

Optimization of sampling strata with the **SamplingStrata** package

Package version 0.9.3

Giulio Barcaroli

March 19, 2012

Abstract

*In stratified random sampling the problem of determining the optimal size and allocation of units in strata is solved by considering the stratification of the population as given. Conversely, the definition of the optimal stratification of a sampling frame for a given survey is investigated without choosing, as objective function, the sampling size required to satisfy given precision constraints on the parameters of interest of a given survey. This package allows the determination of the best stratification of a target population, the one that ensures the minimum sample size (or the minimum fieldwork and interviewing costs) so to satisfy precision constraints in a multivariate and multidomain case. The underlying algorithm is based on a non deterministic evolutionary approach, making use of the genetic algorithm paradigm. The specific functions for the execution of the genetic algorithm are a modified version of those contained in the **genalg** package.*

Contents

1	Introduction	3
2	Procedural steps	3
3	Steps 1 and 2: analysis of the frame data and manipulation of auxiliary information	4
4	Construction of atomic strata and association of the information related to target variables (steps 3 and 4)	7
5	Choice of the precision constraints for each target estimate (step 5)	10
6	Optimisation of frame stratification (step 6)	11
7	Analysis of results (step 7)	13
8	Updating the frame and selecting the sample (steps 8 and 9)	16

1 Introduction

Let us suppose we need to design a sample survey, having a complete frame containing information on the target population (identifiers plus auxiliary information). If our sample design is a stratified one, we need to choose how to form strata in the population, in order to get the maximum advantage by the available auxiliary information. In other words, we have to decide in which way to combine the values of the auxiliary variables (from now on, the 'X' variables) in order to determine a new variable, called 'stratum'. To do so, we have to take into consideration the target variables of our sample surveys (from now on, the 'Y' variables): if, to form strata, we choose the X variables most correlated to the Ys, the efficiency of the samples drawn by the resulting stratified frame may be greatly increased. In order to handle the whole auxiliary information in a homogenous way, we have to reduce continuous data to categorical (by mean of a k-means clustering technique, for example). Then, for every set of candidate auxiliary variables Xs, we have to decide (i) what variables to consider as active variables in strata determination, and (ii) for each active variable, what set of values (in general, what aggregation of atomic values) have to be considered. Every combination of values of each active variable determine a particular stratification of the target population, i.e. a possible solution to the problem of 'best' stratification. Here, by best stratification, we mean the stratification that ensures the minimum sample size, sufficient to satisfy a set of precision constraints, set on the accuracy of the estimates of the survey target variables Ys (constraints expressed as maximum allowable sampling variance on estimates in different domains of interest). Therefore, the validity of a particular stratification can be measured by the associated minimum size of a sample, whose estimates are expected to satisfy given accuracy levels. This minimum size can be determined by applying the Bethel algorithm, with its Chromy variant. In general, the number of possible alternative stratifications for a given population may be very high, depending on the number of variables and on the number of their values, and in these cases it is not possible to enumerate them in order to assess the best one. A very convenient solution to this, is the adoption of the evolutionary approach, consisting in applying a genetic algorithm that may converge towards a near-optimal solution after a finite number of iterations. The implementation of the genetic algorithm is based on a modification of the functions in the `genalg` package (see Willighagen (2005)).

2 Procedural steps

The optimization of the sampling design starts by making the sampling frame available, defining the target estimates of the survey and establishing the precision constraints on them. It is then possible to determine the best stratification and the optimal allocation. Finally, we proceed with the selection of the sample. Formalising, these are the required steps:

1. analysis of the frame data: identification of available auxiliary information;

2. manipulation of auxiliary information: in case auxiliary variables are of the continuous type, they must be transformed into a categorical form;
3. construction of atomic strata: on the basis of the categorical auxiliary variables available in the sampling frame, a set of strata can be constructed by calculating the Cartesian product of the values of all the auxiliary variables;
4. characterization of each atomic stratum with the information related to the target variables: in order to optimise both strata and allocation of sampling units in strata, we need information on the distributions of the target variables (means and standard deviations);
5. choice of the precision constraints for each target estimate, possibly differentiated by domain;
6. optimization of stratification and determination of required sample size and allocation in order to satisfy precision constraints on target estimates;
7. analysis of the resulting optimized strata;
8. association of new labels to sampling frame units, each of them indicating the new strata resulting by the optimal aggregation of the atomic strata;
9. selection of units from the sampling frame with a *stratified random sample* selection scheme.

In the following, we will illustrate each step starting from a real sampling frame, the one that comes with the R package `sampling` (the dataframe `swissmunicipalities`).

3 Steps 1 and 2: analysis of the frame data and manipulation of auxiliary information

As a first step, we have to define a frame dataframe containing the following information:

- a unique identifier of the unit (no restriction on the name, may be 'cod');
- the (optional) identifier of the stratum to which the unit belongs;
- the values of m auxiliary variables (named from X1 to Xm);
- the (optional) values of p target variables (named from Y1 to Yp);
- the value of the domain of interest for which we want to produce estimates (named 'domainvalue').

By typing the following statements in the R environment:

```
> library(SamplingStrata)
> data(swissmunicipalities)
```

we get the `swissmunicipalities` dataframe, that contains 2896 observations (each observation refers to a Swiss municipality). Among the others, there are the following variables (data are referred to 2003):

- REG: Swiss region.
- Nom: municipality name.
- Surfacesbois: wood area.
- Surfacescult: area under cultivation.
- Alp: mountain pasture area.
- Airbat: area with buildings.
- Airind: industrial area.
- Pop020: number of men and women aged between 0 and 19.
- Pop2040: number of men and women aged between 20 and 39.
- Pop4065: number of men and women aged between 40 and 64.
- Pop65P: number of men and women aged between 65 and over.
- POPTOT: total population.

First, we define the identifier of the frame:

```
> swissframe <- NULL
> swissframe$id <- swissmunicipalities$Nom
```

Let us suppose we want to plan a survey whose target estimates are the totals of population by age class in each Swiss region. In this case, our Y s variables will be:

- Y1: number of men and women aged between 0 and 19.
- Y2: number of men and women aged between 20 and 39.
- Y3: number of men and women aged between 40 and 64.
- Y4: number of men and women aged between 65 and over.

So we execute the following statements:

```
> swissframe$Y1 <- swissmunicipalities$Pop020
> swissframe$Y2 <- swissmunicipalities$Pop2040
> swissframe$Y3 <- swissmunicipalities$Pop4065
> swissframe$Y4 <- swissmunicipalities$Pop65P
```

As for the auxiliary variables (Xs), we can use all of those characterising the area use (wood, mountain or pasture, cultivated, industrial, with buildings). As these variables are of the continuous type, first we have to reduce them in a categorical (ordinal) form. A suitable way to do so, is to apply a k-means clustering method (see Hartigan and Wong (1979)) by using the function `var.bin`:

```
> library(SamplingStrata)
> swissframe$X1 <- var.bin(swissmunicipalities$POPTOT, bins=18)
> swissframe$X2 <- var.bin(swissmunicipalities$Surfacesbois, bins=3)
> swissframe$X3 <- var.bin(swissmunicipalities$Surfacescult, bins=3)
> swissframe$X4 <- var.bin(swissmunicipalities$Alp, bins=3)
> swissframe$X5 <- var.bin(swissmunicipalities$Airbat, bins=3)
> swissframe$X6 <- var.bin(swissmunicipalities$Airind, bins=3)
```

Now, we have six different auxiliary variables of the categorical type, the first with 18 different modalities, the others with 3 modalities. Finally, we have to set the values of the 'domainvalue' variable, which is mandatory. As we want to obtain estimates for each region, we set:

```
> swissframe$domainvalue <- swissmunicipalities$REG
> swissframe <- data.frame(swissframe)
```

Now, the `swissframe` dataframe looks like this way:

```
> head(swissframe)
```

	id	Y1	Y2	Y3	Y4	X1	X2	X3	X4	X5	X6
1	Zurich	57324	131422	108178	66349	18	3	2	1	3	3
2	Geneve	32429	60074	57063	28398	17	1	1	1	3	2
3	Basel	28161	50349	53734	34314	17	1	1	1	3	3
4	Bern	19399	44263	39397	25575	17	2	3	1	3	3
5	Lausanne	24291	44202	35421	21000	17	2	2	1	3	2
6	Winterthur	18942	28958	27696	14887	16	3	3	1	3	3
	domainvalue										
1		4									
2		1									
3		3									
4		2									
5		1									
6		4									

that is the format required by the package. We write the dataframe to a tab delimited file:

```
> write.table (swissframe, 'swissframe.txt', row.names=FALSE,
+             col.names=TRUE, sep='\t', quote=FALSE)
```

In any case, this dataframe comes with the package `SamplingStrata`: it can be made available by executing:

```
> library(SamplingStrata)
> data(swissframe)
> head(swissframe)
```

	progr	REG	X1	X2	X3	X4	X5	X6	id	Y1	Y2
1	1	4	18	3	2	1	3	3	Zurich	57324	131422
2	2	1	17	1	1	1	3	2	Geneve	32429	60074
3	3	3	17	1	1	1	3	3	Basel	28161	50349
4	4	2	17	2	3	1	3	3	Bern	19399	44263
5	5	1	17	2	2	1	3	2	Lausanne	24291	44202
6	6	4	16	3	3	1	3	3	Winterthur	18942	28958

	Y3	Y4	domainvalue
1	108178	66349	4
2	57063	28398	1
3	53734	34314	3
4	39397	25575	2
5	35421	21000	1
6	27696	14887	4

4 Construction of atomic strata and association of the information related to target variables (steps 3 and 4)

The **strata** dataframe reports information regarding each stratum in the population. There is one row for each stratum. The total number of strata is given by the number of different combinations of Xs values in the frame. For each stratum, the following information is required:

1. the identifier of the stratum (named 'stratum' or 'strato'), concatenation of the values of the Xs variables;
2. the values of the m auxiliary variables (named from X1 to Xm) corresponding to those in the frame;
3. the total number of units in the population (named 'N');
4. a flag (named 'cens') indicating if the stratum is to be censused (=1) or sampled (=0);
5. a variable indicating the cost of interviewing per unit in the stratum (named 'cost');
6. for each target variable y, its mean and standard deviation, named respectively 'Mi' and 'Si');
7. the value of the domain of interest to which the stratum belongs (named 'DOM1' and corresponding to variable 'domainvalue' in the frame dataframe).

For example:

```
> data(strata)
> head(strata)
```

	stratum	N	X1	X2	X3	M1	M2	S1
1	1	2246	x11	x21	x31	148.1598	443.0137	95.41435
2	2	2972	x11	x21	x32	184.2041	513.8995	81.26956
3	3	1905	x11	x22	x31	193.8927	488.8046	79.66667
4	4	3125	x11	x22	x32	181.3437	597.1925	82.77032
5	5	1733	x12	x21	x31	109.9850	418.2234	88.20289
6	6	1060	x12	x21	x32	114.7943	489.8292	52.71574

	S2	cens	cost	DOM1
1	202.4569	0	1	tot
2	214.9999	0	1	tot
3	261.1876	0	1	tot
4	226.5086	0	1	tot
5	179.1571	0	1	tot
6	166.0292	0	1	tot

If in the `frame` dataframe are also present the values of the target Y variables (from a census, or from administrative data), it is possible to automatically generate the `strata` dataframe by invoking the `buildStrataDF` function. Let us consider again the `swissframe` dataframe that we have in built in previous steps. First, we have to read it from the external file that we had created:

```
> frame <- read.delim('swissframe.txt')
```

Then we can execute the function:

```
> swissstrata <- buildStrataDF(swissframe)
```

Computations have been done on population data

The function takes as unique argument the name of the frame, and also writes out in the working directory the strata file, always named 'strata.txt'. This is the structure of the created dataframe:

```
> str(swissstrata)
```

```
'data.frame':      641 obs. of  19 variables:
 $ STRATO: Factor w/ 295 levels "1*1*1*1*1",...: 1 2 3 5 7 8 9 10 12 14 ...
 $ N      : int   184 1 2 11 9 8 1 1 1 1 ...
 $ M1     : num   48.3 98 57 77.7 58.2 ...
 $ M2     : num   49.4 106 64 81.2 61.6 ...
 $ M3     : num   61.4 116 70 92.4 66.8 ...
 $ M4     : num   28.4 43 50 47 36.2 ...
 $ S1     : num   26.8 0 4 15.2 25.5 ...
 $ S2     : num   28.5 0 0 18.7 20.3 ...
```



```

$ S3      : num  32.6 0 1 17 24.9 ...
$ S4      : num  14.6 0 15 11.1 15.5 ...
$ cost    : int   1 1 1 1 1 1 1 1 1 1 ...
$ cens    : int   0 0 0 0 0 0 0 0 0 0 ...
$ DOM1    : int   1 1 1 1 1 1 1 1 1 1 ...
$ X1      : int   1 1 1 1 1 1 1 1 1 10 ...
$ X2      : int   1 1 1 1 2 2 2 2 3 1 ...
$ X3      : int   1 1 1 2 1 1 1 2 1 1 ...
$ X4      : int   1 1 2 1 1 2 3 1 2 1 ...
$ X5      : int   1 1 1 1 1 1 1 1 1 1 ...
$ X6      : int   1 2 1 1 1 1 1 1 1 1 ...

```

It is worth while to note that the total number of different atomic strata is 641, lower than the dimension of the Cartesian product of the Xs (which is 4374): this is due to the fact that not all combinations of the value of the auxiliary variables are present in the sampling frame. Variables 'cost' and 'cens' are initialised respectively to 1 and 0 for all strata. It is possible to give them different values:

1. for variable 'cost', it is possible to differentiate the cost of interviewing per unit by assigning real values;
2. for variable 'cens', it is possible to set it equal to 1 for all strata that are of the 'take-all' type (i.e. all units in that strata must be selected).

The **swissstrata** dataframe comes together with **SamplingStrata** package, it can be made available by typing:

```
> data(swissstrata)
```

On the contrary, if there is no information in the frame regarding the target variables, it is necessary to build the strata dataframe starting from other sources, for instance a previous round of the same survey, or from other surveys. In this case, we need to read sample data by executing:

```
> samp <- read.delim('samplePrev.txt')
```

The only difference is that computed mean and variances of the Ys are sampling estimates, whose reliability should be evaluated by carefully considering their sampling variances. In addition to the naming constraints previously introduced, this case requires that a variable named 'WEIGHT' is present in the samp dataframe. Then we can execute this function in this way:

```
> strata <- buildStrataDF(samp)
```

The result is much the same than in the previous case: the function creates a new dataframe, **strata**, and writes out in the working directory the strata file, named 'strata.txt'.

Note that in all cases, for each target variable Y, mean and standard deviation are calculated excluding NAs.

5 Choice of the precision constraints for each target estimate (step 5)

The `errors` dataframe contains the accuracy constraints that are set on target estimates. This means to define a maximum coefficient of variation for each variable and for each domain value. Each row of this frame is related to accuracy constraints in a particular subdomain of interest, identified by the `DOM1` value. In the case of the Swiss municipalities, we have chosen to define the following constraints:

```
> data(swiserrors)
> swiserrors
```

	DOM	CV1	CV2	CV3	CV4	domainvalue
1	DOM1	0.08	0.12	0.08	0.12	1
2	DOM1	0.08	0.12	0.08	0.12	2
3	DOM1	0.08	0.12	0.08	0.12	3
4	DOM1	0.08	0.12	0.08	0.12	4
5	DOM1	0.08	0.12	0.08	0.12	5
6	DOM1	0.08	0.12	0.08	0.12	6
7	DOM1	0.08	0.12	0.08	0.12	7

This example reports accuracy constraints on variables `Y1`, `Y2`, `Y3` and `Y4` that are the same for all the 7 different subdomains (Swiss regions) of domain level `DOM1`. Of course we can differentiate the precision constraints region by region. It is important to underline that the values of 'domainvalue' are the same than those in the `frame` dataframe, and correspond to the values of variable 'DOM1' in the `strata` dataframe. Once having defined dataframes containing frame data, strata information and precision constraints, it is worth while to check their internal and reciprocal coherence. It is possible to do that by using the function `checkInput`:

```
> checkInput(swiserrors,swissstrata,swissframe)
```

Input data have been checked and are compliant with requirements

For instance, this function controls that the number of auxiliary variables is the same in the `frame` and in the `strata` dataframes; that the number of target variables indicated in the `frame` dataframe is the same than the number of means and standard deviations in the `strata` dataframe, and the same than the number of coefficient of variations indicated in the `errors` dataframe.

If we try to determine the total size of the sample required to satisfy these precision constraints, considering the current stratification of the frame (the 641 atomic strata), we can do it by simply using the function `bethel`. This function requires a slightly different specification of the constraints dataframe, that is contained in the `errors.chk` dataframe:

```
> data(swiserrors.chk)
> swiserrors.chk

      DOM  CV1  CV2  CV3  CV4
1 DOM1 0.08 0.12 0.08 0.12
```

This is a more compact way to report the precision constraints for all subdomains. Of course, it does not permit to differentiate them by subdomain. In any case, the result of the application of the Bethel algorithm (see Bethel (1989)) is:

```
> sum(bethel(swisstrata,swiserrors.chk))

[1] 893
```

That is, the required amount of units to be selected, with no optimization of sampling strata. In general, after the optimization, this number is sensibly reduced.

6 Optimisation of frame stratification (step 6)

Once the strata and the constraints dataframes have been prepared, it is possible to apply the function that optimises the stratification of the frame, that is `optimizeStrata`. This function operates on all subdomains, identifying the best solution for each one of them. The fundamental parameters to be passed to `optimizeStrata` are:

1. **errors**: the (mandatory) dataframe containing the precision levels expressed in terms of maximum allowable coefficients of variation that regard the estimates on target variables Ys of the survey
2. **strata**: the (mandatory) dataframe containing the information related to 'atomic' strata, i.e. the strata obtained by the Cartesian product of all auxiliary variables Xs. Information concerns the identifiability of strata (values of Xs) and variability of Ys (for each Y, mean and standard error in strata)
3. **cens**: the (optional) dataframe containing the takeall strata, those strata whose units must be selected in whatever sample. It has same structure than **strata** dataframe
4. **strcens**: flag (TRUE/FALSE) to indicate if takeall strata do exist or not. Default is FALSE
5. **initialStrata**: the initial limit on the number of strata for each solution. Default is 3000
6. **addStrataFactor**: this parameter indicates the probability that at each mutation the number of strata may increase with respect to the current value. Default is 0.01

7. **minnumstr**: indicates the minimum number of units that must be allocated in each stratum. Default is 2
8. **iter** Indicated the maximum number of iterations (= generations) of the genetic algorithm. Default is 20
9. **pops** The dimension of each generations in terms of individuals. Default is 50
10. **mut_chance** (mutation chance): for each new individual, the probability to change each single chromosome, i.e. one bit of the solution vector. High values of this parameter allow a deeper exploration of the solution space, but a slower convergence, while low values permit a faster convergence, but the final solution can be distant from the optimal one. Default is 0.05
11. **elitism_rate**: this parameter indicates the rate of better solutions that must be preserved from one generation to another. Default is 0.2 (20
12. **highvalue**: parameter for genetic algorithm. Its default value should not be changed
13. **suggestions**: optional parameter for genetic algorithm that indicates one possible solution (maybe from previous runs) that will be introduced in the initial population. Default is NULL.

In the case of the Swiss municipalities, this is a possible choice of the value of the parameters:

```
> outstrata <- optimizeStrata(
+   errors = swisserrors,
+   strata = swissstrata,
+   cens = NULL,
+   strcens = FALSE,
+   initialStrata = 3000,
+   addStrataFactor = 0.00,
+   minnumstr = 2,
+   iter = 30,
+   pops = 5,
+   mut_chance = 0.05,
+   elitism_rate = 0.2,
+   highvalue = 1e+08,
+   suggestions = NULL
+ )
```

Testing the sanity of parameters...

The start conditions:

GA Settings

Type	=
Population size	= 5

```

Number of Generations = 30
Elitism               = 1
Mutation Chance       = 0.05

```

Starting with random values in the given domains...
Starting iteration 1
Calculating evaluation values...

The execution of `optimizeStrata` produces the solution of 7 different optimization problems, one for each domain (see fig.1).

7 Analysis of results (step 7)

Once the optimisation has been carried out, it is important to analyse the structure of the new strata. By using the function `verify` it is possible (i) to perform a graphical inspection and (ii) to verify how the allocation of units in the new strata ensures the compliance of accuracy constraints in the different domains for the whole set of target variables. By executing:

```

> data(swisserrors.chk)
> pdf("distributionplot.pdf")
> allocation <- verify(swisserrors.chk,outstrata)
> sum(allocation)
> dev.off()

```

we will obtain first a graph containing the distribution of strata by numerosity in the different subdomains (see fig.2).

In this graph, it is possible to click on the different points and, after stopping the identification, selected strata are presented in an edit window for the inspection. We obtain also the the list of domain levels, domain values and variables, and related planned and expected CV, together with the 'sensitivity' (the expected saving in sample size if the CV is relaxed of a 10). This list relates only to the national level, as with the dataframe `errors.chk` we do not refer to the different domains (regions), as we do with the dataframe `errors`. Moreover, we want also to analyse what kind of aggregation of the atomic strata the genetic algorithm did produce. To do so, we apply the function `updateStrata`, that assigns the labels of the new strata to the initial one in the dataframe `strata`, and produces:

1. a new file named 'newstrata.txt' containing all the information in the strata dataframe, plus the labels of the new strata;
2. a table, contained in the dataset 'strata_aggregation.txt', showing in which way the auxiliary variables `Xs` determine the new strata.

The function is invoked in this way:

```

> newstrata <- updateStrata(swisstrata)

```

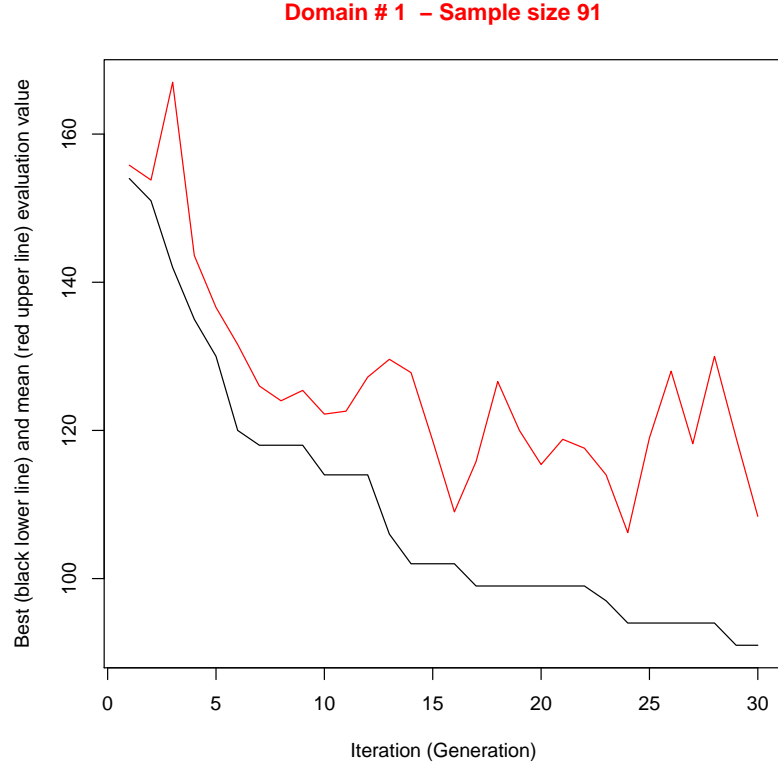


Figure 1: this graph illustrates the convergence of the solution to the final one starting from the initial one (i.e. the one related to the atomic strata). Along the x-axis are reported the executed iterations, from 1 to the maximum, while on the y-axis are reported the size of the sample required to satisfy precision constraints. The upper (red) line represent the average sample size for each iteration, while the lower (black) line represents the best solution found until the i -th iteration.

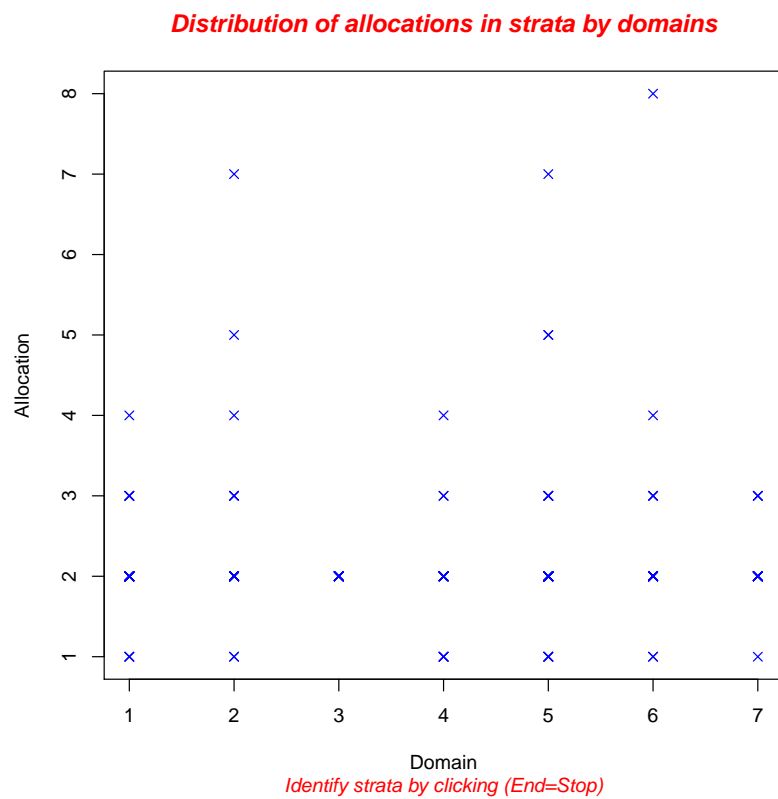


Figure 2: in this graph along the x-axis are reported the different domains (regions), for each of which are listed the associated strata, with the corresponding allocation (reported on the y-axis).

8 Updating the frame and selecting the sample (steps 8 and 9)

Once the optimal stratification has been obtained, to be operational we need to accomplish the following two steps:

1. to update the frame units with new stratum labels (combination of the new values of the auxiliary variables Xs);
2. to select the sample from the frame.

As for the first, we execute the following command:

```
> framenew <- updateFrame(swissframe,newstrata)
```

The function **updateFrame** receives as arguments the indication of the dataframe in which the frame information is memorised, and of the dataframe produced by the execution of the **updateStrata** function. The execution of this function produces a dataframe **framenew**, and also a file (named 'framenew.txt') with the labels of the new strata produced by the optimisation step. The allocation of units is contained in the 'soluz' column of the dataset 'outstrata.txt'. At this point it is possible to select the sample from the new version of the frame:

```
> sample <- selectSample(framenew,outstrata)
```

that produces two .csv files:

1. 'sample.csv' containing the units of the frame that have been selected, together with the weight that has been calculated for each one of them;
2. 'sample.chk.csv' containing information on the selection: for each stratum, the number of units in the population, the planned sample, the number of selected units, the sum of their weights that must equalise the number of units in the population.

References

- Bethel, J. (1989). Sample allocation in multivariate surveys. *Survey Methodology* 15, 47–57.
- Hartigan, J. A. and M. A. Wong (1979). A k-means clustering algorithm. *Applied Statistics* 28, 100–108.
- Willighagen, E. (2005). *genalg: R Based Genetic Algorithm*. R package version 0.1.1.