**Transforming Freight Flow Data Collection**

Milestone #3

Optimization Model Design and Preliminary Designs

Saeed Ghanbartehrani, Sara Akbar Ghanadian

Ohio University

Athens, OH

Rev. 2, Aug 2019

Contents

[1. Introduction 4](#_Toc16061346)

[2. Case study 4](#_Toc16061347)

[2.1. Data sources 4](#_Toc16061348)

[2.2. Pre-processing of the data 5](#_Toc16061349)

[2.3. Frame data generation 5](#_Toc16061350)

[3. Method 1: Optimal stratification and allocation based on Genetic Algorithm 6](#_Toc16061351)

[3.1. Method 1 procedure 8](#_Toc16061352)

[3.2. Method 1 Results 11](#_Toc16061353)

[4. Method 2: Optimal stratification and allocation using Genetic Algorithm and Simulated Annealing 12](#_Toc16061354)

[4.1. Method 2 procedure 13](#_Toc16061355)

[4.2. Method 2 results 16](#_Toc16061356)

[5. Appendix 18](#_Toc16061357)

[5.1. Method 1 Scripts 18](#_Toc16061358)

[5.2. Method 2 Scripts 20](#_Toc16061359)

[6. References 22](#_Toc16061360)

# 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.

## 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 developed 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. Weight and value are estimates from FAF and are included in the function for experimental purposes. 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 and standard deviations calculated based on the user provided CVs (0.1 in this case). 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. If all sampling costs are set to 1, the model minimizes the total sample size.

Barcaroli (2014) implemented their proposed method in and 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, 2019).

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 = "naics")

df is the matrix where 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 “. State, county, and naics are categorical used as auxiliary variables. Value is converted to 15 categories and used as the forth 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.

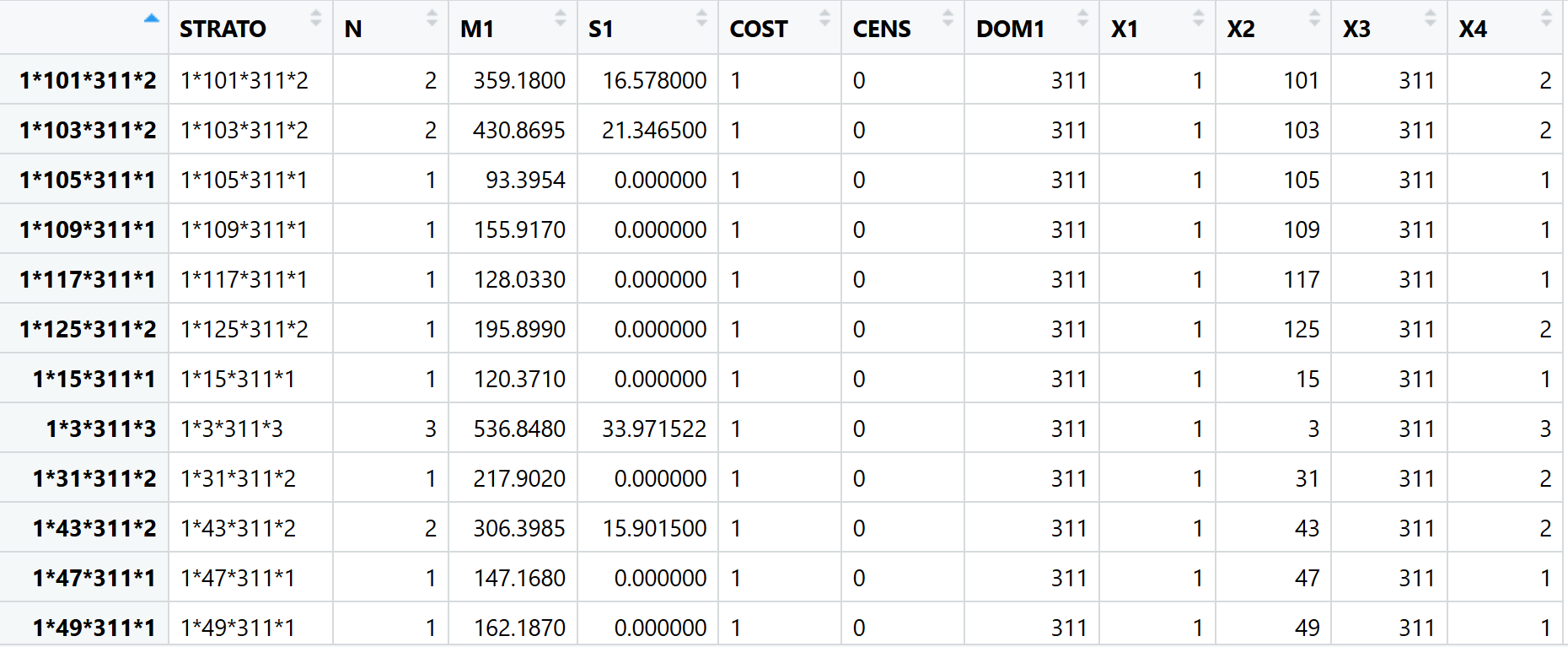


Figure . Atomic strata for the case study

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 (industry identified by NAICS 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, a CV constraint of 0.05 is defined for each NAICS code (presented in “domainvalue” column). Note that the values in “domainvalue” column in “CV.csv” file and the variable assigned to “domanvalue” in “buildFrameDF” function discussed earlier need to be consistent.

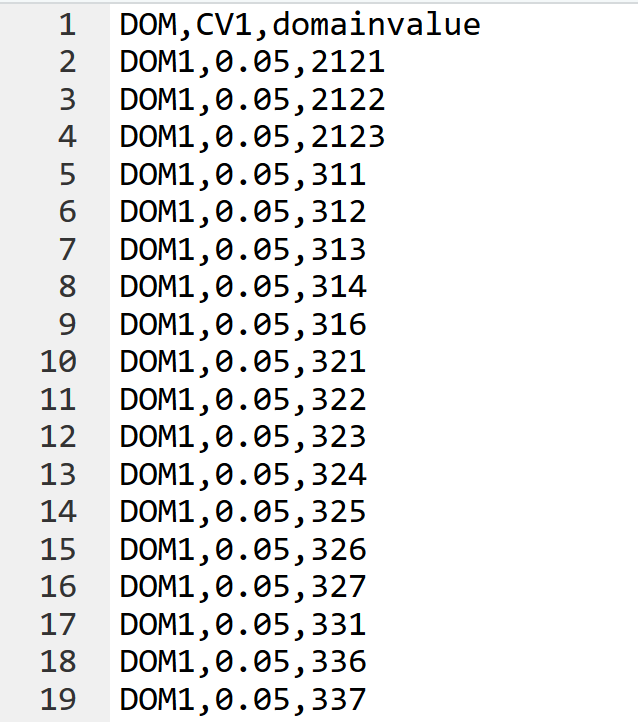


Figure . Selected rows from CV.csv

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

checkInput(errors = CVConst,

strata = AtomicStrata,

sampframe = CFSFrame)

The last 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), output files and plots are specified along with the two required input matrices which are errors (CV constraints) and atomic strata. The full list of input arguments along with their descriptions and an example are provided on the package page on rdrr.io (Barcaroli, 2019).

solution <- optimizeStrata(errors = CVConst,

strata = AtomicStrata,

parallel = TRUE,

iter = 100,

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

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

The stratification is presented in “indices” while “aggr\_strata” shows the number of samples allocated to each stratum along with some other details.

# Method 1 Results

A detailed discussion of the results along with tables and visualizations will be added in this section.

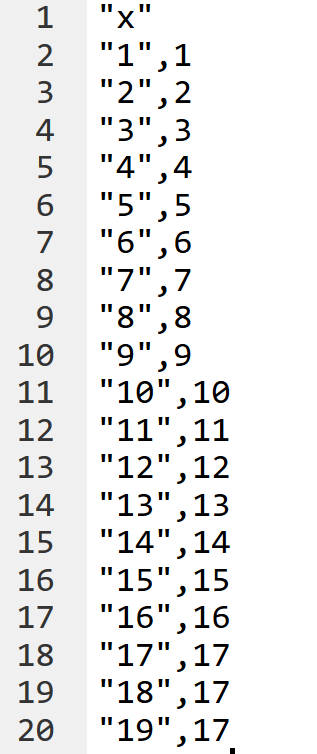


Figure . Selected rows from “indices.csv” file

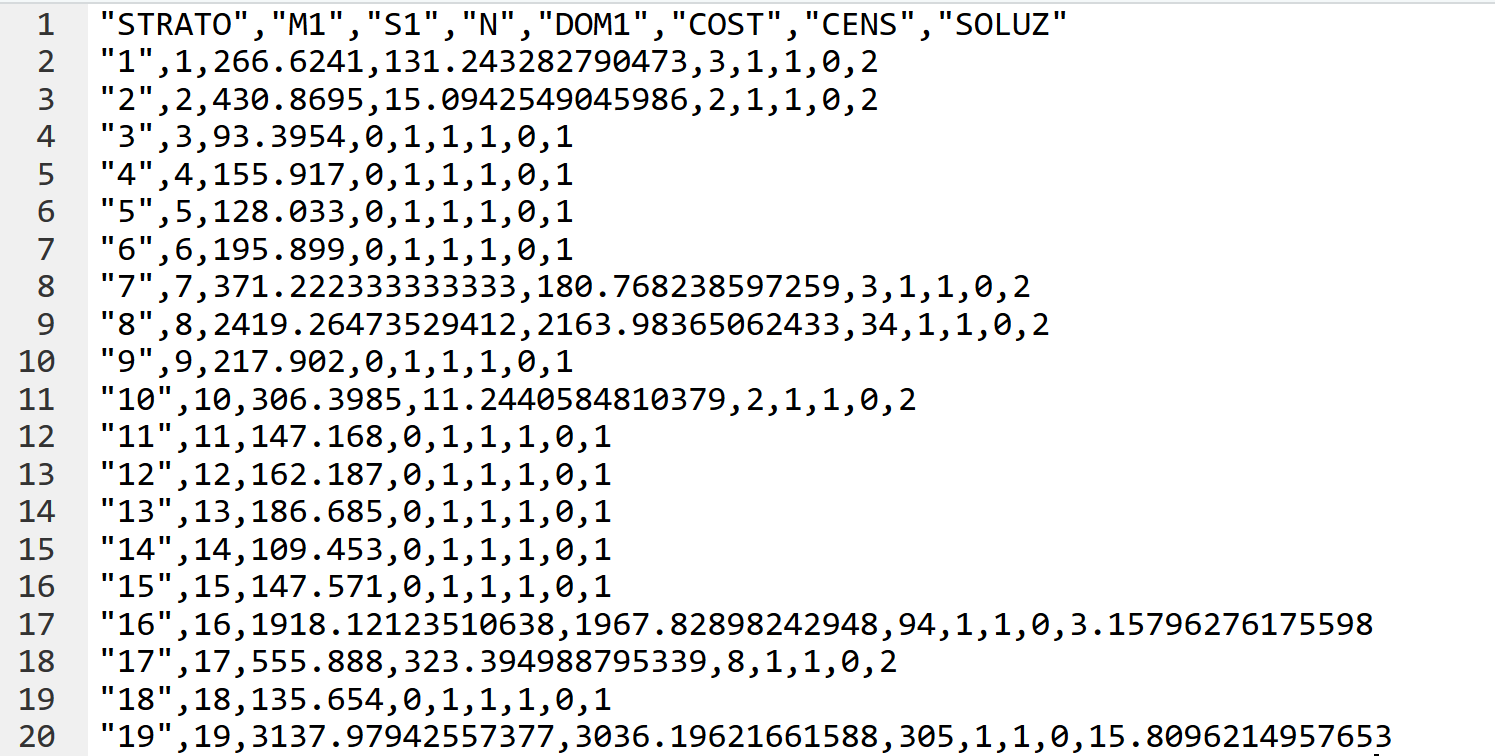


Figure . Selected rows from “aggr\_strata.csv” file

# 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 solution from the GA based method is used as the initial solution for the SA based method. The SA based method further improves the initial solution by minimizing 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 package’s GitHub repository (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 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. 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. 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,]

labeldom1=as.numeric(strata.domain1[,4]) # initial stratification to be used in Simulated Annealing

x1 <- as.matrix(as.numeric(strata.domain1[,5])) #vector of domain 1 PSUs

# 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. This function exchanges 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 different domains (i.e. domain 2 and domain 5) are presented. Number of iterations was set to 1000 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 1.100127 0.2730348 0.02

sa\_solution\_dom5$CVs

Initial Final Target

1 0.6554945 0.3766564 0.02

$sampleSize

Initial Final

n\_1 3 2

n\_2 3 3

n\_3 3 6

n\_4 4 2

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

sa\_solution\_dom2$sampleSize

Initial Final

n\_1 3 3

n\_2 3 5

n\_3 3 2

n\_4 4 3

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

sa\_solution\_dom2$strataSize

Initial Final

0 28 7

1 129 15

2 35 115

3 7 62

sa\_solution\_dom5$strataSize

Initial Final

0 5 56

1 98 236

2 453 233

3 114 145

## Comparing 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 1-SA and GA stratifications



Table 2-Sa and GA allocations

# 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)

# Data manipulation to be used in SA : create labledom1 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,]

labeldom1=as.numeric(strata.domain1[,4]) # initial stratification to be used in simulated anealing

x1 <- as.matrix(as.numeric(strata.domain1[,5])) # vector of target value for each psu

# 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]))

penalty2 = c(10)

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

# 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. (2019, 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. (2019, 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/

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