Stratification as an optimization problem

C4D2 TRAINING
ROME | CENTER FOR DEVELOPMENT DATA

Giulio Barcaroli

November 2019

Stratified sampling

Accordingly to Sarndal, Swensson and Wretman

"…in a stratified sampling design the population is divided into nonoverlapping subpopulations called strata. A probability sample is selected in each stratum. The selection in the different strata are indipendent. "

A stratification is a partition of the population units and it is usually defined according to the values of one or more auxiliary variables available for all units of the population.

Optimal allocation for a given stratification

If the stratification is given, a typical problem that has to be solved is the definition of the total sample size and its allocation among the strata so that the expected accuracies of the sample estimates are compliant with some precision constraints.

Optimal allocation for a given stratification

If we have only one variable of interest (univariate case) the problem can be solved by ***Neyman allocation*** criteria, while in the multivariate case it is possible to make use of the ***Bethel algorithm***.

These criteria define the accuracy in terms of variances of the Horvitz-Thompson estimator:

$$Var(\widehat{Y}_g) = \sum_{h=1}^{H} N_h^2 \left(1 - \frac{n_h}{N_h}\right) \frac{S_{h,g}^2}{n_h} \qquad g = 1, \ldots, G$$

Optimal allocation for a given stratification

If we introduce the following cost function:

$$C(n_1, \ldots, n_H) = C_0 + \sum_{h=1}^{H} C_h n_h$$

the optimization problem can be formalized in this way:

$$min = C_0 + \sum_{h=1}^{H} C_h n_h$$

Optimal allocation for a given stratification

under the constraints

$$\begin{cases} CV(\widehat{Y_1}) < U_1 \\ CV(\widehat{Y_2}) < U_2 \\ \dots \\ CV(\widehat{Y_G}) < U_G \end{cases}$$

where

$$CV(\widehat{Y_g}) = \frac{\sqrt{Var(\widehat{Y_g})}}{mean(\widehat{Y_g})}$$

Optimal allocation for a given stratification

In the univariate case it is possible to obtain an analytic solution

$$n_h = \frac{W_h S_h \sum_{h=1}^{H} W_h S_h}{U}$$

while in the multivariate case solution has to be searched by numeric algorithm.

Since the previous problem is equivalent to search for a minimum of a convex function under linear constraints, such solution always exist.

Example 1: optimization with a given stratification

Let us consider for the moment the case in which strata are given.

We make use of the dataset "swissmunicipalities", containing information on the 2896 Swiss municipalities.

```
library(SamplingStrata)
# get data on the 2896 Swiss municipalities
data(swissmunicipalities)
# add a unique identifier
swissmunicipalities$id <- c(1:nrow(swissmunicipalities))
# only one domain
swissmunicipalities$dom <- 1
# selection of variables
swissmun <- swissmunicipalities[,c("id",        # Identifier
                                   "REG",       # Region
                                   "Nom",       # Municipality name
                                   "Airbat",    # Building area
                                   "Airind",    # Industrial area
                                   "HApoly",    # Total municipality area
                                   "POPTOT",    # Total population
                                   "dom"        # Domain
                                   )]
head(swissmun)
```

```
##    id REG         Nom Airbat Airind HApoly POPTOT dom
## 1   1   4      Zurich   2884     260   8781 363273   1
## 2   2   1      Geneve    773      60   1593 177964   1
## 3   3   3       Basel   1023     213   2391 166558   1
## 4   4   2        Bern   1070     212   5162 128634   1
## 5   5   1    Lausanne    856      64   4136 124914   1
## 6   6   4 Winterthur    972     238   6787  90483   1
```

Example 1: optimization with a given stratification

We decide to use "region", "total area" and "total population" as stratification variables (these latter two after categorizing them)

```
# categorization of total area and total population
swissmun$cl_area <- var.bin(swissmun$HApoly,bins = 10)
swissmun$cl_pop <- var.bin(swissmun$POPTOT,bins = 10)
# definition of sampling frame
frame1 <- buildFrameDF(df = swissmun,
                       id = "id",
                       X = c("REG","cl_pop","cl_area"),
                       Y = c("Airbat","Airind"),
                       domainvalue = "dom")
head(frame1)
```

```
##   id X1 X2 X3   Y1  Y2 domainvalue
## 1  1  4 10  8 2884 260           1
## 2  2  1  9  4  773  60           1
## 3  3  3  9  5 1023 213           1
## 4  4  2  9  7 1070 212           1
## 5  5  1  9  6  856  64           1
## 6  6  4  8  7  972 238           1
```

Example 1: optimization with a given stratification

We decide to use “region”, “total area” and “total population” as stratification variables (these latter two after categorizing them)

```
# construction of atomic strata
strata1 <- buildStrataDF(frame1)
##
## Computations are being done on population data
##
##
## Number of strata:  273
## ... of which with only one unit:  73
head(strata1)
##         STRATO   N       M1        M2        S1        S2 COST CENS DOM1 X1 X2 X3
## 1*1*1    1*1*1 216 12.62037 0.9861111  9.183033 2.3638465    1    0    1  1  1  1
## 1*1*10 1*1*10   1 10.00000 2.0000000  0.000000 0.0000000    1    0    1  1  1 10
## 1*1*2    1*1*2  83 15.92771 1.3855422  7.764273 3.5050096    1    0    1  1  1  2
## 1*1*3    1*1*3  41 21.68293 1.0731707 17.323942 1.8791571    1    0    1  1  1  3
## 1*1*4    1*1*4  20 24.55000 0.8500000 11.732327 1.7965244    1    0    1  1  1  4
## 1*1*5    1*1*5  12 26.00000 0.6666667 14.520101 0.8498366    1    0    1  1  1  5
```

Example 1: optimization with a given stratification

We now define our precision constraints, by setting maximum acceptable values on the expected coefficients of variation of estimates on the two target variables: 5% on “building area” and 10% on “industrial area”.

```
# precision constraints
cv <- as.data.frame(list(DOM="DOM1",
            CV1=0.05,
            CV2=0.10,
            domainvalue=1))
cv
##     DOM   CV1 CV2 domainvalue
## 1 DOM1 0.05 0.1            1
```

Example 1: optimization with a given stratification

It is now possible to apply the Bethel algorithm in order to calculate the total sample size and the allocation in strata, necessary to satisfy these precision constraints.

```
# precision constraints
allocation <- bethel(strata1,cv,printa=TRUE)
head(allocation)
```

```
##  1*1*1 1*1*10  1*1*2  1*1*3  1*1*4  1*1*5
##      2      1      2      2      2      2
```

```
attributes(allocation)$outcv
```

```
##      TYPE   DOMAIN/VAR. PLANNED CV  ACTUAL CV SENSITIVITY 10%
## [1,] "DOM1" "1/V1"      "0.05"      "0.0215"  "1"
## [2,] "DOM1" "1/V2"      "0.1"       "0.0507"  "9"
```

```
sum(allocation)
```

```
## [1] 473
```

Search of the optimum in the universe of stratifications

If the stratification is not given, the solution has to be searched possibly using some optimal criteria.

Coherently with previous definitions, we define the best stratification as the one that support

$$\sum_{h=1}^{H} n_h = \min$$

for a given *U*.

SamplingStrata looks for the best stratification among all possible stratifications

The universe of the stratifications

Given a population frame with M auxiliary qualitative variables $X_1, \ldots, X_M$ we define as **atomic stratification** the one that can be obtained considering the cartesian product of the M variables.

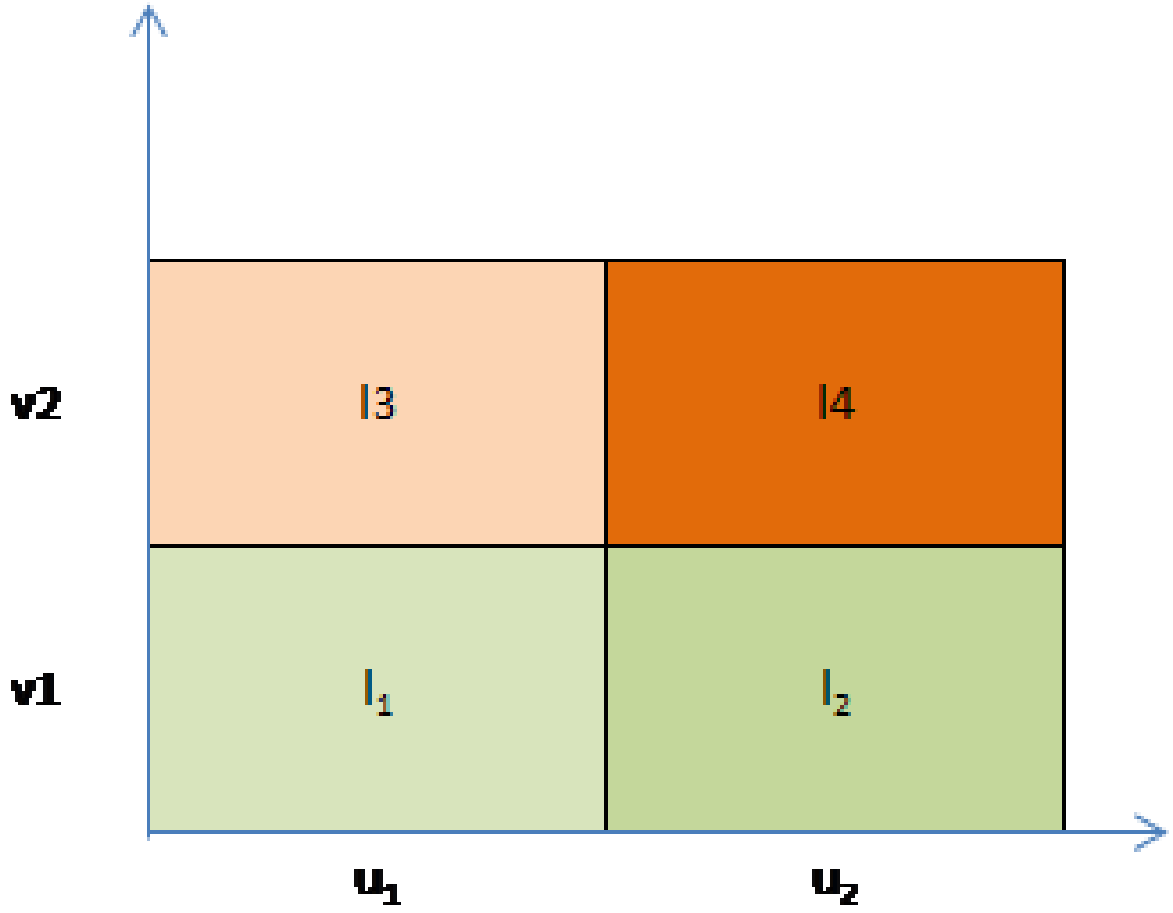
$$L = \{(l_1), (l_2), \ldots, (l_k)\}$$

Starting from the atomic stratification, it is possible to generate all the different stratifications that constitute the universe of stratifications.

The universe of the stratifications

For example, using two dichotomous variables X_1 and X_2 whose modalities are $\{(u_1), (u_2)\}$ and $\{(v_1), (v_2)\}$ respectively, the Cartesian product supports four elements

$$L = \{(l_1), (l_2), (l_3), (l_4)\}$$



The universe of the stratifications

In such conditions, the universe of stratifications is constituted by the following elements:

$$P_1 = \{(l_1, l_2, l_3, l_4)\} \qquad P_2 = \{(l_1), (l_2), (l_3), (l_4)\}$$
$$P_3 = \{(l_1), (l_2, l_3, l_4)\} \qquad P_4 = \{(l_2), (l_1, l_3, l_4)\}$$
$$P_5 = \{(l_3), (l_1, l_2, l_4)\} \qquad P_6 = \{(l_4), (l_1, l_2), l_3\}$$
$$P_7 = \{(l_1, l_2), (l_3, l_4)\} \qquad P_8 = \{(l_1, l_3), (l_2, l_4)\}$$
$$P_9 = \{(l_1, l_4), (l_2, l_3)\}$$
$$P_{10} = \{(l_1, l_2), (l_3), (l_4)\} \qquad P_{11} = \{(l_1, l_3), (l_2), (l_4)\}$$
$$P_{12} = \{(l_1, l_4), (l_2), (l_4)\} \qquad P_{13} = \{(l_2, l_3), (l_1), (l_4)\}$$
$$P_{14} = \{(l_2, l_4), (l_1), (l_3)\} \qquad P_{15} = \{(l_3, l_4), (l_1), (l_2)\}$$

The universe of the stratifications

If we would like to determine which is the best stratification among these fifteen it should be sufficient to solve the optimal allocation problem for each one of them and finally to choose the one that support the minimum total sample size.

Unfortunately, the number of feasible stratifications is exponential with respect to the number of initial atomic strata:

$$B_4 = 15 \quad B_{10} = 115975 \qquad B_{100} \approx 4.76 \times 10^{115}$$

Use of the Genetic Algorithm in the optimization process

Consequently, in concrete cases, it is impossible to examine all alternative stratifications.

The **Genetic Algorithm** allows to explore the universe of stratifications in a very efficient way in order to find the optimal (or close to optimal) solution.

Use of the Genetic Algorithm in the optimization process

The basic implementation of **SamplingStrata** proceeds as follows:

- given the survey variables $Y_1, Y_2, \ldots, Y_p$,**precision constraints** are set on their estimates in the different domains, expressed in terms of coefficients of variations;
- using the auxiliary variables $X_1, \ldots, X_M$ recorded in the frame, **atomic stratification** is built by cross-classifying units in the frame;
- for each atomic stratum the **means** and **standard deviations** of the variables of interest (usually by using proxy variables) are calculated

Use of the Genetic Algorithm in the optimization process

On the basis of these inputs, the *basic version of the optimization algorithm* proceeds in this way:

- in the **first iteration**, a number of different potential stratifications are generated, by randomly aggregating the atomic strata;
- for each stratification, the associated sample size required to satisfy the precision constraints is calculated by applying the Bethel algorithm;
- in **next iterations**, new sets of potential stratifications are generated by selecting previous ones with the lowest associated sample size, and combining them in a convenient way;
- in each iteration, the best stratification (the one with the minimum associated sample size) is retained.

Use of the Genetic Algorithm in the optimization process

After the **final iteration**, this best stratification is given as the optimal solution.

The **solution** is the best in terms of

- **stratification**,
- **total sample size**,
- **sampling units allocation in optimized strata**.

Example 2: optimization with the genetic algorithm

Let us start again from the same dataframe and precision constraints.

Initial strata are the same already seen when applying the Bethel algorithm

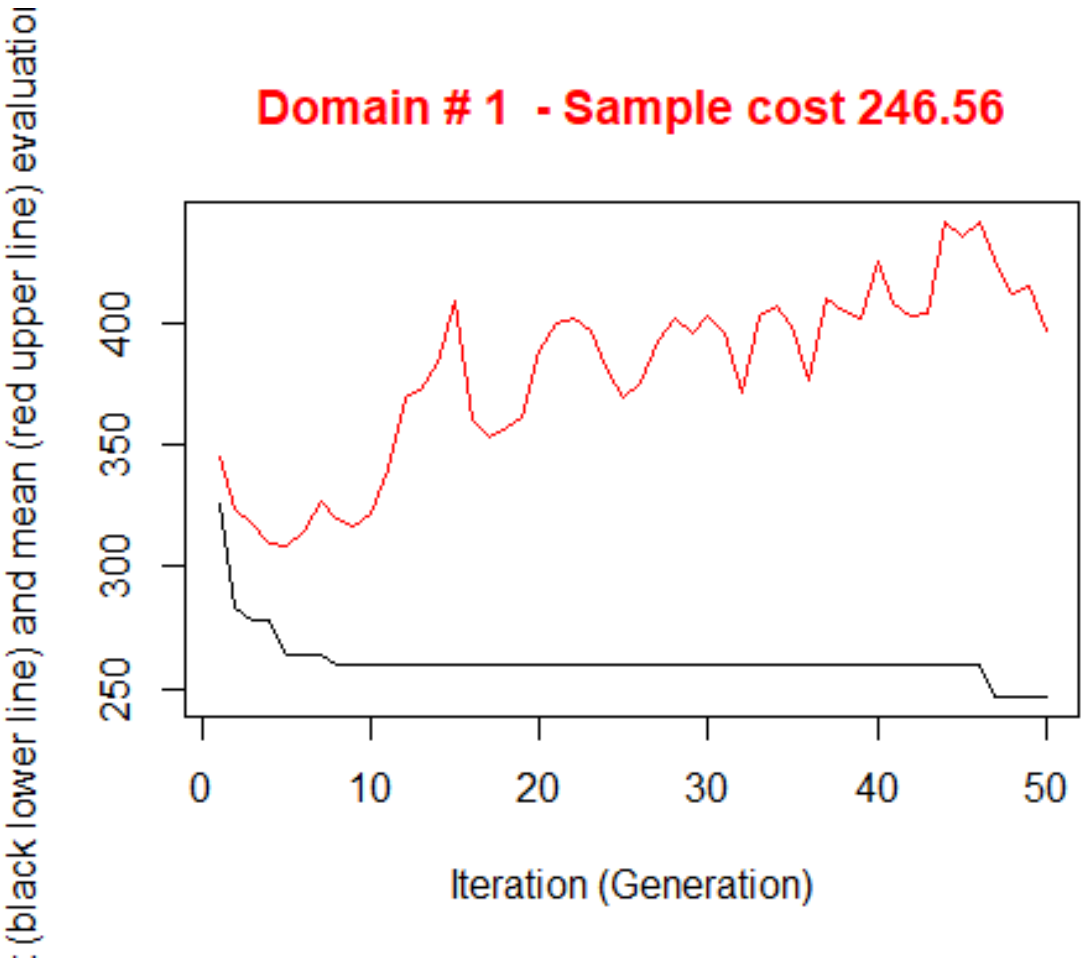
```
solution1 <- optimizeStrata(errors=cv,
                            strata=strata1,
                            iter = 50,
                            pops = 20,
                            parallel=FALSE)
```

```
##
##  *** Domain :  1   1
##  Number of strata :  273
## ---------------------------------------------
## Optimal stratification with Genetic Algorithm
## ---------------------------------------------
##  *** Parameters ***
## --------------------------
## Domain:  1
## Maximum number of strata:  273
## Minimum number of units per stratum:  2
## Take-all strata (TRUE/FALSE):  FALSE
## number of sampling strata :  273
## Number of target variables:  2
## Number of domains:  1
## Number of GA iterations:  50
## 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:  TRUE
```

Example 2: optimization with the genetic algorithm

Let us start again from the same dataframe and precision constraints.

Initial strata are the same already seen when applying the Bethel algorithm




```
##
##   *** Sample cost:  246.5561
##   *** Number of strata:  64
```

Optimization with continuous variables

Up to now, the optimization step has been illustrated assuming that the stratification variables are qualitative variables or assuming that quantitative variables have been previously categorized (*SamplingStrata* has a function that suggests a split of a quantitative variable in a given number of classes).

If we want to stratify using two or more quantitative variables we can use a different version of the optimization algorithm.

Optimization with continuous variables

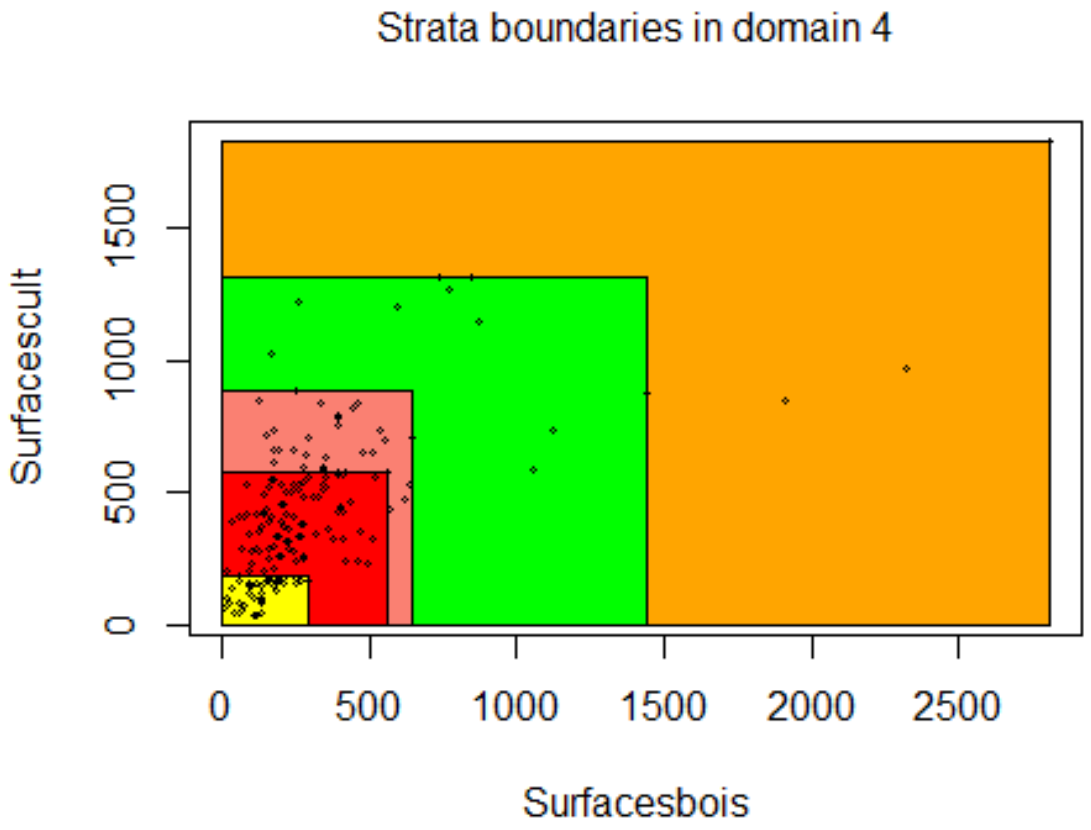
In this case we do not need to build the initial atomic stratification because each potential stratification is obtained by k random cuts of the range of each stratification variable and then by cross-classifying units in the frame accordingly to these “categorized” values.

The algorithm proceeds as in the previous case, iterating the generation of potential stratifications by selecting and combining those more efficient.

Optimization with continuous variables

The resulting strata are typically “7-shaped”.

The “7” shape of the strata has been introduced to meet requests by statisticians in charge of the business surveys who look for simple stratification rules.



Example 3: optimization with continuous stratification variables

This time, we define a sampling frame in which stratification variables are continuous.

We use the same precision constraints.

No need to build the initial stratification.

```
frame3 <- buildFrameDF(df = swissmun,
                       id = "id",
                       X = c("POPTOT","HApoly"),
                       Y = c("Airbat","Airind"),
                       domainvalue = "dom")
head(frame3)
```

```
##   id     X1   X2   Y1  Y2 domainvalue
## 1  1 363273 8781 2884 260           1
## 2  2 177964 1593  773  60           1
## 3  3 166558 2391 1023 213           1
## 4  4 128634 5162 1070 212           1
## 5  5 124914 4136  856  64           1
## 6  6  90483 6787  972 238           1
```

Example 3: optimization with continuous stratification variables

This time, we define a sampling frame in which stratification variables are continuous.

We use the same precision constraints.

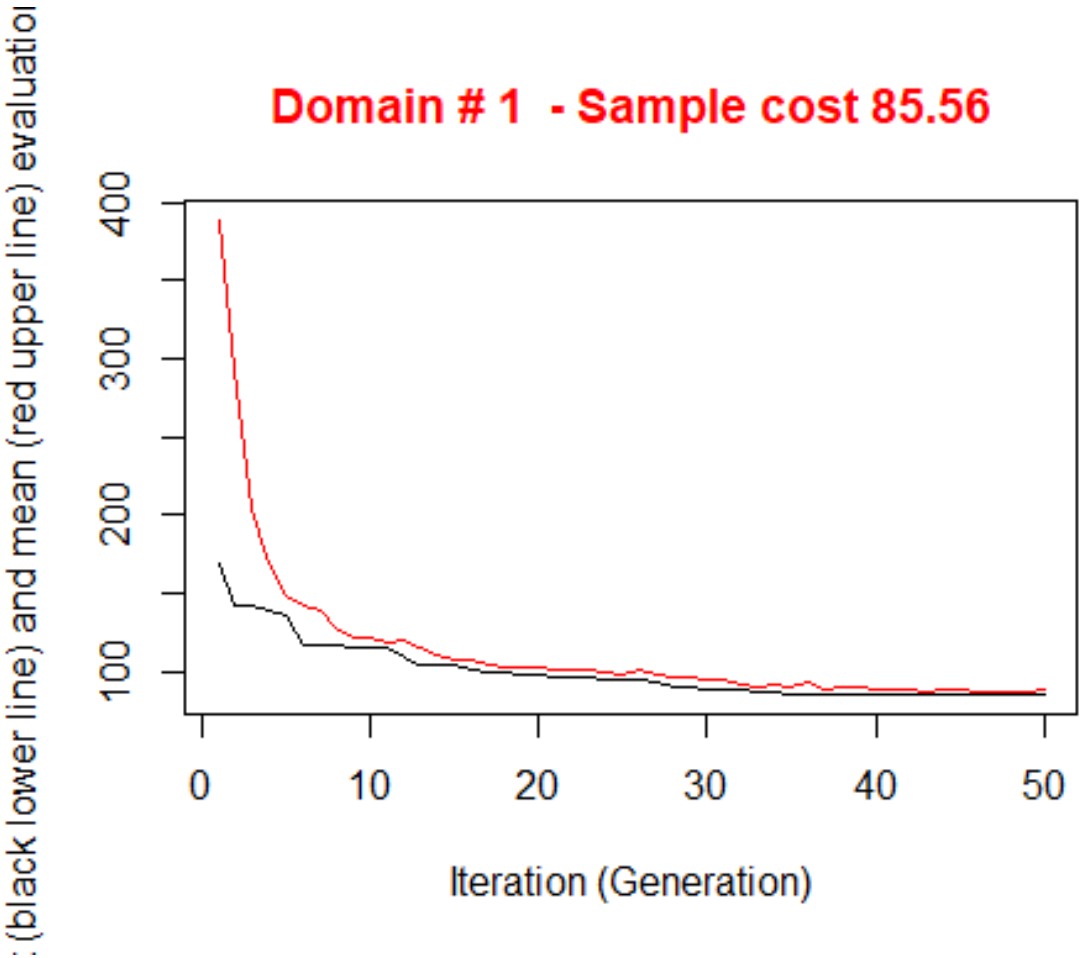
No need to build the initial stratification.

```
set.seed(1234)
solution3 <- optimizeStrata2(errors=cv,
                             framesamp=frame3,
                             nStrata = 5,
                             iter = 50,
                             pops = 20,
                             parallel=FALSE)

##
##  *** Domain :  1   1
##  Number of strata :  2896
## GA Settings
##   Population size        = 20
##   Number of Generations  = 50
##   Elitism                = 4
##   Mutation Chance        = 0.111111111111111
```

Example 3: optimization with continuous stratification variables

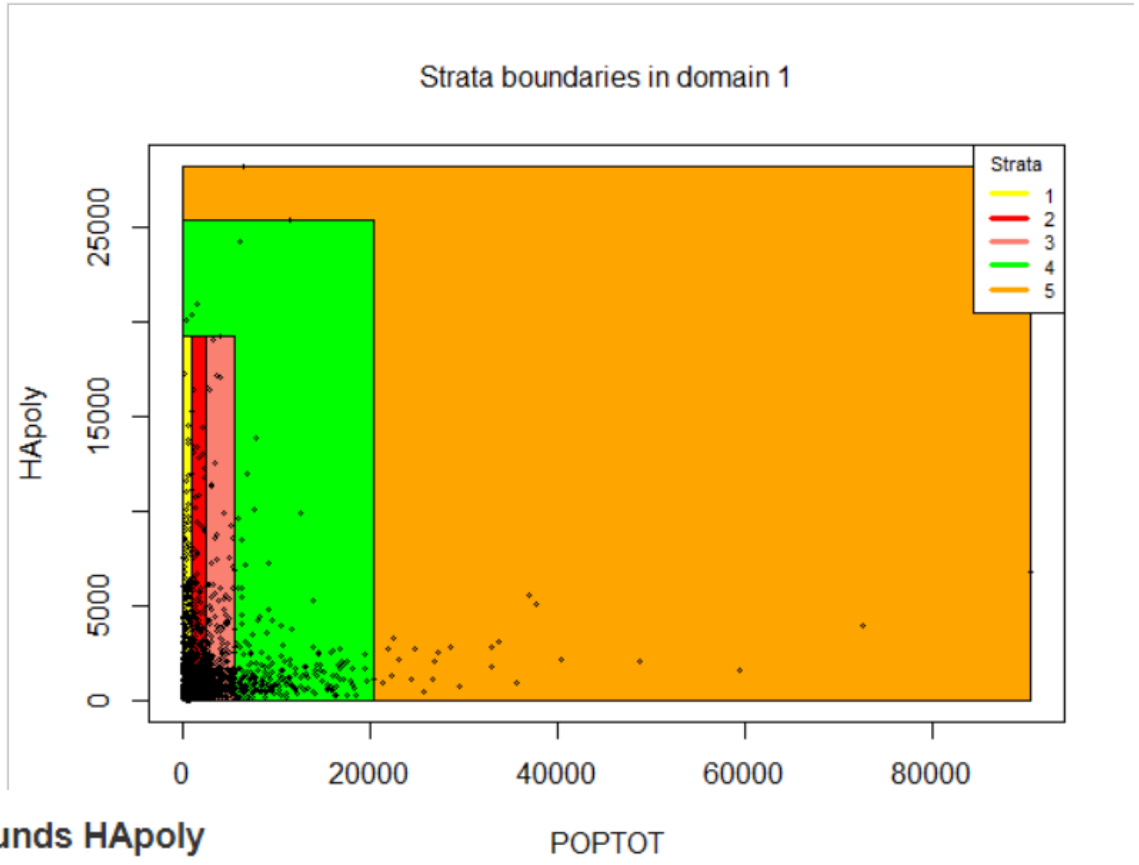
Resulting optimal stratification is much more convenient than the previous ones.




```
##
##   *** Sample cost:  85.56389
##   *** Number of strata:  5
```

Example 3: optimization with continuous stratification variables

The final strata are of the kind “7-shaped”.



Stratum	Population	Allocation	SamplingRate	Bounds POPTOT	Bounds HApoly
1	1438	12	0.008686293	22-862	32-20079
2	802	21	0.025959088	863-2497	48-20994
3	387	19	0.048393613	2501-5447	60-19313
4	239	24	0.100831608	5451-20370	73-25440
5	30	10	0.338445734	6536-90483	477-28225

Example 3: optimization with continuous stratification variables

Finally, a sample can be drawn from the optimized strata..

```
s <- selectSample(framenew,outstrata)
##
## *** Sample has been drawn successfully ***
##  86  units have been selected from  5  strata
head(s)
##   DOMAINVALUE STRATO   ID  X1  X2 Y1 Y2 LABEL  WEIGHTS         FPC
## 1           1      1 1893 496 697 28  0     1 119.8333 0.008344924
## 2           1      1 2096 378 392 10  7     1 119.8333 0.008344924
## 3           1      1 2198 332 751  5  1     1 119.8333 0.008344924
## 4           1      1 1725 614 469 25  0     1 119.8333 0.008344924
## 5           1      1 1852 521 290 12  0     1 119.8333 0.008344924
## 6           1      1 2155 350 139 10  0     1 119.8333 0.008344924
```

Working with anticipated variance

Another important assumption that we have made so far is that the variance and the mean of each variable of interest can be computed for each atomic stratum using data from to a previous survey or available in the frame.

Such assumption is clearly very strong and sometime it cannot be accepted.

If this problem is not taken into account, there will be an under-estimation of variance in strata, and the obtained sample design (stratification, total sample size and allocation in the strata) will not be compliant with the precision constraint.

Working with anticipated variance

In order to cope with this problem, in SamplingStrata it is possible to declare a model for each target variable, and supply related parameters.

In the optimization step, when calculating variance in each stratum, the fact that the available Y is not the real target value, but only a proxy, will be taken into account by considering the given models.

In this way, the variance in each stratum will be correctly calculated, and the obtained sample design will be compliant with the precision constraints.

Working with anticipated variance

In SamplingStrata it is possible to define linear models that assume a relationship between a given variable of interest Z and a proxy variable Y available in the frame.

A typical relationship is the linear model

$$Z_i = \beta * Y_i + \epsilon_i$$

where

$$\epsilon \sim \quad N(0, Y^{\gamma} * \sigma)$$

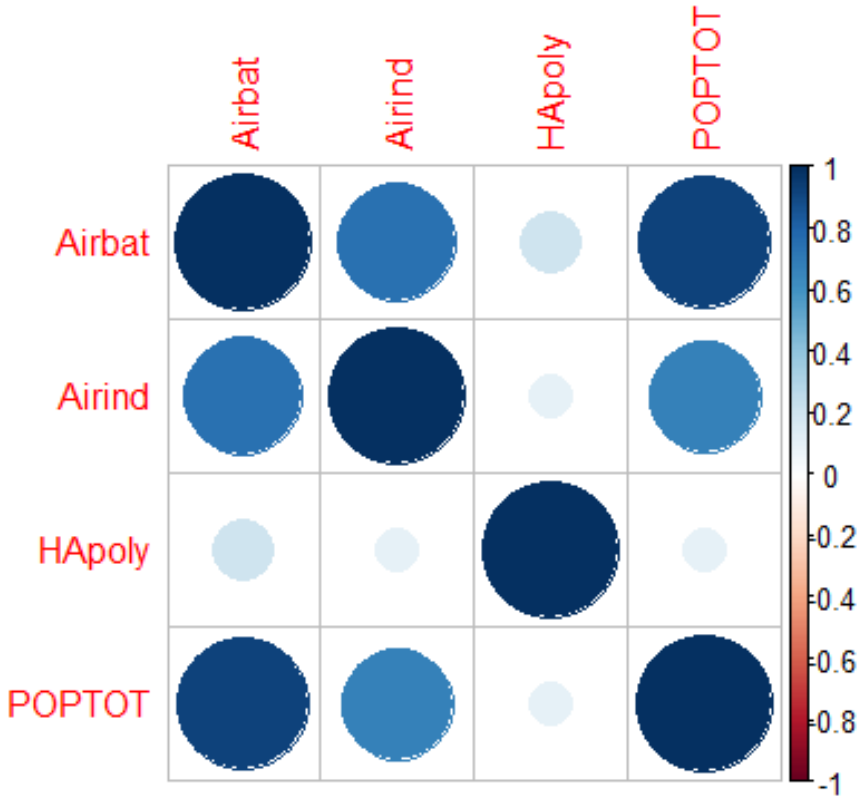
If $\gamma = 0$ the model is homoscedastic otherwise it is heteroscedastic.

Example 4: handling anticipated variance

Let us consider the relations existing in our dataset.

```
correla <- cor(swissmun[,c(4:7)])
correla
```

```
##              Airbat    Airind    HApoly    POPTOT
## Airbat    1.0000000 0.7469404 0.2075253 0.9270286
## Airind    0.7469404 1.0000000 0.1083739 0.6774279
## HApoly    0.2075253 0.1083739 1.0000000 0.1060809
## POPTOT    0.9270286 0.6774279 0.1060809 1.0000000
```



Example 4: handling anticipated variance

If the values of the target variables Airbat and Airind are not available in the frame (at least, not for all units in the frame), we can make use of POPTOT as proxy for both of them, and modeling their relations.

```
mod_Airind_POPTOT <- lm(swiss_sample$Airind ~ swiss_sample$POPTOT)
summary(mod_Airind_POPTOT)

##
## Call:
## lm(formula = swiss_sample$Airind ~ swiss_sample$POPTOT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.466  -2.233  -1.400   0.502  82.261
##
## Coefficients:
##                       Estimate Std. Error t value             Pr(>|t|)
## (Intercept)         1.11483284 0.43046222    2.59              0.00988 **
## swiss_sample$POPTOT 0.00245734 0.00009303   26.41 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.309 on 498 degrees of freedom
## Multiple R-squared:  0.5835, Adjusted R-squared:  0.5827
## F-statistic: 697.7 on 1 and 498 DF,  p-value: < 0.00000000000000022
```

Example 4: handling anticipated variance

If the values of the target variables Airbat and Airind are not available in the frame (at least, not for all units in the frame), we can make use of POPTOT as proxy for both of them, and modeling their relations.

```
mod_Airbat_POPTOT <- lm(swiss_sample$Airbat ~ swiss_sample$POPTOT)
summary(mod_Airbat_POPTOT)
```

```
##
## Call:
## lm(formula = swiss_sample$Airbat ~ swiss_sample$POPTOT)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -143.380  -12.223   -4.391    7.336  208.728
##
## Coefficients:
##                        Estimate Std. Error t value            Pr(>|t|)
## (Intercept)          19.6788127  1.3203745   14.90 <0.0000000000000002 ***
## swiss_sample$POPTOT   0.0119216  0.0002854   41.78 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.49 on 498 degrees of freedom
## Multiple R-squared:  0.778,  Adjusted R-squared:  0.7776
## F-statistic:  1745 on 1 and 498 DF,  p-value: < 0.00000000000000022
```

Example 4: handling anticipated variance

If the values of the target variables Airbat and Airind are not available in the frame (at least, not for all units in the frame), we can make use of POPTOT as proxy for both of them, and modeling their relations.

```
model <- NULL
model$beta[1] <- mod_Airind_POPTOT$coefficients[2]
model$sig2[1] <- summary(mod_Airind_POPTOT)$sigma
model$type[1] <- "linear"
model$gamma[1] <- 0
model$beta[2] <- mod_Airbat_POPTOT$coefficients[2]
model$sig2[2] <- summary(mod_Airbat_POPTOT)$sigma
model$type[2] <- "linear"
model$gamma[2] <- 0
model <- as.data.frame(model)
model
```

```
##          beta       sig2   type gamma
## 1 0.00245734   8.309211 linear     0
## 2 0.01192162 25.487184 linear     0
```

Example 4: handling anticipated variance

Now we define POPTOT, as unique available variable in the frame, both as X and Y variable, and run the optimization step WITHOUT supplying the models.

```
frame4 <- buildFrameDF(swissmun,
                       id="id",
                       X="POPTOT",
                       Y=c("POPTOT","POPTOT"),
                       domainvalue = "dom")
frame4$Airbat <- swissmun$Airbat
frame4$Airind <- swissmun$Airind
```

```
solution4a <- optimizeStrata2(errors=cv,
                              framesamp=frame4,
                              model=NULL,
                              nStrata = 5,
                              iter = 50,
                              pops = 20,
                              parallel=FALSE)
##
##  *** Sample cost:  50.79568
##  *** Number of strata:  5
```

Example 4: handling anticipated variance

If we now run a simulation, by selecting a number of different samples from the frame, we can calculate the CV's on the real target variables. They are, respectively, of 6.4% and 18.1%, by far not compliant.

```
framenew$Y1 <- framenew$AIRBAT
framenew$Y2 <- framenew$AIRIND
val <- evalSolution(framenew,outstrata)
val$coeff_var
```

```
##       CV1    CV2   dom
## 1  0.0639 0.1812 DOM1
```

Example 4: handling anticipated variance

Now we run the optimization step, this time supplying the models.

```
solution4b <- optimizeStrata2(errors=cv,
                              framesamp=frame4,
                              model=model,
                              nStrata = 5,
                              iter = 50,
                              pops = 20,
                              parallel=FALSE)
```

```
##
##   *** Sample cost:  142.1312
##   *** Number of strata:  5
```

Example 4: handling anticipated variance

We run again the simulation, to calculate the CV's on the real target variables. They are, respectively, of 4.3% and 7.9%: now they are compliant.

```
framenew$Y1 <- framenew$AIRBAT
framenew$Y2 <- framenew$AIRIND
val <-
evalSolution(framenew,outstrata,progress=FALSE)
val$coeff_var
```

```
##       CV1    CV2   dom
## 1 0.0434 0.0788 DOM1
```