**Transforming Freight Flow Data Collection**

Milestones #2 and #3

Adaptive Sampling Literature Review and Proposed Methodology

Optimization Model Design and Preliminary Results

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

Ohio University

Athens, OH

Rev. 5, Nov 2019

Contents

[1. Introduction 4](#_Toc24491017)

[2. Stratified Sampling 4](#_Toc24491018)

[3. Adaptive Sampling 5](#_Toc24491019)

[4. Related Literature 6](#_Toc24491020)

[4.1. Stratification 6](#_Toc24491021)

[4.2. Allocation 9](#_Toc24491022)

[4.3. Joint Stratification-Allocation 10](#_Toc24491023)

[5. Proposed Methodology 11](#_Toc24491024)

[6. Comparison of the Candidate Methodologies 13](#_Toc24491025)

[7. Case Study 14](#_Toc24491026)

[7.1. Data sources 15](#_Toc24491027)

[7.2. Pre-processing of the data 15](#_Toc24491028)

[7.3. Frame data generation 16](#_Toc24491029)

[8. Method 1: Optimal stratification and allocation based on Genetic Algorithm 17](#_Toc24491030)

[8.1. Method 1 procedure 19](#_Toc24491031)

[8.2. Method 1 Results 25](#_Toc24491032)

[9. Method 2: Optimal stratification and allocation using Genetic Algorithm and Simulated Annealing 30](#_Toc24491033)

[9.1. Method 2 procedure 31](#_Toc24491034)

[9.2. Method 2 results 36](#_Toc24491035)

[9.3. Comparing the result of GA and SA 37](#_Toc24491036)

[10. Method3: Generalized Lavallee-Hidiroglou Method for Strata Construction 39](#_Toc24491037)

[10.1. Method 3 results 41](#_Toc24491038)

[11. Appendix 43](#_Toc24491039)

[11.1. Method 1 Scripts 43](#_Toc24491040)

[11.2. Method 1 summary of the results with random initial solution (no K-means) 44](#_Toc24491041)

[11.3. Method 1 summary of the results with K-means initial solution 46](#_Toc24491042)

[11.4. Method 2 Scripts 50](#_Toc24491043)

[11.5. Method 2 results based on GA with random initial solution (no K-means) 53](#_Toc24491044)

[11.6. Method 2 results based on GA with K-means initial solution 62](#_Toc24491045)

[12. Appendix 67](#_Toc24491046)

[13. References 68](#_Toc24491047)

# Introduction

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.

# 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.

# 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). 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).

# Related Literature

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

## 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.

## 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.

# 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. Both methods have readily available code implemented in R applied to case studies involving real surveys.

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 implemented 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) the number of strata and also the total sample size are assumed to be fixed. In Barcaroli's (2014) model, 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 Lisic et al.’s methodology 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 Barcaroli’s 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 models take the same approach in defining the constraints. Constraints are on CV and budget. In both methodologies, K-means clustering algorithm is used in the stratification stage and Bethel's (1989) method is employed in the allocation framework.

In Lisic’s et al. method, simulated annealing algorithm is used for optimizing the objective function. The algorithm stops after a given number of iteration or when a threshold is met. Although Barcaroli used genetic algorithm as the solution method, the algorithm terminates after a specific number of iterations or when the value of the objective function reaches a given minimum. Both models rely on historic data as their administrative data variables along with other source of data.

# 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 ‎7.3.

## 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 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. Finally, 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). 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, state, 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 ‎7.2) 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 is a better choice to simulate the skewness of the population. Values generated from normal distribution in generate\_est() function can be easily converted to Log-normal by applying exp() function. 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.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, 2014b) 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.

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

Table 3 An example of CENS binary vector

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.

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

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

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 last 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).

framenew\_CFS <- updateFrame(CFSFrame,

newstrata,

writeFiles=FALSE)

sample\_CFS <- selectSample(framenew\_CFS,

ga\_solution$aggr\_strata,

writeFiles=TRUE,

verbatim = TRUE)

## 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 1. Results for all domains are reported in section ‎11.2. 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 1. 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 2. Results for all domains are reported in the in section ‎11.3.

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 2 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 1018 while the random initial solution resulted in the sample size of 2015. Therefore, the K-means initial solution results in 50% reduction in sample size while satisfying the same CV constraints.

# 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 variance and fixed sample size constraints. This method minimizes sum of penalties of deviations from the target CVs as a proxy for quality constraints.

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 (a stratification and allocation that meets the CV constraints) 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, combining the optimal stratification and allocation based on GA and the SA methods is expected to result in a higher qulity solution compared to each method individually.

The method discussed in this section uses the GA based method to find a feasible solution that meets the CV constraints with 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. 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 ‎8.1), “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. Note that Rtools is a standalone application and not an add-on for R, so the installation process involves download and execution of a setup file.

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’ tables were extracted from the GA result. 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.

To simplify ‘newstrata’ table, necessary columns including atomic strata, number of atomic strata, domain, and stratification labels are selected and stored in newstrata1 frame for further use in creating the primary sampling unit (PSU) list used in SA method.

ga\_solution$aggr\_strata

newstrata <- updateStrata(CFSstrata,

ga\_solution,

writeFiles = TRUE)

newstrata1 <- newstrata[,c(1,2,7,11)]

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

Stratification.GA <- newstrata1

j = 1

for (i in 1:nrow(newstrata1)) {

num = newstrata1[i,2]

if (num == 1) {

Stratification.GA[j,] = newstrata1[i,]

j = j + 1

} else {

for (k in 1:num) {

Stratification.GA[j,-2] = newstrata1[i,-2]

Stratification.GA[j,2] = 1

j = j + 1

}

}

}

View(Stratification.GA)

# Add target values from CFSFrameData to the result of stratification\_GA

PSU.GA<- cbind.data.frame(Stratification.GA[,1:4],value = CFSFrameData$value)

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 scripts are 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. The values of PSU.GA[i,3] and ga\_soulution$agg\_strata[i,5] in the if statements are used to select the results for the domain i. For example, to specify the GA result for domain 1, ‘PSU.GA[i,3] ==1’ and ‘ga\_soulution$agg\_strata[i,5] ==1’ are used as conditions in the two if statements in the code below. All strata with one PSU are excluded from all domains. The output includes matrix x1 (domain 1 PSUs), “labeldom1” (initial stratification from the GA method), and “sampleSizeMultiDOM1” (the optimal sample sizes based on GA result for domain1).

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

S = matrix(0,nrow=1, ncol=5)

SS = matrix(0,nrow=1, ncol=5)

for (i in 1:nrow(PSU.GA)) {

if (PSU.GA[i,3] == 1) {

S[1,] = as.matrix(PSU.GA[i,])

SS = rbind(SS,S)

}

}

strata.domain1 = SS[-1,]

# Remove PSU with one sampling unit

strata\_dom1=data.frame(strata.domain1)

colnames(strata\_dom1)= c("STRATO", "N", "DOM1", "LABEL","value")

strata\_dom1$LABEL <- as.numeric(as.character(strata\_dom1$LABEL))

r= as.data.frame(which(table(strata\_dom1$LABEL)== 1, arr.ind = TRUE))

strata\_dom1<- strata\_dom1[!(strata\_dom1$LABEL %in% r$dim1),]

labeldom1=as.numeric(strata\_dom1[,4]) # Initial stratification to be used in

x1 <- as.matrix(as.double(matrix(strata\_dom1[,5]))) # Matrix xi

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

R = matrix(0,nrow=1, ncol=8)

RR = matrix(0,nrow=1, ncol=8)

for (i in 1:nrow(ga\_solution$aggr\_strata)) {

if (ga\_solution$aggr\_strata[i,5] == 1) {

R[1,] = as.matrix(ga\_solution$aggr\_strata[i,])

RR = rbind(RR,R)

}

}

samplesize.domain1 = RR[-1,]

sampleSizeMultiDOM1=sum(ceiling(samplesize.domain1[,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 (the value of target CV), and penalty (of deviating from target CVs) are specified along with the input matrices which are xi (PSUs in domain i), label (initial stratification), samplesize (required sample size for each domain). We used the GA method output for x1, labeldom1, and sampleSizeMultiDOM1 here. TargetCV, iteration, and penalty parameters are set to 0.02, 1000, and 10 respectively.

library(saAlloc)

sa\_solution\_acv\_multi\_1 <- saMinCV(

x=x1,

label= labeldom1,

targetCV=(0.02),

sampleSize=sampleSizeMultiDOM1,

iterations=1000,

penalty = 10,

preserveSatisfied=TRUE,

fpc=FALSE)

summary(sa\_solution\_acv\_multi\_1)

sa\_solution\_acv\_multi\_1$label

Although the results for each domain are stored in ‘sa\_solution’, selected elements can be stored in separate csv files for convenience and further analysis. The elements in ‘sa\_solution’ are as follows: “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” while in “StrataSize”, the number of distinct strata in the sample is shown. To display the stratification labels vector, “sa\_solution\_acv\_multi\_1$label” is called.

After running the SA based method using the same sample size and initial stratification from the GA method, CV in the final sample is improved. The result of GA and SA method for joint stratification and allocation are displayed by running the following script.

# comparing result of GA and SA

SA\_GA\_compare\_startification <- cbind.data.frame(SA\_stratification=sa\_solution\_acv\_multi\_1$label,GA\_stratification=c(labeldom1))

SA\_GA\_compare\_allocation <- cbind.data.frame(SA\_allocation=sa\_solution\_acv\_multi\_1$sampleSize,GA\_allocation=ceiling(samplesize.domain1[,8])

## Method 2 results

Summary of the results for two domains (i.e. domains 1 and 2) are presented in this section. Number of iterations was set to 1,000 and target CV was set to 0.02. $CVs indicates the Initial CV (i.e. prior to optimization using the SA based method) and the final CV.

sa\_solution\_acv\_multi\_dom1

$CVs

Initial Final Target

1 0.1500073 0.07733092 0.02

sa\_solution\_acv\_multi\_dom2

$CVs

Initial Final Target

1 0.6194158 0.1589023 0.02

$samplesize table indicates the initial and final sample sizes for each stratum.

Total sample size\_dom2= 350

Total sample size\_dom5= 620

“$stratasize” compares the initial and final number of strata in the sample which is provided in section ‎11.5.

Similar to the method 1, the GA 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.

sa\_solution\_acv\_multi\_dom1

$CVs

Initial Final Target

1 0.5474734 0.08412992 0.02

sa\_solution\_acv\_multi\_dom2

$CVs

Initial Final Target

1 0.1050398 0.07348977 0.02

The initial and final sample sizes based on the GA with K-means initial solution are as follows.

Total sample size\_dom1= 47

Total sample size\_dom2= 434

## Comparing the result of GA and SA

The comparison of the stratification based on GA and SA methods on domain 2 is presented in Table 9 (Only 13 rows are shown).

Table 9. SA and GA stratifications

|  |  |  |
| --- | --- | --- |
|  | SA\_stratification | GA\_stratification |
| 1 | 1 | 1 |
| 2 | 9 | 1 |
| 3 | 15 | 1 |
| 4 | 2 | 2 |
| 5 | 2 | 2 |
| 6 | 12 | 3 |
| 7 | 15 | 3 |
| 8 | 7 | 1 |
| 9 | 1 | 1 |
| 10 | 15 | 1 |
| 11 | 1 | 1 |
| 12 | 1 | 1 |

Sampling based on SA and GA are compared in Table 10.

Table 10. SA and GA allocations

|  |  |  |
| --- | --- | --- |
|  | SA\_allocation | GA\_allocation |
| n\_1 | 14 | 2 |
| n\_2 | 10 | 7 |
| n\_3 | 12 | 5 |
| n\_4 | 21 | 14 |
| n\_5 | 8 | 2 |
| n\_6 | 7 | 2 |
| n\_7 | 2 | 7 |
| n\_8 | 2 | 2 |
| n\_9 | 17 | 4 |
| n\_10 | 3 | 17 |
| n\_11 | 9 | 34 |
| n\_12 | 3 | 3 |
| n\_13 | 2 | 2 |
| n\_14 | 2 | 8 |
| n\_15 | 4 | 7 |

# Method3: Generalized Lavallee-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. Lavallee-Hidiroglou (LH) algorithm is used to determine the optimal stratum boundaries that minimize the variance of the mean estimate for a given total number of strata. Neyman allocation was used to determine the sample size for each 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 Lavallee-Hidiroglou (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 “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 (Rivest, 2014/2017).

In this section, the optimal strata boundaries and the optimal sample size are determined in one step using generalized Lavallee-Hidiroglou (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 of the package is ‘strata.LH” function 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 two parameters of CV (target coefficient of variation) or target sample size (n) need to be provided as input. 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, 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 3. The method fail to reach the minimum sample size when using large number of sample size (Ls greater than 20) .The following script is for domain 1 and domain 2.

source("C:/Users/sa129715/Documents/R/function3/selectDomain.R")

domain1<- selectDomain(CFSFrameData,1)

domain2<- selectDomain(CFSFrameData,2)

library(stratification)

#-----domain1---------------

X1= as.vector(domain1[,5])

LH1 <-strata.LH(X1,

CV= 0.05,

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

model='none',

takenone = 0,

Ls=3

)

print(LH1)

X12=as.vector(domain2[,5])

LH2 <-strata.LH(X12,

CV= 0.05,

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

model='none',

takenone = 0,

Ls=3

)

print(LH2)

## Method 3 results

In this method, the number of sampled strata (Ls) was set to the default value which is 3. 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.

Given arguments:

x = x11

CV = 0.05, Ls = 3, 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 | 524.85 141.34 15539.39 835 23 0.03

stratum 2 | take-some 1 | 2936.21 1213.09 370918.62 245 33 0.13

stratum 3 | take-some 1 | 12838.80 6746.93 11939803.35 42 32 0.76

Total 1122 88 0.08

Total sample size: 88

Anticipated population mean: 622.6382

Anticipated CV: 0.04955742

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

Given arguments:

x = X2

CV = 0.05, Ls = 3, 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 | 201.88 42.22 1839.73 120 3 0.02

stratum 2 | take-some 1 | 1174.99 517.19 55034.86 62 9 0.15

stratum 3 | take-all 1 | 12168.70 4454.02 14241912.57 17 17 1.00

Total 199 29 0.15

Total sample size: 29

Anticipated population mean: 567.0858

Anticipated CV: 0.04747531

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

# 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

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=",")

## 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 the 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-----------------------------------------

# Adding labels to strata result of GA

newstrata <- updateStrata(AtomicStrata,

ga\_solution,

writeFiles = TRUE)

newstrata1 <- newstrata[,c(1,2,7,11)]

# Data manipulation to be used in SA: Create PSU from stratification result of GA

Stratification.GA = newstrata1

j = 1

for (i in 1:nrow(newstrata1)) {

num = newstrata1[i,2]

if (num == 1) {

Stratification.GA[j,] = newstrata1[i,]

j = j + 1

}

else {

for (k in 1:num) {

Stratification.GA[j,-2] = newstrata1[i,-2]

Stratification.GA[j,2] = 1

j = j + 1

}

}

}

View(Stratification.GA)

PSU.GA<- cbind.data.frame(Stratification.GA[,1:4],value = CFSFrameData$value)

# -------------Domain1-------------------------------------------

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

S = matrix(0,nrow=1, ncol=5)

SS = matrix(0,nrow=1, ncol=5)

for (i in 1:nrow(PSU.GA)) {

if (PSU.GA[i,3] == 1) {

S[1,] = as.matrix(PSU.GA[i,])

SS = rbind(SS,S)

}

}

strata.domain1 = SS[-1,]

# remove SPU with one sampling unit

strata\_dom1=data.frame(strata.domain1)

colnames(strata\_dom1)= c("STRATO", "N", "DOM1", "LABEL","value")

strata\_dom1$LABEL <- as.numeric(as.character(strata\_dom1$LABEL))

r= as.data.frame(which(table(strata\_dom1$LABEL)== 1, arr.ind = TRUE))

strata\_dom1<- strata\_dom1[!(strata\_dom1$LABEL %in% r$dim1),]

#--------------------------------------

labeldom1=as.numeric(strata\_dom1[,4]) # initial stratification to be used in simulated anealing

x1 <- as.matrix(as.double(matrix(strata\_dom1[,5]))) # Matrix xi

# using ga\_solution$aggr\_strata result for creating parameter samplesizeMultiDOM1

R = matrix(0,nrow=1, ncol=8)

RR = matrix(0,nrow=1, ncol=8)

for (i in 1:nrow(ga\_solution$aggr\_strata)) {

if (ga\_solution$aggr\_strata[i,5] == 1) {

R[1,] = as.matrix(ga\_solution$aggr\_strata[i,])

RR = rbind(RR,R)

}

}

samplesize.domain1 = RR[-1,]

sampleSizeMultiDOM1=sum(ceiling(samplesize.domain1[,8]))-----------------------------------------------------------------------------------

# Optimal allocation stratification ("saAlloc") package

library(saAlloc)

sa\_solution\_acv\_multi\_1 <- saMinCV(

x=x1 ,

label= labeldom1,

# label =kMeansCluster1,

targetCV=(0.02),

sampleSize=sampleSizeMultiDOM1,

terations=100,

penalty = penalty2,

preserveSatisfied=TRUE,

fpc=FALSE

)

summary(sa\_solution\_acv\_multi\_1)

## Method 2 results based on GA with random initial solution (no K-means)

sa\_solution\_acv\_multi\_1

$CVs

Initial Final Target

1 0.1500073 0.07733092 0.02

$sampleSize

Initial Final

n\_2 6 2

n\_3 6 2

n\_4 6 3

n\_5 7 3

n\_6 6 8

n\_7 6 2

n\_8 6 5

n\_9 6 3

n\_10 6 8

n\_11 6 9

n\_12 6 2

n\_13 6 2

n\_14 7 7

n\_15 6 4

n\_16 6 9

n\_17 6 12

n\_18 7 7

n\_19 6 5

n\_20 7 2

n\_21 6 2

n\_22 6 2

n\_23 6 10

n\_24 6 4

n\_25 7 2

n\_26 7 2

n\_27 6 2

n\_28 7 8

n\_29 6 3

n\_30 6 2

n\_31 6 11

n\_32 6 8

n\_33 6 7

n\_34 6 2

n\_35 7 3

n\_36 6 5

n\_37 6 4

n\_38 6 5

n\_39 6 32

n\_40 6 7

n\_41 6 11

n\_42 6 6

n\_43 7 20

n\_44 6 7

n\_45 6 4

n\_46 6 2

n\_47 7 28

n\_48 6 2

n\_49 6 10

n\_50 6 18

n\_51 7 2

n\_53 6 6

n\_54 6 4

n\_55 7 3

n\_56 6 4

n\_57 7 2

n\_59 7 5

$strataSize

Initial Final

0 4 13

1 16 30

2 2 6

3 20 22

4 18 8

5 3 20

6 8 5

7 25 31

8 15 14

9 19 24

10 16 19

11 8 29

12 37 28

13 20 22

14 28 27

15 18 19

16 15 14

17 16 12

18 10 17

19 16 25

20 11 20

21 15 10

22 11 14

23 11 12

24 10 24

25 10 15

26 19 21

27 17 21

28 22 25

29 29 17

30 60 41

31 11 7

32 20 27

33 26 26

34 14 19

35 15 20

36 20 16

37 54 32

38 28 14

39 32 25

40 28 21

41 35 20

42 5 11

43 13 25

44 8 16

45 54 35

46 19 17

47 13 10

48 71 44

49 14 16

50 33 23

51 9 8

52 9 19

53 14 19

54 28 27

55 17 17

sa\_solution\_acv\_multi\_2

$CVs

Initial Final Target

1 0.6194158 0.1589023 0.02

$sampleSize

Initial Final

n\_1 10 8

n\_2 11 9

n\_3 10 11

n\_4 10 10

n\_5 10 15

n\_6 11 9

$strataSize

Initial Final

0 22 69

1 21 30

2 100 11

3 21 23

4 11 15

5 24 51

sa\_solution\_acv\_multi\_3

$CVs

Initial Final Target

1 0.1905244 0.1140954 0.02

$sampleSize

Initial Final

n\_1 3 2

n\_2 3 2

n\_3 2 3

n\_4 2 2

n\_5 2 2

n\_6 3 2

n\_7 3 2

n\_8 3 2

n\_9 3 2

n\_10 2 2

n\_11 3 2

n\_12 2 2

n\_13 2 4

n\_14 3 2

n\_15 3 6

n\_16 3 9

n\_17 2 2

n\_18 2 2

n\_19 3 2

n\_20 3 2

n\_21 2 2

n\_22 3 2

n\_23 2 2

n\_24 3 2

$strataSize

Initial Final

0 3 26

1 17 40

2 129 92

3 101 69

4 27 49

5 50 59

6 33 43

7 76 65

8 58 64

9 26 45

10 88 74

11 48 42

12 100 92

13 36 52

14 88 65

15 146 121

16 18 33

17 50 74

18 129 98

19 65 70

20 53 53

21 61 62

22 54 72

23 80 76

$runTime

user system elapsed

0.11 0.00 0.11

sa\_solution\_acv\_multi\_4

$CVs

Initial Final Target

1 0.2375809 0.09156403 0.02

$sampleSize

Initial Final

n\_1 18 16

n\_2 17 5

n\_3 17 11

n\_4 17 69

n\_5 18 10

n\_6 18 21

n\_7 17 7

n\_8 17 11

n\_9 18 16

n\_10 17 8

$strataSize

Initial Final

0 14 44

1 16 64

2 11 22

3 361 155

4 80 93

5 88 90

6 38 95

7 12 64

8 42 31

9 8 12

sa\_solution\_acv\_multi\_5

$CVs

Initial Final Target

1 0.108169 0.04806214 0.02

$sampleSize

Initial Final

n\_1 8 2

n\_2 8 29

n\_3 8 18

n\_4 8 2

n\_5 8 2

n\_6 9 29

n\_7 8 35

n\_8 8 10

n\_9 8 24

n\_10 8 19

n\_11 8 47

n\_12 8 6

n\_13 8 6

n\_14 8 11

n\_15 8 6

n\_16 8 2

n\_17 9 30

n\_18 8 2

n\_19 8 3

n\_20 8 9

n\_21 8 4

n\_22 8 10

n\_23 8 10

n\_24 8 4

n\_25 8 3

n\_26 8 14

n\_27 8 2

n\_28 8 3

n\_29 8 35

n\_30 9 48

n\_31 9 2

n\_32 8 7

n\_33 8 15

n\_34 8 9

n\_35 8 5

n\_36 9 14

n\_37 8 6

n\_38 8 24

n\_39 9 8

n\_40 8 46

n\_41 8 3

n\_42 8 2

n\_43 8 6

n\_44 8 18

n\_45 8 20

n\_46 8 23

n\_47 8 42

n\_48 8 6

n\_49 8 11

n\_50 8 25

n\_51 8 2

n\_52 8 2

n\_53 8 3

n\_54 8 4

n\_55 9 5

n\_56 8 14

n\_57 8 4

n\_58 8 2

n\_59 8 4

n\_60 8 2

n\_61 8 14

n\_63 8 3

n\_64 8 2

n\_65 8 8

n\_66 8 2

n\_67 8 2

n\_68 8 6

n\_70 8 2

n\_71 8 2

n\_72 8 2

n\_73 8 5

n\_74 8 2

n\_75 8 2

n\_76 8 2

n\_77 8 6

n\_78 8 2

n\_79 8 2

n\_80 8 6

n\_81 8 3

n\_82 8 6

n\_83 8 3

n\_84 9 2

n\_85 8 2

n\_86 8 4

n\_87 8 2

n\_88 8 6

n\_89 8 3

n\_90 8 2

n\_91 8 3

n\_92 8 2

n\_93 8 2

n\_94 9 4

n\_95 8 2

n\_96 8 2

n\_97 8 2

n\_98 9 2

n\_99 9 2

n\_100 8 2

n\_101 8 2

n\_102 8 2

n\_103 8 2

n\_104 8 2

n\_105 8 4

n\_106 8 2

n\_107 8 2

n\_108 8 2

n\_110 8 2

n\_111 9 2

n\_112 8 2

n\_113 9 2

n\_114 8 2

n\_115 8 2

n\_116 8 2

$strataSize

Initial Final

0 4 16

1 302 277

2 124 119

3 104 105

4 42 54

5 453 405

6 464 419

7 183 174

8 359 323

9 358 324

10 669 580

11 189 183

12 81 88

13 215 199

14 138 138

15 26 32

16 271 249

17 9 20

18 60 65

19 195 199

20 59 58

21 39 38

22 95 95

23 55 69

24 75 79

25 103 106

26 21 35

27 76 80

28 642 556

29 638 550

30 7 22

31 248 245

32 141 139

33 189 195

34 22 39

35 63 62

36 53 52

37 189 183

38 62 71

39 1000 868

40 13 26

41 30 39

42 28 37

43 238 217

44 308 290

45 104 101

46 195 185

47 106 111

48 150 150

49 288 265

50 3 11

51 13 27

52 4 10

53 143 140

54 27 40

55 163 156

56 49 53

57 12 21

58 69 73

59 35 52

60 241 229

61 7 17

62 15 24

63 226 214

64 24 37

65 3 14

66 247 238

67 47 61

68 30 43

69 41 53

70 179 183

71 72 81

72 9 19

73 28 34

74 92 93

75 4 15

76 26 31

77 58 70

78 15 21

79 34 40

80 88 92

81 6 23

82 15 30

83 19 30

84 20 37

85 221 201

86 19 27

87 38 46

88 194 193

89 67 72

90 22 34

91 21 30

92 18 32

93 15 28

94 20 35

95 32 48

96 3 15

97 92 106

98 11 25

99 4 20

100 16 28

101 3 18

102 7 16

103 29 44

104 14 30

105 26 36

106 10 18

107 2 17

108 4 21

109 2 19

110 6 15

111 3 15

112 2 15

## Method 2 results based on GA with K-means initial solution

sa\_solution\_acv\_multi\_1

$CVs

Initial Final Target

1 0.2504047 0.1501664 0.02

$sampleSize

Initial Final

n\_1 8 14

n\_2 8 10

n\_3 8 12

n\_4 8 21

n\_5 8 8

n\_6 8 7

n\_7 8 2

n\_8 7 2

n\_9 8 17

n\_10 8 3

n\_11 7 9

n\_12 8 3

n\_13 7 2

n\_14 7 2

n\_15 8 4

$strataSize

Initial Final

0 101 56

1 131 115

2 145 97

3 111 52

4 149 143

5 142 117

6 13 70

7 48 92

8 78 49

9 53 77

10 55 48

11 43 53

12 2 37

13 14 53

14 37 63

sa\_solution\_acv\_multi\_2

$CVs

Initial Final Target

1 0.5472368 0.08057815 0.02

$sampleSize

Initial Final

n\_1 6 4

n\_2 6 3

n\_3 5 4

n\_4 6 13

n\_5 6 4

n\_6 6 7

n\_7 6 9

n\_8 6 3

$strataSize

Initial Final

0 25 33

1 47 3

2 4 4

3 42 17

4 21 22

5 28 71

6 25 22

7 7 27

sa\_solution\_acv\_multi\_3

$CVs

Initial Final Target

1 0.2364553 0.1829049 0.02

$sampleSize

Initial Final

n\_1 2 2

n\_2 2 2

n\_3 2 2

n\_4 2 2

n\_5 2 2

n\_6 2 2

n\_7 2 2

n\_8 2 2

n\_9 2 2

n\_10 2 2

n\_11 2 2

n\_12 2 2

n\_13 2 2

n\_14 2 2

n\_15 2 2

n\_16 2 2

n\_17 2 2

n\_18 2 2

n\_19 2 2

n\_20 2 2

n\_21 2 2

n\_22 2 2

n\_23 2 2

n\_24 2 2

n\_25 2 2

n\_26 2 2

$strataSize

Initial Final

0 28 53

1 16 30

2 23 34

3 9 23

4 41 57

5 14 34

6 115 66

7 20 30

8 57 72

9 34 54

10 35 35

11 56 52

12 57 46

13 178 163

14 72 82

15 24 43

16 101 68

17 48 44

18 104 80

19 36 40

20 23 43

21 144 100

22 114 97

23 125 93

24 15 26

25 47 71

sa\_solution\_acv\_multi\_4

$CVs

Initial Final Target

1 0.2571328 0.1360638 0.02

$sampleSize

Initial Final

n\_1 5 13

n\_2 5 13

n\_3 5 4

n\_4 6 8

n\_5 5 8

n\_6 6 5

n\_7 5 2

n\_8 5 3

n\_9 6 3

n\_10 5 3

n\_11 5 2

n\_12 5 2

n\_13 6 3

$strataSize

Initial Final

0 109 54

1 106 52

2 101 103

3 118 71

4 68 51

5 61 73

6 20 38

7 19 15

8 9 23

9 25 53

10 21 68

11 11 57

12 2 12

sa\_solution\_acv\_multi\_5

$CVs

Initial Final Target

1 0.1049951 0.07351552 0.02

$sampleSize

Initial Final

n\_1 29 38

n\_2 29 12

n\_3 29 23

n\_4 29 12

n\_5 29 19

n\_6 28 31

n\_7 29 18

n\_8 29 40

n\_9 29 17

n\_10 29 124

n\_11 29 40

n\_12 29 10

n\_13 29 38

n\_14 29 2

n\_15 29 10

$strataSize

Initial Final

0 1289 1258

1 564 598

2 984 977

3 356 373

4 596 592

5 1546 1552

6 979 988

7 1431 1404

8 581 596

9 1741 1669

10 1101 1053

11 575 598

12 327 335

13 89 152

14 292 306

## Related formulae

|  |  |  |
| --- | --- | --- |
| Estimator for the total 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

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*(1), 1–24. 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*(4). 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. Retrieved April 23, 2019, from 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. Retrieved August 12, 2019, from https://rdrr.io/cran/SamplingStrata/man/optimizeStrata.html

Basoglu, ismail. (2014). *Ismail\_basoglu\_phd\_thesis.pdf* (Bogazici University). Retrieved from 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. Retrieved April 23, 2019, from https://www.bts.gov/archive/publications/commodity\_flow\_survey/methodology\_2012

Bureau of Transportation Statistics. (2019b, March 29). Freight Analysis Framework. Retrieved April 23, 2019, from https://www.bts.gov/faf

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

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. Retrieved April 23, 2019, from GitHub website: 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

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. Retrieved January 7, 2019, from https://www.nass.usda.gov/Surveys/Guide\_to\_NASS\_Surveys/June\_Area/

R Consortium. (2014, March 27). RStudio. Retrieved April 23, 2019, from RStudio website: https://www.rstudio.com/

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

Statistics Dictionary. (2019a). Neyman Allocation: Definition. Retrieved January 8, 2019, from https://stattrek.com/statistics/dictionary.aspx?definition=neyman\_allocation

Statistics Dictionary. (2019b). Optimum Allocation: Definition. Retrieved January 8, 2019, from https://stattrek.com/statistics/dictionary.aspx?definition=optimum\_allocation

The Comprehensive R Archive Network. (2019, March 11). Retrieved April 18, 2019, from https://cran.r-project.org/

The PostgreSQL Global Development Group. (2019a). PostgreSQL: Documentation: 11: 43.1. Overview. Retrieved April 23, 2019, from https://www.postgresql.org/docs/current/plpgsql-overview.html

The PostgreSQL Global Development Group. (2019b). PostgreSQL: The world’s most advanced open source database. Retrieved April 23, 2019, from https://www.postgresql.org/

The R Foundation. (2019). R: The R Project for Statistical Computing. Retrieved April 23, 2019, from https://www.r-project.org/

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

Thompson, S. K. (2012). *Sampling* (3rd ed). Hoboken, N.J: Wiley.

US Census Bureau. (2018, September 12). County Business Patterns: 2016. Retrieved April 23, 2019, from https://www.census.gov/data/datasets/2016/econ/cbp/2016-cbp.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