316

[3,] 3 0.0189

[4,] 4 0.0410

[5,] 5 0.0204

Figure 4. Quality of the found solution in GA method

Target CV constraints and calculated CVs are compared in Table 9. The results show that all CV constraints are satisfied in the selected sample.

Table 9. compare CVconstraints(CVConst) and TargetCV of the GA result

|  |  |  |
| --- | --- | --- |
|  | CV Constraint | Actual CV |
| Domain1 | 0.02 | 0.021 |
| Domain2 | 0.03 | 0.031 |
| Domain3 | 0.02 | 0.019 |
| Domain4 | 0.04 | 0.042 |
| Domain5 | 0.02 | 0.019 |

## Method 2: Optimal Stratification and Allocation Using Genetic Algorithm and Simulated Annealing

Lisic et al. (2018) proposed an optimal stratification and allocation method based on simulated annealing that considers coefficient of variation and fixed sample size constraints. This method minimizes sum of penalties of deviations from the target CVs.

Simulated annealing (SA) heuristic is used to minimze the objective function. Simulated annealing is a stochastic optimization process that allows the objective function to explore some randomly generated nonoptimal states. In each iteration, a primary sampling unit (PSU) is exchanged by choosing a stratum to lose and another stratun to accept a PSU. The algorithm stops after a specified number of iterations or when the threshold is met.

The sample size needs to be determined prior to running the model. Additionally, an initial solution is needed for which the authors use k-means clustering method. K-means is a clustering heursitic that does not perform any optimization in terms of CV or sample size. On the other hand, the quality of the final solution resulted from Simulated Annealing depends on the quality of the initial solution (i.e. a higher quality initial solution is expected to result in a higher quality final solution). Therefore, using the optimal stratification and allocation based on GA method as the initial solution for the SA method is expected to result in a higher quality solution compared to a randomly generated initial solution for the SA method. This approach also helps with determining the minimum sample size required to meet the CV constriants which is more convinient and efficient than experimenting with sample size values.

The method discussed in this section uses the GA based method to find a feasible solution that meets the CV constraints and the minimum sample size. Then, the optimal stratification and sample size from the GA solution is used as initial stratification and sample size for the SA based method respectively. The SA based method further improves the stratification and allocation to minimize CV while keeping the sample size constant.

### Method 2 Procedure

After installing “SamplingStrata” package (see the instructions in ‎3.1), “saAlloc” package needs to be installed. “saAlloc” package requires R (The R Foundation, 2019) version 2.15 or newer. The source code, documentation, and samples are available on the GitHub repository for the package (Lisic, 2019). Following are the instructions to install the package.

install.packages("remotes")

remotes::install\_github("jlisic/saAlloc")

The installation procedure (the second command above) should install ‘Rtools’ automatically if it is not already installed. In some cases, this does not happen and results installation failure which can be fixed by installing ‘Rtools’ (The R Foundation, n.d.) manually.

The GA based method is used to produce the initial solution for the simulated annealing process according to the instructions presented in ‎Method 1 Scripts.

‘ga\_solution$aggr\_strata’ and ‘solution$indices’ variables contain the GA results. The stratification is presented in “indices” while “aggr\_strata” shows the number of samples allocated to each stratum. The ‘SOLUZ’ column in ‘ga\_solution$aggr\_strata’ table is used for calculating the total sample size in each domain for the SA method. The function ‘updateStrata’ is used to create a frame referred to as ‘newstrata’ by adding a new stratum label based on solution$indices table to each atomic stratum. This step is necessary because the indices generated by the GA based solution are the combination of all auxiliary variable values while the SA based method requires a singular label for each stratum.

ga\_solution$aggr\_strata

newstrata <- updateStrata(CFSstrata,

ga\_solution,

writeFiles = TRUE)

SamplingStrata reports the sample size for each stratum, however the SA method requires the primary sampling units (PSU) in each stratum. The following script creates a frame referred to as PSU.GA based on the results from the GA method in the format compatible with the SA based method.

Stratification.GA = newstrata

j = 1

for (i in 1:nrow(newstrata)) {

num = newstrata[i,2]

if (num == 1) {

Stratification.GA[j,] = newstrata[i,]

j = j + 1

}

else {

for (k in 1:num) {

Stratification.GA[j,-2] = newstrata[i,-2]

Stratification.GA[j,2] = 1

j = j + 1

}

}

}

View(Stratification.GA)

St1 <- Stratification.GA[

with(Stratification.GA, order(Stratification.GA$DOM1,Stratification.GA$X1)),

]

orderd\_startification.GA<-St1[

with(St1, order(St1$DOM1,St1$X2)),

]

St2 <- CFSFrameData[

with(CFSFrameData, order(CFSFrameData$state,CFSFrameData$county)),

]

orderd\_CFSFrameData<-St2[

with(St2, order(St2$state,St2$naics)),

]

PSU.GA<- cbind.data.frame(orderd\_startification.GA[,1:11],value = orderd\_CFsFrameData$value)

# remove allocation from a stratum with one units from ga\_solution$aggr\_strata

ga\_solution$aggr\_strata\_subset\_SOLUZ <- subset(ga\_solution$aggr\_strata, N != 1, select = c(STRATO:SOLUZ))

In this scenario, state is considered the domain variable which allows for setting independent CV constraints for each state. Therefore, the SA based method needs to be executed for each domain (state) separately. The following script is used to break down the GA output into different domain classes. Also, SA method requires at least one PSU in each stratum to start the algorithm.

PSU\_domain function is used to select the result from PSU.GA for each domain. All strata with one PSU are excluded from all domains. Before executing PSU\_domain function, it is required to set the path to ‘PSU\_domain’ function R using ‘source’ command. This file is available on the project GitHub repository. After setting the path to the function, PSU\_domain function is simply called by typing the name of the function. The input arguments to PSU\_domain function are PSU.GA and the domain which needs to be set to the value of the domain being processed.

To specify samplesize parameter in the SA based method, the function ‘samplesizedom’ is used the same way as PSU\_domain function. The function inputs arguments are ga\_solution$aggr\_strata and domain which needs to be set on the domain being processed. In the following example, PSU\_domain function is called with PSU.GA as input with domain set to 1, and the result is stored in ‘domain1’ variable. The PSUs in domain 1 is then stored in matrix x1, initial stratification from the GA method is stored in ‘labeldom1’, while the total sample size and sample size for individual strata in domain 1 (from the GA based method) are stored in samplesizedom1, and ‘sampleSizeMultiDOM1’ respectively.

# using PSU.GA result for creating parameter labeldom1 and x1

source("<path to the source directory>/PSU\_domain.R")

domain1<- PSU\_domain(PSU.GA,1)

# Initial stratification to be used in simulated Annealing

labeldom1=as.numeric(domain1$LABEL)

# create matrix xi to be used in simulated Annealing

x1 <- as.matrix(as.double(matrix(domain1$value)))

# Using ga\_solution$aggr\_strata\_subset\_SOLUZ result for creating parameter samplesizeMultiDOM1

source("<path to the source directory>/ samplesizedom.R")

samplesizedom1 <-samplesizedom(ga\_solution$aggr\_strata\_subset\_SOLUZ , 1 )

sampleSizeMultiDOM1= sum(ceiling(samplesizedom1[,8]))

Finally, “saMinCV” function from the ‘saAlloc’ R package is called. This function performs the joint optimal stratification and allocation based on Simulated Annealing. The optimal stratification and allocation is determined by exchanging the PSUs between selected strata iteratively to minimize the coefficient of variation (CV). In ‘samincv” function, some parameters such as iteration (number of iterations), targetCV, and penalty (of deviating from target CVs) are specified.

It is worth mentioning that tragetCV is the stopping criteria for simulated annealing (the algorithm stops after reaching the specified value) rather than a CV constraint. This means that the algorithm may not always be able to meet the specified targetCV. Also, setting the iteration parameter to an extremely high value may result in not converging to optimal result in reasonable time. Therefore, trying different targetCV and iteration values may lead to different solutions. Two experiments based on different targetCV and iteration values are discussed here. In the first experiment, the targetCV and iterations are set to 0.02, 100 while in the second experiment, the targetCV and iterations are set to 0.002, 1,000. The final CV reported in the first experiment is 0.024 while the final CV in the second experiment is 0.014. This result shows that Lower targetCV and higher iterations leads to better performance in minimizing CV.

The final results are reported based on 0.0002, 10,000, and 10 values for targetCV, iteration, and penalty parameters respectively. The inputs including xi (PSUs in domain i), label (initial stratification), and samplesize (required total sample size or sample size vector for each domain) are specified by breaking down the GA output into separate domains. In the first instance, samplesize parameter is set to the total sample size for domain 1 (sampleSizeMultiDOM1) while in the second instance, it is set to the vector (strata\_samplesize) reflecting the specified sample size for each strata in domain 1 based on the GA results.

library(saAlloc)

sa\_solution\_dom1\_1 <- saMinCV(

x=x1,

label= labeldom1,

targetCV=(0.0002),

sampleSize=sampleSizeMultiDOM1,

iterations=10000,

penalty = 10,

preserveSatisfied=TRUE,

fpc=TRUE

)

summary(sa\_solution\_dom1\_1)

strata\_samplesize <- samplesizedom1[,8]

sa\_solution\_dom1\_2 <- saMinCV(

x=x1 ,

label= labeldom1,

targetCV=(0.0002),

sampleSize=strata\_samplesize,

iterations=10000,

penalty = 10,

preserveSatisfied=TRUE,

fpc=TRUE

)

summary(sa\_solution\_dom1\_2)

The following script breaks down GA results into separate domains so that saMinCV function can be called for each domain.

source("<path to the source directory>/PSU\_domain.R")

source("<path to the source directory>/samplesizedom.R")

sa\_solutions <- list()

for (i in 1:5){

domain <-PSU\_domain(PSU.GA,i)

labeldom<- as.numeric(domain$LABEL)

X <- as.matrix(as.double(matrix(domain$value)))

samplesizedomm <- samplesizedom(ga\_solution$aggr\_strata\_subset\_SOLUZ ,i)

sampleSizeMultiDOM <-sum(ceiling(samplesizedomm[,8]))

strata\_samplesize <-samplesizedomm[,8]

sa\_solutions[[i]] <- saMinCV(

x = X,

label = labeldom,

targetCV = (0.0002),

sampleSize = strata\_samplesize,

iterations = 10000,

penalty = 10,

preserveSatisfied = TRUE,

fpc = TRUE

)

}

Results for all domains are stored in ‘sa\_solutions’ list. Therefore, the results corresponding with domain i can be accessed by ‘sa\_solutions[i]’. In the results for each domain, ‘CVs’ indicates the initial coefficient of variation (prior to optimization) and the final coefficient variation (after optimization), Initial and final sample sizes are presented in ‘Samplesize’, ‘StrataSize’ is the number of distinct strata in the sample is shown, and stratification labels vector are stored in ‘label’. To display a specific vector in a given domain ‘sa\_solutions[[i]]$vector’ can be used. For example ‘sa\_solutions[[1]]$Samplesize’ shows the sample size in domain 1.

Similar to the previous method, ‘myEval’ function is used here to select a sample and calculate the CV based on it. Before applying this function, results from all domains need to be merged. The following script merges the elements in ‘sa\_solutions’ list in ‘Framenew\_CFS\_SA’ and ‘SA\_aggregatestrata’ objects. CVs for each domain is reported and compared in eval\_SA$coeff\_var.

#integrate solutions for all domains

sa\_solution\_LABEL <- vector()

sa\_solution\_samplesize <- vector()

sa\_solution\_stratasize <- vector()

for (i in 1:length(sa\_solutions)){

sa\_solution\_LABEL <- c(sa\_solution\_LABEL, sa\_solutions[[i]]$label)

sa\_solution\_samplesize <- c(sa\_solution\_samplesize, sa\_solutions[[i]]$samplesize)

sa\_solution\_stratasize <- c(sa\_solution\_stratasize, sa\_solutions[[i]]$strataSize$x)

}

## create Framenew\_CFS\_SA for SA method

Framenew\_CFS\_SA\_1<- PSU.GA[,7:12]

Framenew\_CFS\_SA <- Framenew\_CFS\_SA\_1[

with(Framenew\_CFS\_SA\_1, order(Framenew\_CFS\_SA\_1$DOM1)), ]

Framenew\_CFS\_SA$ID<-orderd\_CFSFrameData$estno

Framenew\_CFS\_SA$LABEL <-sa\_solution\_LABEL

Framenew\_CFS\_SA <- Framenew\_CFS\_SA[, c(1,5,7,2,3,6,4)]

colnames(Framenew\_CFS\_SA)<-colnames(framenew\_CFS)

## create SA\_aggregatestrata for SA method

A <- aggregate(x=Framenew\_CFS\_SA$Y1,by=list(Framenew\_CFS\_SA$DOMAINVALUE,

Framenew\_CFS\_SA$LABEL), FUN="mean")

colnames(A) <- c("domainvalue","LABEL","Y")

aggregate\_mean <-A[order(A$domainvalue),]

B <- aggregate(x=Framenew\_CFS\_SA$Y1,by=list(Framenew\_CFS\_SA$DOMAINVALUE,

Framenew\_CFS\_SA$LABEL), FUN="sd")

colnames(B) <- c("domainvalue","LABEL","Y")

aggregate\_sd <- B[order(B$domainvalue),]

SA\_aggregatestrata <- data.frame(matrix(NA, nrow = length(sa\_solution\_stratasize), ncol = 8))

colnames(SA\_aggregatestrata)= colnames(ga\_solution$aggr\_strata)

SA\_aggregatestrata$SOLUZ <-sa\_solution\_samplesize

SA\_aggregatestrata$N <- sa\_solution\_stratasize

SA\_aggregatestrata$COST <- c(1)

SA\_aggregatestrata$CENS <-c(0)

SA\_aggregatestrata$STRATO <- ga\_solution$aggr\_strata$STRATO

SA\_aggregatestrata$DOM1 <- ga\_solution$aggr\_strata$DOM1

SA\_aggregatestrata$M1 <- aggregate\_mean$Y

SA\_aggregatestrata$S1 <- aggregate\_sd$Y

source("<path to the source directory>/myeval.R")

eval\_SA<- myEval(Framenew\_CFS\_SA,

SA\_aggregatestrata,

nsampl=sum(SA\_aggregatestrata$SOLUZ),

writeFiles=TRUE,

progress=FALSE)

eval\_SA$coeff\_var

### Method 2 Results

#### SA Method Using GA Initial Solution Without K-means

Summary of the results for two domains (i.e. domains 1 and 2) are presented in this section.

Number of iterations was set to 10,000 and target CV was set to 0.0002. $CVs indicates the Initial CV (i.e. prior to optimization using the SA based method) and the final CV.

sa\_solution1

$CVs

Initial Final Target

1 0.01943588 0.005356144 2e-04

sa\_solution2

$CVs

Initial Final Target

1 0.03144267 0.01039823 2e-04

Figure 5. CVs for SA method with GA initial solution

$samplesize table indicates the initial and final sample sizes for each stratum. The total sample size for SA is almost same as GA. However, the stratum having the population equal to one is removed in SA methods.

Total sample size\_dom1= 347

Total sample size\_dom2= 62

Figure 6. Sample sizes for SA method with GA initial solution

#### SA Method Using GA Initial Solution With K-means

Similar to the first method, the GA based method with K-means initial solution can be used to improve the final results. Following are the summary of the results for domains 1 and 2 based on the GA method with K-means initial solution. As discussed in Method 2 procedure, two instances with sample size set to the total sample size (sampleSizeMultiDOM1) and the sample size vector (‘strata\_samplesize’ vector which contains the sample size determined for each strata in the domain) form the GA based solution are provided. The summary of the results for domains 1 and 2 are presented in the following.

**SA method with total sample size from GA method with K-means initial solution**

**Domain1**

$CVs

Initial Final Target

1 0.03951785 0.0137545 0.0002

$sampleSize

Initial Final

n\_1 10 3

n\_2 9 5

n\_3 10 5

n\_4 9 8

n\_5 10 2

n\_6 9 3

n\_7 9 13

n\_8 9 7

n\_9 10 5

n\_10 9 17

n\_11 9 36

n\_12 9 6

n\_13 10 12

$strataSize

Initial Final

0 101 109

1 131 109

2 144 124

3 111 66

4 149 172

5 142 153

6 13 16

7 49 72

8 79 106

9 62 54

10 83 64

11 44 63

12 14 14

**Domain2:**

$CVs

Initial Final Target

1 0.04207412 0.0003603264 0.0002

$sampleSize

Initial Final

n\_1 7 2

n\_2 7 6

n\_3 8 3

n\_4 8 18

n\_5 7 7

n\_6 8 7

n\_7 7 9

$strataSize

Initial Final

0 46 53

1 5 4

2 43 44

3 46 43

4 44 39

5 8 7

6 7 9

**SA method using sample size vector from GA method with K-means initial solution**

**Domain1:**

sa\_solution1

$CVs

Initial Final Target

1 0.02109322 0.01362659 0.0002

$sampleSize

Initial Final

n\_1 2 2

n\_2 7 5

n\_3 5 5

n\_4 15 10

n\_5 2 2

n\_6 2 2

n\_7 7 15

n\_8 2 6

n\_9 4 4

n\_10 20 20

n\_11 44 31

n\_12 3 8

n\_13 9 12

$strataSize

Initial Final

0 101 103

1 131 116

2 144 135

3 111 83

4 149 169

5 142 148

6 13 17

7 49 61

8 79 94

9 62 52

10 83 60

11 44 70

12 14 14

**Domain2:**

$CVs

Initial Final Target

1 0.03156651 0.009075392 0.0002

$sampleSize

Initial Final

n\_1 2 4

n\_2 5 13

n\_3 2 6

n\_4 21 11

n\_5 7 7

n\_6 8 3

n\_7 7 8

$strataSize

Initial Final

0 46 73

1 5 13

2 43 45

3 46 29

4 44 26

5 8 5

6 7 8

Figure 7. Results for SA method with GA with K-means initial solution

The comparison of total sample size and sample size vector are provided in the following table. The sample size is the same in both instances which is due to the fact that the total sample size in both cases are equal. There is a slight difference between the final CVs which can be attributed to the stochasticity of simulated annealing process. This means that if the process is repeated, either of the methods can show superior /inferior final CV values in each domain. Therefore, both methods are expected to result in equally well results.

Table 10. Comparison of total sample size and sample size vector

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Domain | SA/GA with k-means – total sample size | | SA/GA with k-means – sample size vector | |
| Sample Size | Final CV | Sample Size | Final CV |
| 1 | 122 | 0.0137545 | 122 | 0.0136265 |
| 2 | 52 | 0.0003603 | 52 | 0.0090753 |

## Comparison of GA and GA/SA Methods

The SA based method uses the optimal solution from the GA based method with the same sample size and minimizes the CV while keeping the sample size constant. As a result of combining SA and GA methods, not only the CV constraints in all domains are satisfied, but also the CVs in all domains are dramatically reduced as shown in the comparison table below. The sample sizes are the same among both methods, so they are not included in the table. The improvement in CV varies between 12% and 70% among the five domains.

Table 11. Comparison of GA and GA/SA methods

|  |  |  |  |
| --- | --- | --- | --- |
| Domain | Method 1  (GA) eval$coeff\_var | Method 2  (GA\_SA) eval$coeff\_var | Improvement |
| 1 | 0.0209 | 0.0136 | 34.93% |
| 2 | 0.0316 | 0.0094 | 70.25% |
| 3 | 0.0189 | 0.0075 | 60.32% |
| 4 | 0.0410 | 0.0173 | 57.8% |
| 5 | 0.0204 | 0.0180 | 11.76% |

## Method 3: Generalized Lavallée-Hidiroglou Method for Strata Construction

This method is developed to simulate the currant CFS sampling approach to provide a basis for comparison of the results. In the current CFS sample design, the auxiliary variables (i.e. variables used for stratification) are CFS area and industries strata (NAICS) while the target variable is the MOS which is used as the proxy for annual total value of shipment. Lavallée-Hidiroglou (LH) algorithm is used to determine the optimal stratum boundaries that maximize the level of precision or minimize the variance of the mean estimate of stratification for a given total number of strata. In LH algorithm, the CV in each stratum are required to be equalized without any significant increase in the variance of the stratified sample mean. Neyman allocation is used to determine the sample size for each MOS stratum.

In this study, “Stratification” package is applied to simulate the currant CFS sampling approach to provide a basis for comparison of the results. This package uses generalized LH algorithm to construct the strata, and it has the option to perform the allocation using Neyman allocation.

“Stratification” package needs to be installed prior to running which requires R (The R Foundation, 2019) version 2.15 or newer. The package can be installed by clicking on “Install Packages” from “Tools” menu in R Studio (R Consortium, 2014) and typing “Stratification” in the search box while choosing “Repository (CRAN)” as source. The source code, documentation, and samples are available on the package’s GitHub repository (Rivest, 2014/2017).

In this section, the optimal strata boundaries and the optimal sample size are determined in one step using generalized LH method (similar to the CFS sample design) with Kozak’s algorithm. The allocation is performed using Neyman allocation similar to the current CFS sampling method. The same frame from the previously discussed methods is used here too.

The main function in ‘stratification’ package is ‘strata.LH’ which is executed for each domain separately. In ‘strata.LH’ function, the input vector x contains the values of the target variable for each unit in the domain. Also, only one of the parameters of CV or target sample size (n) need to be provided as input. The CV is selected to estimate the population total Ty. The model parameter was set to "none", so the original Lavallee-Hidiroglou method is used. The alloc parameter contains numeric objects q1, q2, q3 to specify the allocation schema including proportional allocation, power allocation, and Nayman allocation. To use Neyman allocation, the alloc parameter was set to (q1=q3=0.5 and q2=0). In this method, the number of desired sampled strata are required to be identified from the beginning. For this purpose, parameter Ls is defined as the desired number of sampled strata (Ls). We used Ls equal to 4. The method fails to converge to the optimal solution if the number of strata (LS) is set to values greater than 20. The following script is for 5 domains.

library(stratification)

source("<path to the source directory>/selectDomain.R")

LH\_solutions <- list()

for (i in 1:5) {

domain <-PSU\_domain(PSU.GA,i)

X <- as.vector(as.numeric(domain$value))

CV <- as.vector(CVConst$CV1)

LH\_solutions[[i]]<-strata.LH(X,

CV= 0.02,

alloc = list(q1 = 0.5, q2 =0, q3 = 0.5),

model='none',

takenone = 0,

Ls=4

)

}

Print(LH\_solutions)

Similar to the two previous methods, ’myEval’ function is used to evaluate the quality of the solution based on a sample. For this reason, ‘Framenew\_CFS\_LH’ and ‘LH\_aggregatestrata’ are created from the ‘LH\_solutions’. The number of sample parameter for myEval function is set to the same number of total sample size from GA and SA which is 972.

LH\_bh <- vector()

LH\_Nh <- vector()

LH\_nh <- vector()

LH\_Label <- vector()

LH\_var <- vector()

LH\_mean <- vector()

for (i in 1:length(LH\_solutions)){

LH\_bh <- c(LH\_bh, LH\_solutions[[i]]$bh)

LH\_Nh <- c(LH\_Nh, LH\_solutions[[i]]$Nh)

LH\_nh <- c(LH\_nh, LH\_solutions[[i]]$nh)

LH\_Label <- c(LH\_Label, LH\_solutions[[i]]$stratumID)

LH\_var <- c(LH\_var, LH\_solutions[[i]]$varh)

LH\_mean <- c(LH\_mean, LH\_solutions[[i]]$meanh)

}

LH\_aggregatestrata <- data.frame(matrix(NA, nrow =20, ncol = 8))

colnames(LH\_aggregatestrata)= colnames(SA\_aggregatestrata)

LH\_aggregatestrata$SOLUZ <-LH\_nh

LH\_aggregatestrata$N <- LH\_Nh

LH\_aggregatestrata$COST <- c(1)

LH\_aggregatestrata$CENS <-c(0)

LH\_aggregatestrata$STRATO <- c(1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4)

LH\_aggregatestrata$DOM1 <- c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5)

LH\_aggregatestrata$M1 <- LH\_mean

LH\_aggregatestrata$S1 <- sqrt(LH\_var)

Framenew\_CFS\_LH\_1<- PSU.GA[,7:12]

Framenew\_CFS\_LH <- Framenew\_CFS\_LH\_1[

with(Framenew\_CFS\_LH\_1, order(Framenew\_CFS\_LH\_1$DOM1)),

]

Framenew\_CFS\_LH$ID<-orderd\_CFSFrameData$estno

Framenew\_CFS\_LH$LABEL <-LH\_Label

Framenew\_CFS\_LH <- Framenew\_CFS\_LH[, c(1,5,7,2,3,6,4)]

colnames(Framenew\_CFS\_LH)<-colnames(framenew\_CFS)

library(SamplingStrata)

source("<path to the source directory>/myeval.R")

eval\_LH<- myEval(Framenew\_CFS\_LH,

LH\_aggregatestrata,

nsampl=972,

writeFiles=TRUE,

progress=FALSE)

eval\_LH$coeff\_var

### Method 3 results

In this method, the number of sampled strata (Ls) was set to value 4. Type column indicates the type of stratification which is take-some or take-all (certainty) stratum, bh indicates optimal stratum boundaries, Nh presents the number of units in each stratum, and nh is the number of units to sample (i.e. sample size) in each stratum. Total sample size indicates the final sampled number (sum(nh) over all strata), anticipated CV indicates the root mean squared error (RMSE) of the mean of the target variable. The formulae to calculate RMSE and Anticipated CV are presented in appendix section (Related formulae). The results for domains 1 and 2 are shown in the following.

> print(LH\_solutions[[1]])

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 100, maxstill = 500, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 275.5 138.0 6302.00 275 12 0.04

stratum 2 | take-some 1 | 550.5 413.0 6302.00 275 12 0.04

stratum 3 | take-some 1 | 836.5 693.5 6816.25 286 13 0.05

stratum 4 | take-some 1 | 1123.0 979.5 6816.25 286 13 0.05

Total 1122 50 0.04

Total sample size: 50

Anticipated population mean: 561.5

Anticipated CV: 0.01994201

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

> print(LH\_solutions[[2]])

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 20, maxstill = 200, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 51.5 26.0 216.67 51 11 0.22

stratum 2 | take-some 1 | 101.5 76.5 208.25 50 11 0.22

stratum 3 | take-some 1 | 150.5 126.0 200.00 49 10 0.20

stratum 4 | take-some 1 | 200.0 175.0 200.00 49 10 0.20

Total 199 42 0.21

Total sample size: 42

Anticipated population mean: 100

Anticipated CV: 0.01969047

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

The CVs in 5 domains based on the selected sample after running ‘myeval’ function are as follows.

Table 12. Comparison of GA and GA/SA methods

|  |  |  |
| --- | --- | --- |
| Domain | Sample size | CV |
| 1 | 50 | 0.3369 |
| 2 | 42 | 0.3837 |
| 3 | 51 | 0.2024 |
| 4 | 49 | 0.2066 |
| 5 | 53 | 0.2749 |

The results show that the although the sample sizes in each domain are lower than methods 1 and 2, the actual CVs based on the selected sample are significantly higher and did not meet the specified CV constraints.

### Comparing Results from Method 2 (SA\_GA) and Generalized LH Method

The result from SA/GA method results indicate that the CV constraints are not only satisfied but also CV in each of the domains was further reduced after the application of the SA based method. However, the LH method results show that the CV constraints are not even satisfied in the selected sample.

Table 12. Comparison of methods 2 and 3 results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CV | | | Sample Size | | |
| domain | Method 2 | LH | % Difference | Method 2 | LH | % Difference |
| 1 | 0.0136 | 0.3403 | 96% | 122 | 51 | 139% |
| 2 | 0.0094 | 0.4183 | 98% | 52 | 42 | 24% |
| 3 | 0.0075 | 0.206 | 96% | 305 | 51 | 498% |
| 4 | 0.0173 | 0.2023 | 91% | 74 | 49 | 51% |
| 5 | 0.018 | 0.2749 | 93% | 419 | 53 | 691% |

Increasing the sample size for the LH method can result in improving the CV to some extent, however, the main cause of the observed significant difference in the CVs is the optimal stratification and allocation performed in method 2. The generalized LH method performs the stratification and allocation separately and therefore their combination is far from optimal.

# Appendix

The complete R script for each of the methods are presented in this section. The latest code is also available in “R\_Scripts” folder on the GitHub repository (Ghanbartehrani, 2019).

## Method 1 Scripts

library(SamplingStrata)

# Read the input frame data from 100K\_Frame.csv file

CFSFrameData <- read.csv(file="./100K\_Frame.csv", header=TRUE, sep=",")

# Build the frame using estno column as identifier, state, county, and naics

# as auxiliary variables and value as target variable

frame <- NULL

frame$id <- CFSFrameData$estno

frame$Y1 <- CFSFrameData$value

CFSFrame <- buildFrameDF(df = CFSFrameData,

id = "estno",

X = c(

"county",

"naics"

),

Y = c("value"

),

domainvalue = "state")

str(CFSFrame)

AtomicStrata <- buildStrataDF(CFSFrame, progress = TRUE)

head(AtomicStrata)

# Read the CV constraints from CV.csv file

CVConst <- read.csv("./CV.csv", header=TRUE, sep=",")

# Check the input data for errors

checkInput(errors = CVConst,

strata = AtomicStrata,

sampframe = CFSFrame)

# Optimization of stratification\_allocation without K-Mean solution

ga\_solution\_withoutkmeans<- optimizeStrata(

errors = CVConst,

strata = AtomicStrata,

cens = NULL,

strcens = FALSE,

alldomains = TRUE,

dom = NULL,

initialStrata = NA,

addStrataFactor = 0.0,

minnumstr = 2,

iter = 500,

pops = 20,

mut\_chance = NA,

elitism\_rate = 0.2,

highvalue = 1e+08,

realAllocation = FALSE,

writeFiles = FALSE,

showPlot = FALSE,

parallel = FALSE

)

sum(ga\_solution\_withoutkmeans $aggr\_strata$SOLUZ)

expected\_CV(ga\_solution\_withoutkmeans $aggr\_strata)

# Writing the stratification and allocation results to csv files

write.table(ga\_solution\_withoutkmeans $aggr\_strata,file="./aggr\_strata.csv", sep=",")

write.table(ga\_solution\_withoutkmeans $indices,file="./indices.csv", sep=",")

# GA with K-mean as initial solution

solutionKmeans1 <- KmeansSolution(AtomicStrata,

CVConst,

nstrata=NA,

minnumstrat=2,

maxclusters=NA,

showPlot=FALSE)

ga\_solution\_with\_kmean<- optimizeStrata(

errors = CVConst,

strata = AtomicStrata,

suggestions = solutionKmeans1,

cens = NULL,

strcens = FALSE,

alldomains = TRUE,

dom = NULL,

initialStrata = NA,

addStrataFactor = 0.0,

minnumstr = 2,

iter = 500,

pops = 20,

mut\_chance = NA,

elitism\_rate = 0.2,

highvalue = 1e+08,

realAllocation = FALSE,

writeFiles = FALSE,

showPlot = FALSE,

parallel = FALSE

)

sum(ga\_solution\_with\_kmean<- $aggr\_strata$SOLUZ)

expected\_CV(ga\_solution\_with\_kmean<- $aggr\_strata)

# select sample and Evaluate CVs in each domain

newstrata <- updateStrata(AtomicStrata,

ga\_solution\_with\_kmean,

writeFiles = TRUE)

framenew\_CFS <- updateFrame(CFSFrame, newstrata, writeFiles=FALSE)

sample\_CFS <- selectSample(framenew\_CFS, ga\_solution\_with\_kmean$aggr\_strata, writeFiles=TRUE, verbatim = TRUE)

source("<path to the source directory>/myeval.R")

eval\_GA<- myEval(framenew\_CFS,

ga\_solution\_with\_kmean $aggr\_strata,

nsampl=sum(ga\_solution\_with\_kmean $aggr\_strata$SOLUZ),

writeFiles=TRUE,

progress=FALSE)

eval\_GA$coeff\_var

## Method 1 Summary of the Results With Random Initial Solution (no K-means)

\*\*\* Domain : 1 1

Number of strata : 426

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 1

Maximum number of strata: 426

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 426

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 350

\*\*\* Number of strata: 59

\*\*\* Domain : 2 2

Number of strata : 80

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 2

Maximum number of strata: 80

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 80

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 62

\*\*\* Number of strata: 6

\*\*\* Domain : 3 3

Number of strata : 181

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 3

Maximum number of strata: 181

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 181

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 328

\*\*\* Number of strata: 24

\*\*\* Domain : 4 4

Number of strata : 297

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 4

Maximum number of strata: 297

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 297

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 174

\*\*\* Number of strata: 10

\*\*\* Domain : 5 5

Number of strata : 834

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 5

Maximum number of strata: 834

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 834

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

\*\*\* Sample cost: 917

\*\*\* Number of strata: 116

\*\*\* Sample size : 1831

\*\*\* Number of strata : 215

---------------------------

## Method 1 Summary of Results With K-means Initial Solution

\*\*\* Domain : 1 1

Number of strata : 426

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 1

Maximum number of strata: 426

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 426

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 2 2 8 5 2 2 6 11 6 4 8 5 10 11 13 13 6 5 5 8 8 9 6 9 4 8 8 11 13 11 2 5 12 11 9 6 4 5 5 9 9 2 6 4 6 11 11 4 11 11 5 12 5 5 4 2 8 5 5 4 2 11 8 9 3 11 13 7 5 6 2 6 8 4 2 11 13 11 2 5 5 11 5 8 8 9 4 3 2 8 11 4 9 5 3 11 8 8 6 4 8 4 8 5 11 11 4 5 5 4 2 9 8 11 4 2 5 8 2 4 8 11 13 4 8 4 12 11 5 8 8 5 11 2 4 2 11 4 2 9 4 4 2 8 6 9 2 4 5 8 11 4 6 5 7 11 9 6 6 2 9 2 9 11 9 11 4 4 8 5 11 5 13 11 11 5 2 7 9 5 5 11 2 11 13 4 5 4 8 4 8 11 9 11 11 8 3 11 2 5 4 8 2 2 11 5 5 2 13 5 4 2 9 13 12 4 11 8 11 9 5 5 5 5 4 11 8 11 11 2 4 2 5 5 2 2 11 4 9 11 6 11 5 13 11 9 5 2 5 2 4 5 4 11 5 4 4 5 4 4 4 2 4 9 4 6 11 11 8 5 3 11 6 8 6 5 9 4 2 9 2 9 11 6 4 5 2 8 6 7 8 6 5 8 8 12 8 13 11 3 4 6 6 1 4 12 3 9 3 8 7 6 2 2 5 2 11 2 9 11 6 9 2 7 11 5 5 9 4 4 4 4 5 8 4 13 11 4 5 2 6 11 5 5 4 5 2 5 4 11 9 4 7 11 5 4 11 9 8 9 9 8 2 9 6 11 7 4 9 9 3 11 13 13 6 5 8 5 11 13 5 2 2 11 4 5 5 11 13 2 11 2 13 11 4 5 3 9 5 2 2 7 13 8 2 3 6 8 11 13 5 8 9 10 4 12 12 12 9 8 5 2 4 2 8 4

\*\*\* Sample cost: 122

\*\*\* Number of strata: 13

\*\*\* Domain : 2 2

Number of strata : 80

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 2

Maximum number of strata: 80

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 80

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 2 2 8 7 7 2 5 7 7 7 7 7 2 7 5 2 5 2 2 2 7 6 2 2 6 2 7 7 6 2 4 7 1 2 7 2 6 2 7 6 6 7 7 7 7 8 6 2 7 5 1 5 2 3 1 1 7 1 2 2 6 2 6 5 2 5 7 2 7 7 2 3 2 6 7 2 4 7 6 1

\*\*\* Sample cost: 50

\*\*\* Number of strata: 9

\*\*\* Domain : 3 3

Number of strata : 181

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 3

Maximum number of strata: 181

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 181

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 56 78 69 67 28 65 71 38 70 13 27 89 82 4 13 49 76 69 76 75 32 90 45 84 11 71 43 88 65 82 49 61 58 72 83 23 64 17 48 29 14 18 77 16 38 86 85 78 72 69 8 42 48 12 33 15 57 25 31 44 85 14 86 5 73 51 46 50 2 34 30 27 66 81 20 16 57 54 85 26 87 5 14 54 21 46 24 62 83 16 50 84 36 3 1 47 77 63 10 86 40 15 3 55 9 56 18 17 68 77 60 24 37 37 58 66 80 22 74 4 54 24 25 18 16 35 42 52 79 56 58 62 84 6 36 33 7 41 85 53 38 79 18 87 78 85 36 41 85 19 36 66 78 23 56 10 86 56 72 67 73 39 63 87 59 8 22 9 33 78 30 36 43 86 23 66 59 21 23 58 25

\*\*\* Sample cost: 323

\*\*\* Number of strata: 33

\*\*\* Domain : 4 4

Number of strata : 297

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 4

Maximum number of strata: 297

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 297

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 9 4 4 4 4 4 4 8 1 4 2 4 4 4 4 8 4 4 8 1 4 9 4 8 5 8 5 8 1 1 5 1 8 1 2 1 9 4 9 6 2 7 8 5 5 9 2 3 8 6 6 1 6 9 5 2 4 8 4 4 4 8 8 1 4 1 5 1 1 1 8 4 4 5 1 5 4 1 2 8 5 1 9 1 7 8 2 2 9 4 9 4 1 1 1 9 4 2 1 4 9 1 9 8 9 1 9 7 8 2 2 6 1 9 9 1 4 5 8 5 5 1 1 4 1 4 4 9 1 1 4 4 4 4 4 1 4 4 1 4 4 9 8 9 4 8 4 4 2 8 5 2 9 9 1 9 4 9 8 1 5 8 9 5 5 8 5 1 9 8 4 4 4 9 8 1 4 5 8 1 8 1 1 9 8 1 5 9 4 1 8 5 9 1 1 4 4 4 5 8 9 9 1 9 8 1 1 4 5 8 1 1 4 4 8 9 8 1 4 1 1 4 4 4 4 1 8 4 1 1 4 4 4 1 4 8 4 4 9 8 1 4 9 9 4 2 1 4 1 4 1 7 8 1 5 9 2 8 5 5 1 6 4 9 5 1 1 4 9 4 4 1 9 4 4 4 5 5 4 1 9 4 9 4 4 5 1 4 8 7 1 4 1 8 8 4 2

\*\*\* Sample cost: 72

\*\*\* Number of strata: 9

\*\*\* Domain : 5 5

Number of strata : 834

---------------------------------------------

Optimal stratification with Genetic Algorithm

---------------------------------------------

\*\*\* Parameters \*\*\*

---------------------------

Domain: 5

Maximum number of strata: 834

Minimum number of units per stratum: 2

Take-all strata (TRUE/FALSE): FALSE

number of sampling strata : 834

Number of target variables: 1

Number of domains: 1

Number of GA iterations: 500

Dimension of GA population: 20

Mutation chance in GA generation: NA

Elitism rate in GA generation: 0.2

Chance to add strata to maximum: 0

Allocation with real numbers instead of integers: FALSE

Suggestion: 2 15 5 1 6 12 2 6 2 11 5 12 8 1 11 1 6 4 8 3 12 8 6 5 14 1 5 14 8 8 8 14 8 14 8 8 14 8 8 8 8 8 8 11 8 8 5 8 8 5 8 8 8 5 8 1 5 2 11 8 14 8 8 8 8 8 14 8 8 5 1 14 6 5 14 5 14 14 1 8 11 11 5 11 6 8 4 1 8 5 14 5 14 2 2 8 8 8 8 14 8 14 8 8 11 8 11 5 14 5 8 8 8 2 8 8 8 8 8 11 14 6 14 8 1 2 2 8 2 14 5 5 1 8 15 11 8 2 2 8 5 14 14 8 14 8 14 14 14 14 8 5 14 8 8 14 8 14 6 14 1 6 14 11 8 5 14 5 8 5 5 5 2 12 8 4 6 8 5 2 8 11 8 14 8 8 8 5 8 2 14 8 8 8 14 8 14 8 14 8 14 8 14 8 14 8 14 8 11 2 8 14 2 8 8 8 8 2 11 8 5 14 14 8 1 14 2 8 2 2 14 6 8 12 6 2 2 8 5 8 2 8 8 8 14 8 2 8 8 8 8 8 8 8 1 13 11 10 9 9 12 3 12 4 4 13 14 3 7 3 3 10 14 10 13 2 3 15 12 3 12 14 8 8 8 14 8 8 8 2 2 14 8 2 8 14 5 8 6 8 8 8 14 8 14 2 14 2 8 1 2 8 14 14 8 2 14 14 14 14 8 8 8 8 8 8 8 8 8 14 8 14 5 8 8 8 8 8 8 8 14 14 8 2 8 8 14 14 8 8 2 1 14 11 8 8 8 8 14 14 5 8 1 2 8 5 8 2 8 2 12 8 8 14 8 14 8 8 14 8 8 14 8 14 8 8 8 8 8 14 8 8 14 14 8 8 8 14 8 14 4 5 4 3 3 11 15 11 11 6 7 8 15 4 15 4 13 8 9 7 8 12 1 5 15 11 2 8 11 14 8 8 2 8 14 2 14 5 8 6 2 8 8 8 5 8 8 8 8 11 6 2 6 15 5 11 14 5 5 15 8 1 6 1 1 7 8 3 12 8 5 2 14 12 2 1 14 12 2 11 14 5 14 1 8 5 5 11 11 15 8 7 1 8 5 2 8 11 14 14 14 8 8 8 8 8 8 14 2 2 14 8 2 8 14 14 14 8 11 2 8 8 8 14 8 1 12 14 6 15 12 11 12 14 1 5 4 8 12 12 6 1 3 8 13 15 8 11 11 2 1 14 11 4 11 4 7 7 5 6 5 1 6 15 8 1 12 12 15 7 8 13 15 8 12 2 5 15 11 12 14 12 4 14 14 14 14 8 14 11 11 5 8 12 2 8 11 2 14 11 5 2 1 2 1 2 11 8 14 11 8 2 2 2 2 1 8 15 11 8 2 5 8 2 8 5 5 11 11 8 8 5 8 2 8 14 14 2 14 2 8 11 14 8 14 2 8 2 8 2 6 14 5 14 8 5 2 2 8 14 5 11 6 8 15 11 8 11 8 2 14 5 11 5 11 14 8 14 5 8 14 2 2 2 5 8 6 5 8 14 8 5 14 12 14 1 12 14 11 2 2 2 11 8 1 5 1 12 15 8 7 12 8 11 14 14 1 2 5 14 2 6 8 8 14 8 14 14 2 2 2 8 5 14 8 14 8 2 8 8 8 14 8 8 8 14 8 14 2 8 1 14 8 8 2 8 8 8 8 8 14 8 8 8 14 2 8 5 8 8 14 14 8 8 2 14 5 8 1 5 8 8 14 8 2 6 12 5 6 2 8 14 5 8 2 2 11 5 5 8 12 5 8 2 14 8 5 8 6 8 11 14 14 8 8 2 8 14 2 14 11 8 1 11 8 14 11 8 5 8 14 8 8 8

\*\*\* Sample cost: 434

\*\*\* Number of strata: 15

\*\*\* Sample size : 1001

\*\*\* Number of strata : 79

---------------------------

## Method 2 Scripts

# GA method

library(SamplingStrata)

# Read the input frame data from 100K\_Frame.csv file

CFSFrameData <- read.csv(file="./100K\_Frame.csv", header=TRUE, sep=",")

# Build the frame using estno column as identifier, state, county, and naics

# as auxiliary variables and value as target variable

CFSFrame <- buildFrameDF(df = CFSFrameData,

id = "estno",

X = c("state","county","naics"),

Y = c("value"),

domainvalue = "naics")

# Converting value to 15 categories and using it as the fourth auxiliary variable in the frame

CFSFrame$X4 <- var.bin(CFSFrameData$value, bins=15)

# Building the atomic strat based on the frame

AtomicStrata <- buildStrataDF(CFSFrame, progress = TRUE)

# Uncomment and run the following line to view the atomic strata

#str(AtomicStrata)

# Read the CV constraints from CV.csv file

CVConst <- read.csv("./CV.csv", header=TRUE, sep=",")

# Check the input data for errors

checkInput(errors = CVConst,

strata = AtomicStrata,

sampframe = CFSFrame)

# Optimization of stratification

solution <- optimizeStrata(errors = CVConst,

strata = AtomicStrata,

parallel = TRUE,

iter = 100,

writeFiles = FALSE,

showPlot = FALSE)

# Writing the stratification and allocation results to csv files

write.table(solution$aggr\_strata,file="./aggr\_strata.csv", sep=",")

write.table(solution$indices,file="./indices.csv", sep=",")

#------------------------SA method-----------------------------------------

Stratification.GA = newstrata

j = 1

for (i in 1:nrow(newstrata)) {

num = newstrata[i,2]

if (num == 1) {

Stratification.GA[j,] = newstrata[i,]

j = j + 1

}

else {

for (k in 1:num) {

Stratification.GA[j,-2] = newstrata[i,-2]

Stratification.GA[j,2] = 1

j = j + 1

}

}

}

View(Stratification.GA)

St1 <- Stratification.GA[

with(Stratification.GA, order(Stratification.GA$DOM1,Stratification.GA$X1)),

]

orderd\_startification.GA<-St1[

with(St1, order(St1$X2)),

]

St2 <- CFSFrameData[

with(CFSFrameData, order(CFSFrameData$state,CFSFrameData$county)),

]

orderd\_CFSFrameData<-St2[

with(St2, order(St2$naics)),

]

PSU.GA<- cbind.data.frame(orderd\_startification.GA[,1:11],value = orderd\_CFSFrameData$value)

source("<path to the source directory>/PSU\_domain.R")

source("<path to the source directory>/samplesizedom.R")

library(saAlloc)

sa\_solutions <- list()

for (i in 1:5){

domain <-PSU\_domain(PSU.GA,i)

labeldom<- as.numeric(domain$LABEL)

X <- as.matrix(as.double(matrix(domain$value)))

samplesizedomm <- samplesizedom(ga\_solution$aggr\_strata,i)

sampleSizeMultiDOM <-sum(ceiling(samplesizedomm[,8]))

strata\_samplesize <-samplesizedomm[,8]

sa\_solutions[[i]] <- saMinCV(

x = X,

label = labeldom,

targetCV = (0.0002),

sampleSize = strata\_samplesize,

iterations = 10000,

penalty = 10,

preserveSatisfied = TRUE,

fpc = TRUE

)

summary(sa\_solutions[[i]])

}

# comparing selectsample and muyEval function in Sa method

#integrate solutions from 5 domians for first scenario

sa\_solution\_LABEL <- vector()

sa\_solution\_samplesize <- vector()

sa\_solution\_stratasize <- vector()

for (i in 1:length(sa\_solutions)){

sa\_solution\_LABEL <- c(sa\_solution\_LABEL, sa\_solutions[[i]]$label)

sa\_solution\_samplesize <- c(sa\_solution\_samplesize, sa\_solutions[[i]]$samplesize)

sa\_solution\_stratasize <- c(sa\_solution\_stratasize, sa\_solutions[[i]]$strataSize$x)

}

write.table(Framenew\_CFS\_SA,file="C:/Users/sa129715/Documents/R/test11.19/Framenew\_CFS\_SA\_updated.csv", sep=",")

## create Framenew\_CFS\_SA for Sa method

Framenew\_CFS\_SA\_1<- PSU.GA[,7:12]

Framenew\_CFS\_SA <- Framenew\_CFS\_SA\_1[

with(Framenew\_CFS\_SA\_1, order(Framenew\_CFS\_SA\_1$DOM1)),

]

Framenew\_CFS\_SA$ID<-orderd\_CFSFrameData$estno

Framenew\_CFS\_SA$LABEL <-sa\_solution\_LABEL

Framenew\_CFS\_SA <- Framenew\_CFS\_SA[, c(1,5,7,2,3,6,4)]

colnames(Framenew\_CFS\_SA)<-colnames(framenew\_CFS)

## create SA\_aggregatestrata for Sa method

A <-aggregate(x=Framenew\_CFS\_SA$Y1,by=list(Framenew\_CFS\_SA$DOMAINVALUE,Framenew\_CFS\_SA$LABEL), FUN="mean")

colnames(A) <- c("domainvalue","LABEL","Y")

aggregate\_mean <-A[order(A$domainvalue),]

B<- aggregate(x=Framenew\_CFS\_SA$Y1,by=list(Framenew\_CFS\_SA$DOMAINVALUE,Framenew\_CFS\_SA$LABEL), FUN="sd")

colnames(B) <- c("domainvalue","LABEL","Y")

aggregate\_sd <- B[order(B$domainvalue),]

SA\_aggregatestrata <- data.frame(matrix(NA, nrow = length(sa\_solution\_stratasize), ncol = 8))

colnames(SA\_aggregatestrata)= colnames(ga\_solution$aggr\_strata)

SA\_aggregatestrata$SOLUZ <-sa\_solution\_samplesize

SA\_aggregatestrata$N <- sa\_solution\_stratasize

SA\_aggregatestrata$COST <- c(1)

SA\_aggregatestrata$CENS <-c(0)

SA\_aggregatestrata$STRATO <- ga\_solution$aggr\_strata$STRATO

SA\_aggregatestrata$DOM1 <- ga\_solution$aggr\_strata$DOM1

SA\_aggregatestrata$M1 <- aggregate\_mean$Y

SA\_aggregatestrata$S1 <- aggregate\_sd$Y

# select sample and eval function

sample\_CFS\_SA <- selectSample(Framenew\_CFS\_SA, SA\_aggregatestrata, writeFiles=TRUE, verbatim = TRUE)

source("<path to the source directory>/myeval.R")

eval\_SA<- myEval(Framenew\_CFS\_SA,

SA\_aggregatestrata,

nsampl=sum(SA\_aggregatestrata$SOLUZ),

writeFiles=TRUE,

progress=FALSE)

eval\_SA$coeff\_var

## Method 2 Results Based on Sample Size From GA Without K-means Initial Solution

$CVs

Initial Final Target

1 0.01949074 0.005139291 0.0002

$sampleSize

Initial Final

n\_2 2 6

n\_3 44 11

n\_4 2 3

n\_5 2 4

n\_6 8 5

n\_7 4 4

n\_8 20 12

n\_9 7 6

n\_10 30 6

n\_11 2 3

n\_12 3 4

n\_13 16 22

n\_14 10 5

n\_15 2 6

n\_16 8 12

n\_17 6 3

n\_18 8 8

n\_19 10 6

n\_20 2 4

n\_21 4 5

n\_22 2 5

n\_23 3 6

n\_24 6 6

n\_25 2 7

n\_26 5 8

n\_27 2 5

n\_28 7 4

n\_29 4 6

n\_30 7 19

n\_31 23 13

n\_32 2 4

n\_33 2 4

n\_34 2 7

n\_35 2 4

n\_36 2 5

n\_37 2 6

n\_38 2 4

n\_39 4 7

n\_40 14 10

n\_41 10 5

n\_42 2 5

n\_43 2 5

n\_44 2 4

n\_46 2 4

n\_47 4 6

n\_48 2 8

n\_50 37 13

n\_51 2 5

n\_52 2 4

n\_53 16 9

n\_54 6 18

n\_55 3 5

n\_56 2 5

n\_57 2 9

n\_58 3 10

$strataSize

Initial Final

0 3 24

1 58 11

2 19 31

3 15 30

4 31 13

5 29 28

6 39 12

7 26 18

8 30 6

9 18 29

10 13 9

11 50 30

12 62 15

13 12 16

14 28 16

15 31 11

16 55 30

17 21 6

18 4 18

19 38 26

20 4 31

21 29 46

22 26 6

23 3 9

24 21 15

25 9 10

26 12 4

27 22 25

28 14 21

29 23 13

30 18 22

31 4 17

32 10 32

33 9 22

34 4 23

35 2 6

36 13 26

37 22 25

38 14 10

39 11 5

40 9 19

41 36 46

42 11 25

43 33 52

44 23 10

45 9 18

46 37 13

47 18 43

48 15 25

49 16 9

50 28 25

51 7 18

52 6 41

53 2 9

54 17 19

$runTime

user system elapsed

0.15 0.00 0.17

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.03155018 0.006461559 0.0002

$sampleSize

Initial Final

n\_1 2 4

n\_3 3 9

n\_4 10 10

n\_5 7 10

n\_6 39 13

n\_8 2 7

n\_9 2 12

$strataSize

Initial Final

0 12 29

1 3 9

2 101 87

3 25 26

4 39 13

5 11 21

6 6 12

$runTime

user system elapsed

0.03 0.00 0.04

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.01974265 0.007020116 0.0002

$sampleSize

Initial Final

n\_2 3 12

n\_3 2 9

n\_4 7 11

n\_5 2 5

n\_6 2 7

n\_7 2 12

n\_8 19 21

n\_9 2 7

n\_10 22 25

n\_11 2 8

n\_12 2 5

n\_13 2 4

n\_14 2 9

n\_15 2 7

n\_16 53 28

n\_17 7 21

n\_18 2 8

n\_19 8 31

n\_20 8 20

n\_21 152 45

n\_22 25 20

n\_23 2 7

n\_24 2 5

n\_25 2 5

$strataSize

Initial Final

0 43 72

1 33 80

2 28 24

3 2 31

4 20 59

5 27 50

6 68 23

7 38 48

8 183 116

9 58 92

10 18 28

11 33 114

12 10 30

13 36 38

14 93 28

15 48 55

16 102 135

17 36 48

18 88 79

19 156 45

20 219 72

21 59 89

22 85 98

23 52 81

$runTime

user system elapsed

0.11 0.00 0.11

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.0402844 0.009421362 0.0002

$sampleSize

Initial Final

n\_1 2 8

n\_2 2 8

n\_3 2 6

n\_4 11 6

n\_5 14 5

n\_6 2 6

n\_7 6 10

n\_8 14 8

n\_9 3 7

n\_10 7 4

n\_11 42 18

n\_12 3 25

n\_13 5 6

n\_14 9 8

n\_15 5 11

n\_16 32 18

n\_17 2 7

$strataSize

Initial Final

0 16 52

1 5 65

2 12 26

3 53 21

4 89 18

5 14 35

6 18 20

7 92 49

8 45 78

9 39 11

10 42 18

11 24 37

12 47 61

13 87 72

14 33 29

15 43 18

16 11 60

$runTime

user system elapsed

0.06 0.00 0.06

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.01976252 0.01606354 0.0002

$sampleSize

Initial Final

n\_2 3 4

n\_3 15 17

n\_5 2 2

n\_6 3 4

n\_7 50 47

n\_8 10 9

n\_9 10 11

n\_10 12 13

n\_11 34 33

n\_12 46 45

n\_13 42 41

n\_14 6 7

n\_15 10 10

n\_16 5 7

n\_17 14 13

n\_18 34 31

n\_19 6 7

n\_20 6 8

n\_21 2 2

n\_22 2 2

n\_23 15 17

n\_24 14 13

n\_25 2 3

n\_26 26 24

n\_27 64 57

n\_28 11 13

n\_29 6 8

n\_30 6 8

n\_31 8 9

n\_32 2 3

n\_33 4 4

n\_34 2 2

n\_35 6 8

n\_36 5 7

n\_37 2 3

n\_38 2 2

n\_39 32 30

n\_40 232 228

n\_41 15 16

n\_42 28 24

n\_43 44 40

n\_44 47 44

n\_45 13 14

n\_46 3 4

n\_47 2 2

n\_48 4 4

n\_49 3 3

n\_50 2 2

n\_51 4 3

n\_52 2 4

n\_53 2 2

n\_54 3 3

n\_55 3 5

n\_56 5 6

n\_57 6 6

n\_58 18 18

n\_59 2 3

n\_60 2 2

n\_61 2 3

n\_62 2 3

$strataSize

Initial Final

0 79 102

1 102 101

2 2 48

3 57 68

4 563 397

5 459 464

6 378 381

7 316 287

8 352 278

9 1110 871

10 423 300

11 279 298

12 359 342

13 134 160

14 327 299

15 120 82

16 214 232

17 130 144

18 30 63

19 96 164

20 167 174

21 330 268

22 45 110

23 197 155

24 339 249

25 237 235

26 112 129

27 38 51

28 127 136

29 90 151

30 156 173

31 126 175

32 79 94

33 103 113

34 72 122

35 23 37

36 353 349

37 1034 792

38 109 104

39 105 103

40 79 61

41 203 200

42 323 287

43 239 264

44 622 684

45 283 308

46 169 225

47 84 164

48 68 73

49 50 70

50 66 136

51 41 45

52 111 156

53 81 93

54 165 179

55 319 268

56 94 135

57 30 109

58 37 91

59 13 100

$runTime

user system elapsed

0.37 0.00 0.37

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

## Method 2 Results Based on GA With Random Initial Solution (No K-means)

$CVs

Initial Final Target

1 0.01943588 0.005356144 2e-04

$sampleSize

Initial Final

n\_2 2 4

n\_3 6 7

n\_4 2 5

n\_5 2 6

n\_6 4 12

n\_7 2 2

n\_8 2 4

n\_9 2 3

n\_10 11 6

n\_11 2 3

n\_12 5 2

n\_13 8 5

n\_14 14 8

n\_15 4 7

n\_16 6 4

n\_17 4 4

n\_18 10 11

n\_19 3 5

n\_20 2 3

n\_21 2 5

n\_22 9 11

n\_23 6 7

n\_24 2 2

n\_25 2 4

n\_26 2 4

n\_27 4 15

n\_28 3 4

n\_29 6 11

n\_30 22 10

n\_31 4 5

n\_32 10 10

n\_33 11 3

n\_34 2 3

n\_35 6 4

n\_36 2 4

n\_37 2 3

n\_38 16 8

n\_39 9 13

n\_40 2 4

n\_41 6 5

n\_42 2 3

n\_43 10 10

n\_44 2 3

n\_45 2 4

n\_46 2 5

n\_47 54 13

n\_48 6 3

n\_49 2 4

n\_50 7 4

n\_51 2 6

n\_53 23 16

n\_54 2 5

n\_55 2 5

n\_56 4 2

n\_57 5 21

n\_59 3 7

$strataSize

Initial Final

0 4 19

1 16 7

2 2 20

3 20 30

4 18 12

5 3 24

6 8 7

7 25 31

8 15 6

9 19 34

10 16 3

11 8 6

12 37 18

13 20 29

14 28 14

15 18 9

16 15 11

17 16 26

18 10 7

19 16 20

20 11 11

21 15 7

22 11 29

23 11 28

24 10 23

25 10 15

26 19 10

27 17 26

28 22 10

29 29 48

30 60 18

31 11 3

32 20 34

33 26 4

34 14 35

35 15 23

36 20 8

37 54 16

38 28 57

39 32 17

40 28 28

41 35 22

42 5 26

43 13 23

44 8 16

45 54 13

46 19 11

47 13 39

48 71 46

49 14 27

50 33 16

51 9 23

52 9 21

53 14 7

54 28 30

55 17 16

$runTime

user system elapsed

0.72 0.00 0.78

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution2)

$syscall

sa\_solution2

$CVs

Initial Final Target

1 0.03144267 0.01039823 2e-04

$sampleSize

Initial Final

n\_1 4 11

n\_2 21 4

n\_3 10 16

n\_4 10 16

n\_5 2 8

n\_6 15 7

$strataSize

Initial Final

0 22 32

1 21 5

2 100 112

3 21 16

4 11 25

5 24 9

$runTime

user system elapsed

0.25 0.00 0.25

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution3)

$syscall

sa\_solution3

$CVs

Initial Final Target

1 0.01979399 0.007330385 2e-04

$sampleSize

Initial Final

n\_1 2 7

n\_2 2 6

n\_3 4 8

n\_4 7 12

n\_5 2 6

n\_6 2 6

n\_7 3 5

n\_8 16 22

n\_9 3 10

n\_10 3 13

n\_11 50 31

n\_12 7 18

n\_13 2 6

n\_14 8 31

n\_15 8 20

n\_16 146 43

n\_17 2 6

n\_18 6 12

n\_19 16 17

n\_20 3 8

n\_21 3 4

n\_22 10 14

n\_23 17 14

n\_24 6 9

$strataSize

Initial Final

0 3 66

1 17 41

2 129 134

3 101 73

4 27 33

5 50 67

6 33 31

7 76 66

8 58 75

9 26 37

10 88 31

11 48 52

12 100 154

13 36 44

14 88 80

15 146 43

16 18 80

17 50 67

18 129 64

19 65 78

20 53 95

21 61 36

22 54 29

23 80 60

$runTime

user system elapsed

0.45 0.00 0.45

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution4)

$syscall

sa\_solution4

$CVs

Initial Final Target

1 0.04051477 0.00924114 2e-04

$sampleSize

Initial Final

n\_1 3 10

n\_2 2 21

n\_3 2 11

n\_4 54 19

n\_5 76 37

n\_6 9 18

n\_7 15 27

n\_8 3 7

n\_9 8 11

n\_10 2 13

$strataSize

Initial Final

0 14 45

1 16 163

2 11 68

3 361 103

4 80 37

5 88 114

6 38 39

7 12 29

8 42 16

9 8 56

$runTime

user system elapsed

0.25 0.00 0.33

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution5)

$syscall

sa\_solution5

$CVs

Initial Final Target

1 0.01956838 0.01533046 2e-04

$sampleSize

Initial Final

n\_1 2 2

n\_2 18 18

n\_3 2 2

n\_4 13 13

n\_5 2 2

n\_6 6 6

n\_7 16 16

n\_8 3 3

n\_9 37 37

n\_10 10 9

n\_11 34 33

n\_12 3 3

n\_13 12 12

n\_14 4 4

n\_15 23 22

n\_16 2 2

n\_17 6 5

n\_18 2 2

n\_19 2 2

n\_20 3 3

n\_21 2 2

n\_22 2 2

n\_23 2 2

n\_24 2 2

n\_25 2 2

n\_26 2 2

n\_27 2 2

n\_28 2 2

n\_29 2 2

n\_30 123 122

n\_31 2 2

n\_32 18 17

n\_33 6 5

n\_34 23 24

n\_35 2 2

n\_36 2 2

n\_37 2 2

n\_38 6 6

n\_39 5 5

n\_40 209 211

n\_41 2 2

n\_42 4 5

n\_43 9 11

n\_44 6 6

n\_45 13 14

n\_46 9 10

n\_47 18 24

n\_48 29 22

n\_49 7 7

n\_50 4 4

n\_51 2 3

n\_52 2 2

n\_53 2 2

n\_54 9 9

n\_55 2 2

n\_56 5 6

n\_57 2 2

n\_58 2 2

n\_59 3 3

n\_60 2 2

n\_61 22 21

n\_63 2 2

n\_64 2 2

n\_65 12 12

n\_66 2 2

n\_67 2 2

n\_68 13 13

n\_70 2 2

n\_71 4 3

n\_72 2 3

n\_73 4 3

n\_74 2 2

n\_75 2 2

n\_76 2 3

n\_77 2 2

n\_78 2 2

n\_79 2 2

n\_80 2 2

n\_81 2 2

n\_82 2 2

n\_83 3 3

n\_84 2 2

n\_85 2 2

n\_86 2 2

n\_87 2 3

n\_88 9 8

n\_89 2 2

n\_90 2 2

n\_91 7 7

n\_92 2 2

n\_93 2 2

n\_94 2 2

n\_95 2 2

n\_96 2 2

n\_97 2 2

n\_98 2 2

n\_99 2 2

n\_100 4 3

n\_101 2 2

n\_102 2 2

n\_103 2 2

n\_104 2 2

n\_105 2 2

n\_106 2 2

n\_107 2 2

n\_108 2 2

n\_110 2 2

n\_111 2 2

n\_112 2 2

n\_113 2 2

n\_114 2 2

n\_115 2 2

n\_116 2 2

$strataSize

Initial Final

0 4 30

1 302 219

2 124 111

3 104 78

4 42 83

5 453 399

6 464 340

7 183 155

8 359 259

9 358 270

10 669 448

11 189 163

12 81 65

13 215 181

14 138 103

15 26 58

16 271 230

17 9 43

18 60 83

19 195 166

20 59 106

21 39 85

22 95 117

23 55 63

24 75 101

25 103 135

26 21 64

27 76 70

28 642 663

29 638 495

30 7 36

31 248 250

32 141 105

33 189 152

34 22 38

35 63 91

36 53 65

37 189 161

38 62 67

39 1000 720

40 13 40

41 30 32

42 28 26

43 238 204

44 308 254

45 104 108

46 195 195

47 106 101

48 150 104

49 288 293

50 3 9

51 13 48

52 4 16

53 143 134

54 27 65

55 163 154

56 49 51

57 12 32

58 69 71

59 35 62

60 241 168

61 7 38

62 15 49

63 226 177

64 24 67

65 3 23

66 247 178

67 47 79

68 30 30

69 41 45

70 179 157

71 72 105

72 9 42

73 28 33

74 92 118

75 4 42

76 26 43

77 58 73

78 15 27

79 34 64

80 88 82

81 6 44

82 15 33

83 19 64

84 20 44

85 221 151

86 19 53

87 38 69

88 194 154

89 67 96

90 22 55

91 21 50

92 18 69

93 15 32

94 20 55

95 32 59

96 3 38

97 92 67

98 11 43

99 4 46

100 16 43

101 3 43

102 7 38

103 29 65

104 14 40

105 26 62

106 10 42

107 2 26

108 4 27

109 2 19

110 6 35

111 3 53

112 2 26

$runTime

user system elapsed

2.30 0.06 2.40

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

## Method 2 Results Based on Sample Size Vector From GA With K-means Initial Solution

$CVs

Initial Final Target

1 0.02109322 0.01362659 0.0002

$sampleSize

Initial Final

n\_1 2 2

n\_2 7 5

n\_3 5 5

n\_4 15 10

n\_5 2 2

n\_6 2 2

n\_7 7 15

n\_8 2 6

n\_9 4 4

n\_10 20 20

n\_11 44 31

n\_12 3 8

n\_13 9 12

$strataSize

Initial Final

0 101 103

1 131 116

2 144 135

3 111 83

4 149 169

5 142 148

6 13 17

7 49 61

8 79 94

9 62 52

10 83 60

11 44 70

12 14 14

$runTime

user system elapsed

0.07 0.00 0.31

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.03156651 0.009075392 0.0002

$sampleSize

Initial Final

n\_1 2 4

n\_2 5 13

n\_3 2 6

n\_4 21 11

n\_5 7 7

n\_6 8 3

n\_7 7 8

$strataSize

Initial Final

0 46 73

1 5 13

2 43 45

3 46 29

4 44 26

5 8 5

6 7 8

$runTime

user system elapsed

0.02 0.00 0.01

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.01960492 0.007362273 0.0002

$sampleSize

Initial Final

n\_1 17 17

n\_2 2 5

n\_3 7 14

n\_4 4 7

n\_5 9 24

n\_6 6 10

n\_7 4 9

n\_8 17 12

n\_9 2 7

n\_10 2 4

n\_11 2 6

n\_12 3 11

n\_13 2 6

n\_14 48 34

n\_15 6 20

n\_16 2 7

n\_17 7 28

n\_18 2 7

n\_19 142 44

n\_20 6 7

n\_21 3 7

n\_22 2 2

n\_23 3 5

n\_24 2 4

n\_25 3 4

n\_26 2 4

$strataSize

Initial Final

0 172 87

1 23 57

2 48 35

3 38 45

4 115 105

5 83 76

6 46 72

7 61 16

8 56 86

9 18 22

10 10 17

11 26 40

12 27 29

13 88 34

14 48 58

15 100 127

16 36 44

17 28 52

18 150 44

19 61 65

20 113 123

21 32 95

22 54 42

23 73 114

24 11 14

25 19 37

$runTime

user system elapsed

0.11 0.00 0.11

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.04184441 0.01732629 0.0002

$sampleSize

Initial Final

n\_1 6 5

n\_2 2 3

n\_3 2 3

n\_4 3 5

n\_5 7 10

n\_6 4 7

n\_7 46 25

n\_8 2 10

n\_9 2 6

$strataSize

Initial Final

0 109 87

1 106 141

2 101 104

3 118 111

4 72 69

5 61 71

6 83 25

7 11 43

8 9 19

$runTime

user system elapsed

0.04 0.00 0.05

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

$CVs

Initial Final Target

1 0.0198276 0.01799486 0.0002

$sampleSize

Initial Final

n\_1 4 6

n\_2 18 19

n\_3 7 7

n\_4 4 5

n\_5 10 10

n\_6 35 35

n\_7 11 12

n\_8 2 2

n\_9 25 24

n\_10 234 223

n\_11 3 4

n\_12 23 26

n\_13 3 3

n\_14 2 2

n\_15 2 2

n\_16 14 15

n\_17 10 10

n\_18 4 4

n\_19 2 3

n\_20 2 2

n\_21 2 2

n\_22 2 3

$strataSize

Initial Final

0 1291 1345

1 567 552

2 989 1007

3 352 411

4 596 607

5 1546 1467

6 981 984

7 1423 1482

8 581 549

9 1703 1415

10 1101 1138

11 575 579

12 30 34

13 37 46

14 60 87

15 195 193

16 102 100

17 76 92

18 41 58

19 112 135

20 84 112

21 9 58

$runTime

user system elapsed

0.23 0.00 0.23

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

## Method 2 Results Based on Total Sample Size From GA with K-means Initial Solution

$syscall

sa\_solution1\_b

$CVs

Initial Final Target

1 0.03951785 0.0137545 0.0002

$sampleSize

Initial Final

n\_1 10 3

n\_2 9 5

n\_3 10 5

n\_4 9 8

n\_5 10 2

n\_6 9 3

n\_7 9 13

n\_8 9 7

n\_9 10 5

n\_10 9 17

n\_11 9 36

n\_12 9 6

n\_13 10 12

$strataSize

Initial Final

0 101 109

1 131 109

2 144 124

3 111 66

4 149 172

5 142 153

6 13 16

7 49 72

8 79 106

9 62 54

10 83 64

11 44 63

12 14 14

$runTime

user system elapsed

0.06 0.01 0.07

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution2\_b)

$syscall

sa\_solution2\_b

$CVs

Initial Final Target

1 0.04207412 0.0003603264 0.0002

$sampleSize

Initial Final

n\_1 7 2

n\_2 7 6

n\_3 8 3

n\_4 8 18

n\_5 7 7

n\_6 8 7

n\_7 7 9

$strataSize

Initial Final

0 46 53

1 5 4

2 43 44

3 46 43

4 44 39

5 8 7

6 7 9

$runTime

user system elapsed

0.02 0.00 0.02

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution3\_b)

$syscall

sa\_solution3\_b

$CVs

Initial Final Target

1 0.07876066 0.009373916 0.0002

$sampleSize

Initial Final

n\_1 11 18

n\_2 12 6

n\_3 12 11

n\_4 12 10

n\_5 12 22

n\_6 11 13

n\_7 12 10

n\_8 12 11

n\_9 12 7

n\_10 12 8

n\_11 12 8

n\_12 11 15

n\_13 12 7

n\_14 12 29

n\_15 12 17

n\_16 12 8

n\_17 12 22

n\_18 11 4

n\_19 12 29

n\_20 11 12

n\_21 12 7

n\_22 12 8

n\_23 12 7

n\_24 12 5

n\_25 11 5

n\_26 11 6

$strataSize

Initial Final

0 172 94

1 23 51

2 48 33

3 38 56

4 115 107

5 83 13

6 46 76

7 61 11

8 56 74

9 18 37

10 10 33

11 26 39

12 27 34

13 88 43

14 48 49

15 100 117

16 36 46

17 28 92

18 150 29

19 61 75

20 113 142

21 32 8

22 54 71

23 73 119

24 11 16

25 19 71

$runTime

user system elapsed

0.11 0.00 0.11

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution4\_b)

$syscall

sa\_solution4\_b

$CVs

Initial Final Target

1 0.1001696 0.0168143 0.0002

$sampleSize

Initial Final

n\_1 8 7

n\_2 9 4

n\_3 8 3

n\_4 9 6

n\_5 8 11

n\_6 8 7

n\_7 8 13

n\_8 8 15

n\_9 8 8

$strataSize

Initial Final

0 109 95

1 106 130

2 101 98

3 118 119

4 72 68

5 61 78

6 83 13

7 11 48

8 9 21

$runTime

user system elapsed

0.05 0.00 0.05

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

> summary(sa\_solution5\_b)

$syscall

sa\_solution5\_b

$CVs

Initial Final Target

1 0.05723426 0.0210525 0.0002

$sampleSize

Initial Final

n\_1 19 5

n\_2 19 17

n\_3 19 11

n\_4 19 7

n\_5 19 10

n\_6 19 33

n\_7 19 10

n\_8 19 4

n\_9 19 23

n\_10 19 173

n\_11 19 9

n\_12 19 23

n\_13 19 11

n\_14 19 3

n\_15 19 7

n\_16 19 21

n\_17 19 21

n\_18 20 7

n\_19 19 8

n\_20 19 7

n\_21 19 5

n\_22 19 4

$strataSize

Initial Final

0 1291 1345

1 567 574

2 989 967

3 352 408

4 596 635

5 1546 1425

6 981 978

7 1423 1473

8 581 559

9 1703 1292

10 1101 1123

11 575 572

12 30 42

13 37 56

14 60 124

15 195 198

16 102 108

17 76 119

18 41 67

19 112 173

20 84 147

21 9 66

$runTime

user system elapsed

0.25 0.00 0.25

$control

NULL

attr(,"class")

[1] "summary.saAlloc"

## Method 3 Scripts

library(stratification)

source("<path to the source directory>/selectDomain.R")

LH\_solutions <- list()

for (i in 1:5) {

domain <-PSU\_domain(PSU.GA,i)

X <- as.vector(as.numeric(domain$value))

CV <- as.vector(CVConst$CV1)

LH\_solutions[[i]]<-strata.LH(X,

CV= 0.02,

alloc = list(q1 = 0.5, q2 =0, q3 = 0.5),

model='none',

takenone = 0,

Ls=4

)

}

print(LH\_solutions)

LH\_bh <- vector()

LH\_Nh <- vector()

LH\_nh <- vector()

LH\_Label <- vector()

LH\_var <- vector()

LH\_mean <- vector()

for (i in 1:length(LH\_solutions)){

LH\_bh <- c(LH\_bh, LH\_solutions[[i]]$bh)

LH\_Nh <- c(LH\_Nh, LH\_solutions[[i]]$Nh)

LH\_nh <- c(LH\_nh, LH\_solutions[[i]]$nh)

LH\_Label <- c(LH\_Label, LH\_solutions[[i]]$stratumID)

LH\_var <- c(LH\_var, LH\_solutions[[i]]$varh)

LH\_mean <- c(LH\_mean, LH\_solutions[[i]]$meanh)

}

LH\_aggregatestrata <- data.frame(matrix(NA, nrow =20, ncol = 8))

colnames(LH\_aggregatestrata)= colnames(SA\_aggregatestrata)

LH\_aggregatestrata$SOLUZ <-LH\_nh

LH\_aggregatestrata$N <- LH\_Nh

LH\_aggregatestrata$COST <- c(1)

LH\_aggregatestrata$CENS <-c(0)

LH\_aggregatestrata$STRATO <- c(1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4)

LH\_aggregatestrata$DOM1 <- c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5)

LH\_aggregatestrata$M1 <- LH\_mean

LH\_aggregatestrata$S1 <- sqrt(LH\_var)

Framenew\_CFS\_LH\_1<- PSU.GA[,7:12]

Framenew\_CFS\_LH <- Framenew\_CFS\_LH\_1[

with(Framenew\_CFS\_LH\_1, order(Framenew\_CFS\_LH\_1$DOM1)),

]

Framenew\_CFS\_LH$ID<-orderd\_CFSFrameData$estno

Framenew\_CFS\_LH$LABEL <-LH\_Label

Framenew\_CFS\_LH <- Framenew\_CFS\_LH[, c(1,5,7,2,3,6,4)]

colnames(Framenew\_CFS\_LH)<-colnames(framenew\_CFS)

sample\_CFS\_LH <- selectSample(Framenew\_CFS\_LH, LH\_aggregatestrata, writeFiles=TRUE, verbatim = TRUE)

source("C:/Users/sa129715/Documents/R/CFS\_PROJECT/myeval.R")

GA\_totalsamplesize <-sum(ga\_solution$aggr\_strata\_subset\_SOLUZ$SOLUZ)

eval\_LH<- myEval(Framenew\_CFS\_LH,

LH\_aggregatestrata,

nsampl=GA\_553,

writeFiles=TRUE,

progress=FALSE)

eval\_LH$coeff\_var

## Method 3 Results

> print(LH\_solutions)

[[1]]

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 100, maxstill = 500, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 275.5 138.0 6302.00 275 12 0.04

stratum 2 | take-some 1 | 557.5 416.5 6626.92 282 13 0.05

stratum 3 | take-some 1 | 840.5 699.0 6674.00 283 13 0.05

stratum 4 | take-some 1 | 1123.0 981.5 6626.92 282 13 0.05

Total 1122 51 0.05

Total sample size: 51

Anticipated population mean: 561.5

Anticipated CV: 0.01973292

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

[[2]]

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 20, maxstill = 200, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 51.5 26.0 216.67 51 11 0.22

stratum 2 | take-some 1 | 101.5 76.5 208.25 50 11 0.22

stratum 3 | take-some 1 | 150.5 126.0 200.00 49 10 0.20

stratum 4 | take-some 1 | 200.0 175.0 200.00 49 10 0.20

Total 199 42 0.21

Total sample size: 42

Anticipated population mean: 100

Anticipated CV: 0.01969047

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

[[3]]

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 100, maxstill = 500, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 388.5 194.5 12545.25 388 13 0.03

stratum 2 | take-some 1 | 775.5 582.0 12480.67 387 13 0.03

stratum 3 | take-some 1 | 1150.5 963.0 11718.67 375 12 0.03

stratum 4 | take-some 1 | 1537.0 1343.5 12416.25 386 13 0.03

Total 1536 51 0.03

Total sample size: 51

Anticipated population mean: 768.5

Anticipated CV: 0.01986425

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

[[4]]

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 67, maxstill = 500, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 167.5 84 2324 167 12 0.07

stratum 2 | take-some 1 | 334.5 251 2324 167 12 0.07

stratum 3 | take-some 1 | 501.5 418 2324 167 12 0.07

stratum 4 | take-some 1 | 671.0 586 2380 169 13 0.08

Total 670 49 0.07

Total sample size: 49

Anticipated population mean: 335.5

Anticipated CV: 0.01982832

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

[[5]]

Given arguments:

x = X

CV = 0.02, Ls = 4, takenone = 0, takeall = 0

allocation: q1 = 0.5, q2 = 0, q3 = 0.5

model = none

algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,

maxstep = 100, maxstill = 500, rep = 5, trymany = TRUE

Strata information:

| type rh | bh E(Y) Var(Y) Nh nh fh

stratum 1 | take-some 1 | 3096.5 1548.59 800140.3 3113 13 0

stratum 2 | take-some 1 | 6198.5 4646.11 802525.1 3124 14 0

stratum 3 | take-some 1 | 9297.5 7749.60 800891.1 3111 13 0

stratum 4 | take-some 1 | 12396.0 10846.59 800183.0 3103 13 0

Total 12451 53 0

Total sample size: 53

Anticipated population mean: 6192.37

Anticipated CV: 0.01981836

Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.

## Related formulae

The final CV from each method is the ratio of the total target variable in (1) over Mean square error (MSE) is calculated in (2). Nh is the population of each stratum h, is the mean of target variable in each stratum h. is the number of samplesize from each stratum h. is the standard deviation of stratum h. The expected CV of GA method, and anticipated CV of Generalized Lavallee-Hidiroglou Method are both computed based on (3).

|  |  |  |
| --- | --- | --- |
| Estimator for the total mean of target variable Y |  | (1) |
| MSE () |  | (2) |
| Coefficient of variation (CV) |  | (3) |
| RMSE |  | (4) |

# References

Al-Kateb, M., & Lee, B. S. (2014). Adaptive stratified reservoir sampling over heterogeneous data streams. *Information Systems*, *39*, 199–216. https://doi.org/10.1016/j.is.2012.03.005

Auxiliary Variable. (2008). In P. Lavrakas, *Encyclopedia of Survey Research Methods*. Sage Publications, Inc. https://doi.org/10.4135/9781412963947.n31

Ballin, M., & Barcaroli, G. (2013). Joint determination of optimal stratification and sample allocation using genetic algorithm. *Survey Methodology*, 27.

Barcaroli, G. (2014a). SamplingStrata: An *R* Package for the Optimization of Stratified Sampling. *Journal of Statistical Software*, *61*(4). https://doi.org/10.18637/jss.v061.i04

Barcaroli, G. (2014b). SamplingStrata: An R Package for the Optimization of Stratified Sampling. *Journal of Statistical Software*, *61*(1), 1–24. https://doi.org/10.18637/jss.v061.i04

Barcaroli, G. (2019a, April 21). *R package for Optimal Stratification of Sampling Frames for Multipurpose Sampling Surveys: Barcaroli/SamplingStrata*. https://github.com/barcaroli/SamplingStrata

Barcaroli, G. (2019b, May 2). *optimizeStrata: Best stratification of a sampling frame for multipurpose... in SamplingStrata: Optimal Stratification of Sampling Frames for Multipurpose Sampling Surveys*. https://rdrr.io/cran/SamplingStrata/man/optimizeStrata.html

Basoglu, ismail. (2014). *Ismail\_basoglu\_phd\_thesis.pdf* [Bogazici University]. http://www.ie.boun.edu.tr/~hormannw/BounQuantitiveFinance/Thesis/ismail\_basoglu\_phd\_thesis.pdf

Bendeti, R., Bee, M., & Espa, G. (2010). *A Framework for Cut-off Sampling in Business Survey Design*. *26*(4), 651–671.

Benedetti, R., & Piersimoni, F. (2012a). *Multivariate Boundaries of a Self Representing Stratum of Large Units in Agricultural Survey Design*. *6*, 125–135.

Benedetti, R., & Piersimoni, F. (2012b). Multivariate Boundaries of a Self Representing Stratum of Large Units in Agricultural Survey Design. *Survey Research Methods*, *6*(3), 125–135. https://doi.org/10.18148/srm/2012.v6i3.5127

Bethel, J. (1989). *Sample allocation in multivariate surveys*. *15*, 47–57.

Bureau of Transportation Statistics. (2019a, February 25). *2017 Commodity Flow Survey Overview and Methodology*. https://www.bts.gov/archive/publications/commodity\_flow\_survey/methodology\_2012

Bureau of Transportation Statistics. (2019b, March 29). *Freight Analysis Framework*. https://www.bts.gov/faf

Carpentier, A., & Munos, R. (n.d.). *Toward Optimal Stratification for Stratified Monte-Carlo Integration*. 9.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51. https://doi.org/10.1007/BF00940812

Etore, P., Fort, G., Jourdain, B., & Moulines, E. (2011). On adaptive stratification. *Annals of Operations Research*, *189*(1), 127–154. https://doi.org/10.1007/s10479-009-0638-9

Etore, P., & Jourdain, B. (2010). Adaptive optimal allocation in stratified sampling methods. *Methodology and Computing in Applied Probability*, *12*(3), 335–360. https://doi.org/10.1007/s11009-008-9108-0

Ghanbartehrani, S. (2019, April 10). *Saeedt/CFS\_Sampling*. GitHub. https://github.com/saeedt/CFS\_Sampling

Glasserman, P., Heidelberger, P., & Shahabuddin, P. (1999). Asymptotically Optimal Importance Sampling and Stratification for Pricing Path-Dependent Options. *Mathematical Finance*, *9*(2), 117–152. https://doi.org/10.1111/1467-9965.00065

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics*, *28*(1), 100. https://doi.org/10.2307/2346830

Hartigan, J., & Wong, M. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100. https://doi.org/10.2307/2346830

Kawai, R. (2010). Asymptotically optimal allocation of stratified sampling with adaptive variance reduction by strata. *ACM Transactions on Modeling and Computer Simulation*, *20*(2), 1–17. https://doi.org/10.1145/1734222.1734225

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Lavallee, P., & Hidiroglou, M. A. (1988). *On the Stratification of Skewed Populations*. 12.

Lermusiaux, P. F. J. (2007). Adaptive modeling, adaptive data assimilation and adaptive sampling. *Physica D: Nonlinear Phenomena*, *230*(1–2), 172–196. https://doi.org/10.1016/j.physd.2007.02.014

Lisic, J., Sang, H., Zhu, Z., & Zimmer, S. (2018a). Optimal Stratification and Allocation for the June Agricultural Survey. *Journal of Official Statistics*, *34*(1), 121–148. https://doi.org/10.1515/jos-2018-0007

Lisic, J., Sang, H., Zhu, Z., & Zimmer, S. (2018b). Optimal Stratification and Allocation for the June Agricultural Survey. *Journal of Official Statistics*, *34*(1), 121–148. https://doi.org/10.1515/jos-2018-0007

Manly, B. F. J. (2004). Using the bootstrap with two-phase adaptive stratified samples from multiple populations at multiple locations. *Environmental & Ecological Statistics*, *11*(4), 367–383.

National Agricultural Statistics Service. (2018, November 9). *USDA - National Agricultural Statistics Service—Surveys—June Area*. https://www.nass.usda.gov/Surveys/Guide\_to\_NASS\_Surveys/June\_Area/

R Consortium. (2014, March 27). *RStudio*. RStudio. https://www.rstudio.com/

Rivest, L.-P. (2017). *Cran/stratification* [R]. cran. https://github.com/cran/stratification (Original work published 2014)

Statistics Dictionary. (2019a). *Neyman Allocation: Definition*. https://stattrek.com/statistics/dictionary.aspx?definition=neyman\_allocation

Statistics Dictionary. (2019b). *Optimum Allocation: Definition*. https://stattrek.com/statistics/dictionary.aspx?definition=optimum\_allocation

Steven P. Millard. (2019, May 2). *LognormalTrunc: The Truncated Lognormal Distribution in EnvStats: Package for Environmental Statistics, Including US EPA Guidance*. https://rdrr.io/cran/EnvStats/man/LognormalTrunc.html

*The Comprehensive R Archive Network*. (2019, March 11). https://cran.r-project.org/

The PostgreSQL Global Development Group. (2019a). *PostgreSQL: Documentation: 11: 43.1. Overview*. https://www.postgresql.org/docs/current/plpgsql-overview.html

The PostgreSQL Global Development Group. (2019b). *PostgreSQL: The world’s most advanced open source database*. https://www.postgresql.org/

The R Foundation. (n.d.). *Building R for Windows*. Retrieved August 6, 2019, from https://cran.r-project.org/bin/windows/Rtools/

The R Foundation. (2019). *R: The R Project for Statistical Computing*. https://www.r-project.org/

Thompson, S. K. (2012). *Sampling* (3rd ed). Wiley.

US Census Bureau. (2018, September 12). *County Business Patterns: 2016*. https://www.census.gov/data/datasets/2016/econ/cbp/2016-cbp.html

US Census Bureau. (2018, December 10). *About this Survey*. The United States Census Bureau. https://www.census.gov/programs-surveys/cfs/about.html

Yong, F. H., Tian, L., Yu, S., Cai, T., & Wei, L. J. (2016). Optimal stratification in outcome prediction using baseline information. *Biometrika*, *103*(4), 817–828. https://doi.org/10.1093/biomet/asw049