**Transforming Freight Flow Data Collection**

Milestone #3

Optimization Model Design and Preliminary Designs

Saeed Ghanbartehrani, Sara Akbar Ghanadian

Ohio University

Athens, OH

Rev. 4, Oct 2019

Contents

[1. Introduction 4](#_Toc22109611)

[2. Case study 4](#_Toc22109612)

[2.1. Data sources 4](#_Toc22109613)

[2.2. Pre-processing of the data 5](#_Toc22109614)

[2.3. Frame data generation 5](#_Toc22109615)

[3. Method 1: Optimal stratification and allocation based on Genetic Algorithm 7](#_Toc22109616)

[3.1. Method 1 procedure 8](#_Toc22109617)

[3.2. Method 1 Results 12](#_Toc22109618)

[4. Method 2: Optimal stratification and allocation using Genetic Algorithm and Simulated Annealing 16](#_Toc22109619)

[4.1. Method 2 procedure 17](#_Toc22109620)

[4.2. Method 2 results 22](#_Toc22109621)

[4.3. Comparing the result of GA and SA 23](#_Toc22109622)

[5. Method3: Generalized Lavallee-Hidiroglou Method for Strata Construction 25](#_Toc22109623)

[5.1. Method 3 results 26](#_Toc22109624)

[6. Appendix 28](#_Toc22109625)

[6.1. Method 1 Scripts 28](#_Toc22109626)

[7. Method 2 Scripts 29](#_Toc22109627)

[8. References 33](#_Toc22109628)

# Introduction

In this milestone, 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.

# Case study

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 ‎2.3.

## 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.

Total value and total weight, by state x SCTG.

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.

Total number of establishments, by county x NAICS.

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. (The coefficient of variation = standard deviation / mean.) 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. Mileage is simply not on the frame- I don’t think it’s a question of how we determine the mileage.

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 ‎2.2) 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 (For the moment, I presume this is the same thing as minimizing the total sample size, i.e. for the moment, we are assuming that each establishment in the sample “costs” the same.) 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). MOS is a continuous variable, so I’m guessing that the need to make it categorical may be a weakness of the proposed method.

The objective function minimizes the total sampling cost. (Subject to the CV constraints.) 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.

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("state","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,..). State, 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 26,436 and the first few rows are displayed below.

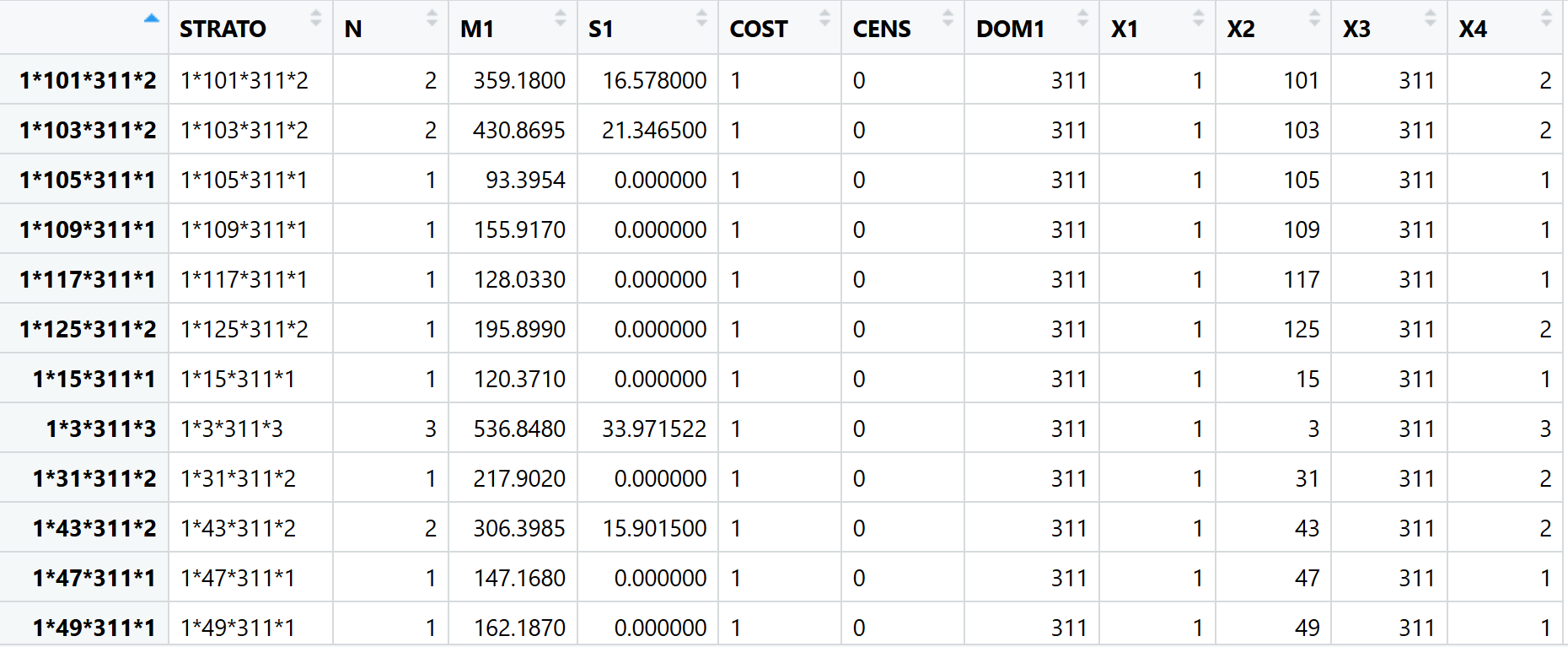


Figure . Atomic strata for the case study

The first column shows the combination of the values for the auxiliary variables (X1, X2, X3, and X4) 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, …,X4 columns are the values of the four auxiliary variables state, county, NAICS, and value class.

Next, “CV.csv” which contains the CV constraints for each domain (State in this case) is imported. In this example, the CV is the CV of the estimate of total value, by state? Under, presumably, stratified, simple random sampling without replacement?

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.



Figure . Selected rows from CV.csv

Then, all input data needs to be checked to ensure consistency using “checkInput” function.

checkInput(errors = CVConst,

strata = AtomicStrata,

sampframe = CFSFrame)

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, initial solution for genetic algorithm is set to null for initialStrata parameter. However, in order to speed up the convergence to the (possibly) optimal solution, an initial solution can be given as a “suggestion” to 'optimizeStrata' function. The number of samples the algorithm generates are real numbers (i.e. numbers with decimal points) which need to be converted to integer before using the results. (I think you mean- The sample sizes the algorithm generates…) ‘realAllocation’ is the parameter that converts the generated sample size values to integer when set to TRUE.

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

Ga\_solution <- optimizeStrata(errors = CVConst,

strata = AtomicStrata,

parallel = TRUE,

initialStrata= null

iter = 100,

realAllocation = False,

writeFiles = FALSE,

showPlot = FALSE)

Although the entire results are stored in “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” while “aggr\_strata” shows the number of samples (the sample sizes) allocated to each stratum along with some other details. After finding the optimal stratification, the function ‘updateStrata’ is used to assign new labels to the atomic strata.

newstrata <- updateStrata(AtomicStrata,

ga\_solution,

writeFiles = TRUE)

The last step is to call ‘updateFrame’ and ‘selectSample’ functions in order to update the frame to create primary sampling unit (I think you mean- to create the strata.) and select the samples using stratified simple random sampling without replacement method. In this case, CFSFrame and newstrata are inputs, and the result is a data frame containing the selected units, along with their weights.

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, the results are visualized and discussed in more detail. 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 Figure 3 and Figure 4. “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.



Figure . Selected rows from “aggr\_strata.csv” file

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 Figure 4.



Figure . Selected rows from “indices.csv” file

“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 Figure 5.



Figure . Selected rows from “newStrata.csv” file

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 Figure 6.



Figure . Selected rows from “sample\_CFS.csv” file

The summary of result from optimizeStrata function is assigned to Ga-solution variable and is printed in R console once the process completes. The summary of the results for two domains is presented in Figure 7.

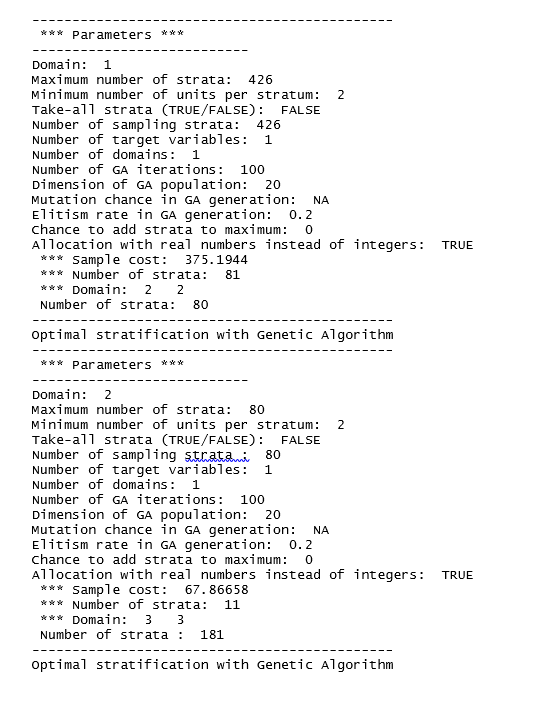


Figure . Summary of Ga\_solution

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

install.packages("remotes")

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

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

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

‘ga\_solution$aggr\_strata’ and ‘solution$indices’ 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

Results for two domains (i.e. domains 2 and 5) 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\_dom2$CVs

Initial Final Target

1 0.6214441 0.03776491 0.02

sa\_solution\_dom5$CVs

Initial Final Target

1 0.02398511 0.00002787737 0.02

$sampleSize\_dom2

Initial Final

n\_1 6 6

n\_2 6 7

n\_3 6 7

n\_4 6 2

n\_5 6 3

n\_6 7 3

n\_7 6 8

n\_8 6 13

n\_9 7 7

$sampleSize\_dom5

Initial Final

n\_1 79 73

n\_2 79 82

n\_3 79 68

n\_4 80 97

n\_5 79 90

n\_6 79 71

n\_7 79 87

n\_8 79 74

n\_9 79 77

n\_10 79 72

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

Total sample size\_dom2= 56

Total sample size\_dom5= 791

sa\_solution\_dom2$strataSize

Initial Final

0 13 6

1 13 23

2 89 7

3 33 3

4 22 32

5 3 52

6 15 22

7 6 22

8 4 31

sa\_solution\_dom5$strataSize

Initial Final

0 21 23

1 75 83

2 53 49

3 121 120

4 114 112

5 53 53

6 95 87

7 64 65

8 22 27

9 52 51

$stratasize compares the initial and final number of strata in the sample.

## Comparing the result of GA and SA

Comparing table of stratification based on GA and SA for domain 2 is presented in Figure 1 (Only 13 rows are shown). Result of comparing sampling in SA and GA are presented in Figure 2.



Table -SA and GA stratifications



Table -Sa and GA allocations

# 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 for a given total number of strata to meet the CV constraint. Neyman allocation was used to determine the sample size for each MOS stratum.

In this study, “Stratification” package is applied to simulate the currant CFS sampling approach to provide a basis for comparison of the results. This package uses generalized 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 parameters of CV (target CV) or target sample size (n) need to be provided as input. To be consistent with the previously discussed methods, we used coefficient of variation constraints for each domain (“CVConst” table used in GA method). The model parameter was set to "none", so the original Lavallee-Hidiroglou method is used. The alloc parameter for Neyman allocation was set to (q1=q3=0.5 and q2=0). Parameter Ls is the desired number of sampled strata (Ls) that was set to 3 (the default value). The following script is for domain 1 and domain 2.

X1= as.vector(as.numeric(x1))

Lavalle\_solution\_dom1 <-strata.LH(X1,

CV= 0.02,

model='random'

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

Ls=3)

print(Lavalle\_solution\_dom2)

#---domain5----------

X5= as.vector(as.numeric(x1))

Lavalle\_solution\_dom1<-strata.LH(X2,

CV= 0.02,

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

Ls=3)

print(Lavalle\_solution\_dom2)

## 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 = X1

CV = 0.02, 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 | 394.35 124.80 10698.35 786 52 0.07

stratum 2 | take-some 1 | 1431.97 790.07 68469.77 219 37 0.17

stratum 3 | take-all 1 | 12838.80 3848.00 9715867.16 111 111 1.00

Total 1116 200 0.18

Total sample size: 200

Anticipated population mean: 625.668

Anticipated CV: 0.01986803

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

Given arguments:

x = X2

CV = 0.03, 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 | 186.42 40.93 1657.48 119 6 0.05

stratum 2 | take-some 1 | 847.38 460.33 32138.01 57 12 0.21

stratum 3 | take-all 1 | 12168.70 3553.93 12825574.48 23 23 1.00

Total 199 41 0.21

Total sample size: 41

Anticipated population mean: 567.0858

Anticipated CV: 0.02882883

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 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)

# References

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

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

Ghanbartehrani, S. (2019, April 10). Saeedt/CFS\_Sampling. Retrieved April 23, 2019, from GitHub website: https://github.com/saeedt/CFS\_Sampling

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

Lisic, J., Sang, H., Zhu, Z., & Zimmer, S. (2018). 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

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)

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/

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