CONTRIBUTI ISTAT

n. 10/2008

Optimal stratification of sampling frames in a multivariate and multidomain sample design

M. Ballin e G. Barcaroli

Istat

SISTEMA STATISTICO NAZIONALE
ISTITUTO NAZIONALE DI STATISTICA

Le collane esistenti presso l'ISTAT - *Rivista di Statistica Ufficiale*, *Contributi ISTAT* e *Documenti ISTAT* - costituiscono strumenti per promuovere e valorizzare l'attività di ricerca e per diffondere i risultati degli studi svolti, in materia di statistica ufficiale, all'interno dell' ISTAT, del SISTAN, o da studiosi esterni.

La *Rivista di Statistica Ufficiale* accoglie lavori che hanno come oggetto la misurazione dei fenomeni economici, sociali, demografici e ambientali, la costruzione di sistemi informativi e di indicatori, le questioni di natura metodologica, tecnologica o istituzionale connesse al funzionamento dei sistemi statistici e al perseguimento dei fini della statistica ufficiale.

I lavori pubblicati in *Contributi ISTAT* sono diffusi allo scopo di stimolare il dibattito intorno ai risultati preliminari di ricerca in corso.

I *Documenti ISTAT* forniscono indicazioni su linee, progressi e miglioramenti di prodotto e di processo che caratterizzano l'attività dell'Istituto.

Il Comitato di redazione esamina le proposte di lavori da pubblicare nelle tre collane sopra indicate. Quelli pubblicati nei Contributi ISTAT e nei Documenti ISTAT sono valutati preventivamente dai dirigenti dell'Istituto, mentre i lavori pubblicati nella Rivista di Statistica Ufficiale sono subordinati al giudizio di referee esterni.

ISTAT – Produzione libraria e centro stampa

CONTRIBUTI ISTAT

n. 10/2008

Optimal stratification of sampling frames in a multivariate and multidomain sample design

M. Ballin(*) e G. Barcaroli(**)

(*) ISTAT - Direzione centrale statistiche strutturali
(**) ISTAT - Direzione centrale tecnologie e supporto metodologico

Contributi e Documenti Istat 2008

Istituto Nazionale di Statistica
Servizio Produzione Editoriale

Produzione libraria e centro stampa:
Carla Pecorario
Via Tuscolana, 1788 - 00173 Roma

Sommario
Fino ad oggi, nel disegno dei campioni stratificati il problema della determinazione della dimensione ottimale, e della corrispondente allocazione delle unità campionarie negli strati, è stato risolto considerando la particolare stratificazione della popolazione come un elemento dato; e, nello stesso tempo, la definizione della stratificazione ottimale di una data popolazione è stata investigata senza considerare il problema di ottimizzazione della dimensione del campione dell'indagine di interesse. Nel recente passato, una interessante proposta è stata avanzata, riguardante la soluzione congiunta di entrambi i problemi: tale proposta è basata sulla costruzione di un albero di classificazione per l'esplorazione delle possibili soluzioni (stratificazioni alternative della popolazione), ognuna delle quali viene valutata risolvendo il corrispondente problema di allocazione multivariata mediante l'algoritmo di Bethel-Chromy. Nel presente lavoro proponiamo un approccio evolutivo non deterministico, basato sull'uso dell'algoritmo genetico. Entrambi i metodi sono stati applicati ad un caso reale di studio, quello dell'indagine sulla Struttura della Produzione Agricola, ed i risultati sono stati analizzati raffrontandoli alla soluzione adottata per l'indagine del 2003.

Abstract
So far, in stratified sampling the problem of determining the optimal size and allocation of units in strata has been solved considering the stratification of population as given; and, conversely, the definition of an optimal stratification has been investigated without considering the optimisation problem of sampling size and allocation. An interesting proposal has been advanced in the recent past, concerning the joint solution of both problems: it is based on a tree search in the space of possible strata configurations, solving for each visited node the corresponding multivariate allocation problem by applying the Bethel-Chromy algorithm. Here we propose a non deterministic evolutionary approach, making use of the genetic algorithm paradigm. Having implemented both methods, we applied them to a real sampling survey (the Italian Farm Structure Survey), and evaluated the obtained results, by comparing them to the ones related to solution adopted in 2003 survey.

Keywords: sampling design; multivariate allocation; optimal stratification; genetic algorithm

1. Introduction

The optimality of a sample design can be defined in terms of costs (associated to fieldwork: number of units to be interviewed) and accuracy (sampling variance related to target estimates). Bethel proposed an algorithm (Bethel, 1985) able to determine total sample size and allocation of units in strata, so to minimise costs under the constraints of defined precision levels of estimates, in the multivariate case (more than one estimate). Input to this algorithm is given by the information on distributional characteristics (total and variance) of target variables in the population strata. Under this approach, population stratification, i.e. the partition of the sampling frame obtained by cross-classifying units by means of potential stratification variables, is given. But stratification has a great impact on the optimal solution determined by Bethel algorithm and, in general, it must be defined in the first steps of a survey planning.

If a frame with a set of potential variables for stratification is available, the survey planner has to choose the "best" auxiliary variables cross product (partition of the frame). Among the possible partitions, the one with the maximum number of strata, given by the Cartesian product of all auxiliary variables, does not always yield the optimal sample size. In fact, organisational considerations, and the necessity to define a minimum amount of units per stratum, oblige not to increase the number of strata beyond a certain limit. In that case, how to determine the best partition among all partitions obtainable combining the auxiliary variables (what auxiliary variables? what values for each of them to take into consideration?) has to be considered as a part of the whole problem.

Until recently, on the contrary, the problem of determining the optimal size and allocation of units in strata has been solved considering the stratification of population as given; and, conversely, the definition of an optimal stratification has been investigated independently by the optimisation problem of sampling size and allocation.

An interesting proposal has been advanced in the recent past (Benedetti *et al* 2005), offering a joint solution to both problems: it is based on a tree search in the space of possible strata configurations, solving for each visited node the corresponding multivariate allocation problem accordingly to Bethel algorithm. At each level, the node that is the best in terms of sample size reduction, is chosen as the branching node. This tree-based approach is deterministic and very fast, but it may heavily suffer for the presence of local minima and, consequently, solutions can be far from optimality.

Together with this tree-based approach, we propose a non deterministic evolutionary approach, based on the genetic algorithm (GA) paradigm. Under the GA approach, each solution (i.e. a particular partition in strata of the sampling frame) is an individual in a population, whose fitness is evaluated by calculating the sampling size satisfying accuracy constraints on the target estimates; crossover and mutation carried out along each iteration ensure an increase of average fitness.

In general, the characteristic of GA are such that the risk of local minima is lower than in the tree search, though processing time is noticeably higher. Our proposal is the following: in complex situations (characterised by a high number of stratification alternative configurations and/or a high number of target variables and domains), first the tree-based algorithm is applied, in order to individuate a solution. This solution is then introduced in the GA initial population, in order to speed its convergence to a better solution. Our experiments show an improvement of the tree-based solution, and encourage the adoption of this procedure.

This paper is structured as follows: in paragraph 2, main questions related to sample design in an official statistics environment are dealt with. Paragraph 3 briefly reports the Bethel algorithm in the Chromy version, extended to the multidomain case. In paragraph 4 the tree-based algorithm is reported, while the GA approach is described in paragraph 5. Paragraph 6 reports and analyses the results of the joint application of both algorithms to the Italian Farm Structure Survey. Possible future work and final conclusions are in paragraph 7

2. Stratified sampling designs in practical cases

Stratified sampling design is certainly one of the most widely used in the context of surveys on

enterprises and on farms carried out by ISTAT.

For such sampling design, given a frame containing the list of units forming the population of interest and some auxiliary variables useful to classify such units, the survey planner have to implement the following steps:

- construction of strata (partition of the population of interest);
- choice of sampling methods, sample size and its allocation among the strata;
- choice of estimation methods.

Among the problem that must be solved in implementing such steps should be reminded:

- which auxiliary variables should be used to stratify the population?
- how many strata should there be?
- how many units should be allocated in each stratum?

Even if it is well recognised that the solutions given to each problem influence each other, the experience shows that in most cases they are treated independently.

Furthermore, while for the problem of sample size and its allocation among strata some efficient solutions are well known and software implementing such solutions are available for both univariate (Cochran, 1977, pag 96-99) and multivariate case, on the contrary, concerning the optimal stratification solutions were well known only for quite simple cases (Cochran, 1977, pag. 127).

Recently, solutions for the choice of the boundaries of strata, when stratification variables are of continuous type, have been given for more complex cases (Keskinturk and Sebnem 2007); a recent very general approach to the description of these solutions can be found in Kozak,Verma and Zieliński (2007).

This last approach is based on the definition of an objective function $f(\mathbf{a})$ that have to be minimized under two (possible) set of constraints

$$g(\mathbf{a})=G_p, \qquad p=1,\ldots P$$
$$h(\mathbf{a})\leq H_q, \qquad q=1,\ldots Q$$

where $\mathbf{a}$ is a set of parameters that univocally determine the boundaries of the strata, and G_p and H_q are constants. In their paper they show as the parameters $\mathbf{a}$, the function f and the constraints have to be specified to deal with particular stratification problems or to describe some of the solutions proposed in the literature.

Nevertheless, in practical cases qualitative variables, with pre-defined sets of possible values, are often available for partitioning the population in strata: these cases are usually managed by "hand" by comparing some alternative solutions. Solutions are commonly obtained by applying some criteria to the entire population. For example, the entire population is split on the basis of geographical localization, dimension of units (number of employees, size of farms in term of agricultural areas, etc.), sector of activities. Different stratifications are then obtained by changing class of dimensions, by grouping sector of activities , etc.

A deep analysis of the interaction among the variables used for stratification and their effects on sample size (e.g. classification by size must be used for each sector of activity? should be considered the sector of interest in each geographical area?) is very demanding because of the number of possible cases that is usually very high. This analysis, as it will be shown later, has been carried out "by hand" in the case of the Italian Farm Structure Survey (FSS).

In this paper, as it has been underlined above, it is proposed an effective approach to deal with the problem of choice among competitive stratifications based on qualitative auxiliary variables. Under this approach, originally continuous variables must be transformed in categorical.

3. The Bethel approach for multivariate multidomain allocation

Given a stratification of P in H strata, let N_h and $S^2_{h,g}$, h=1,..,H, g=1,...,G respectively the population size and variances of the Ys variables in each stratum. Assuming a simple random sampling of n_h units without replacement in each stratum, the variance of the total estimate for Y_g is

$$V(Y_g)=\sum_{h=1}^{H} N_h^2\left(1-\frac{n_h}{N_h}\right)\frac{S_{h,g}^2}{n_h} \quad g=1,..,G$$

Consider the following cost function

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

where C_0 indicates a fixed cost (not dependent on sample size) and C_h represents the cost of observing a unit in the stratum *h*.

Assume U_g , g=1,..,G, to be the upper level for the desired precision of the estimates (expressed in terms of variance).

The optimal multivariate allocation problem can be defined as the search for the solution of the minimum (with respect to n_h) of linear function *C* under the convex constraints $V(Y_g)\le U_g$ g=1,..,G.

Bethel (1989) suggested that the problem can be solved easier considering the following function of n_h

$$x_h=\begin{cases}1/n_h & \text{if } n_h\ge 1\\ \infty & \text{otherwise}\end{cases}.$$

In fact, using x_h instead of n_h the cost function can be written as

$$C(x_1,..,x_H)=C_0+\sum_{h=1}^{H}\frac{C_h}{x_h},$$

and the variances as

$$V(Y_g)=\sum_{h=1}^{H} N_h^2\left(1-\frac{1}{x_h N_h}\right)S_{h,g}^2 x_h \qquad g=1,...,G.$$
$$=\sum_{h=1}^{H} N_h^2 S_{h,g}^2 x_h - N_h S_{h,g}^2$$

Consequently, the multivariate allocation problem can be defined as the search for the minimum (with respect to x_h) of the convex function *C* under a set of linear constraints

$$\sum_{h=1}^{H} N_h^2 S_{h,g}^2 x_h - N_h S_{h,g}^2 \le U_g. \qquad g=1,...,G$$

The equivalent problem can be defined if the constraints are defined in term of coefficient of variations.

An algorithm, that is proved to converge to the solution (if it exists), was provided by Bethel by applying the Lagrangian multipliers method to this problem. Chromy (1987) suggested an improvement to this algorithm in order to increase its effciency.

It should be noted that the same approach can be used to deal with the multidomain problem. In fact, consider the usual transformation for the domain estimation problem (Särndal et al, 1992, pag 29)

$$Y_i^d=\begin{cases}Y_i & \text{if the unit } i \text{ belongs to domain } d\\ 0 & \text{otherwise}\end{cases},$$

then, if the quantities previously defined to describe the Bethel approach are computed using the variables Y^d d=1,…,*D* then the multivariate allocation solution is the solution for the multidomain case.

In ISTAT the survey carried on 1995 for the Intermediate Census on Enterprises and Services represent the very first experience of this approach (Ballin M., Falorsi P.D., Vicard P., 2000). A software based on SAS routines was successively developed and it is available on the Istat web site with a manual for users[1]. A detailed description of the solutions adopted in the software can be found in Falorsi *et al* (1998)

[1] ISTAT supplies a software implementing an algorithm for the optimal solution in the case of multivariate and multidomain problem (Di Giuseppe *et al* 2004)

4. The tree-based approach

4.1 The Tree-based Algorithm

Given a finite population P of *N* units, and a survey on P in order to produce estimates on totals of $Y_1,\ldots Y_G$ variables, with given acceptable standard errors, let us consider the problem of finding a stratification S on P such that its corresponding overall best allocation is minimum with respect to all other possible stratifications on P, where the best allocation is calculated for each stratification with the Bethel-Chromy algorithm.

Each alternative stratification is a partition of P obtained by classifying units in P by means of a set of auxiliary variables $X_1,\ldots, X_t$, each one characterised by a given domain of values.

Given t variables $X_1,\ldots, X_t$, with domain sets $D_i = \{x_{i1},\ldots,x_{im_i}\}$ (i=1,…, t), we can represent a solution by means of a vector $\mathbf{v} = [v_1, \ldots, v_M]$ of cardinality $M = \sum_{k=1}^{t} m_k$, whose elements v_j can assume 1 or 0 values.

A 1 in the *j*-th position, where

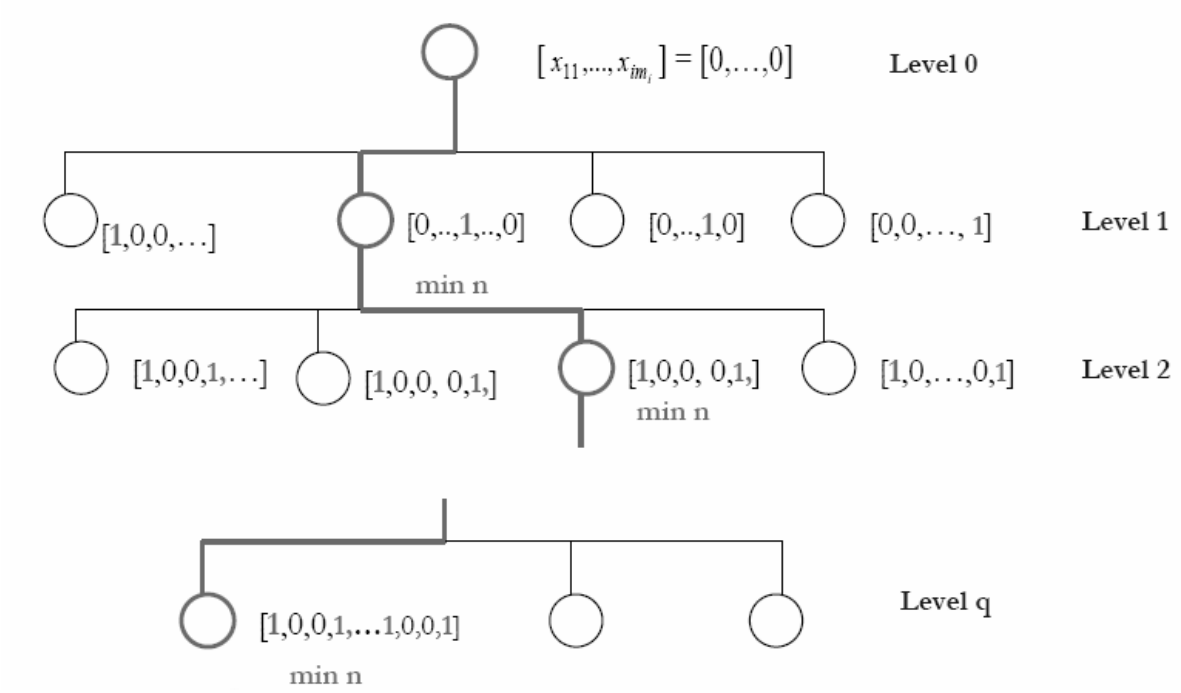
$$j = (\sum_{p=1}^{i-1} m_p) + q \qquad (1 < q < m_i)$$

indicates that the *q*-th value of the *i*-th variable is active in generating strata, while a 0 in that position indicates that this value is not active.

It is possible to restrict the space of alternative stratifications by setting the constraint of a maximum number of strata. Moreover, estimates of totals and of variances of $Y_1,\ldots,Y_G$ variables are available for most available detailed stratification, i.e. for *basic* or *atomic strata*.

The proposed procedure takes into account a set of alternative stratifications, starting from the most aggregate one (i.e. only one stratum coinciding with the whole population), and generating a tree by using a splitting rule such that for any given level a generating node is chosen in such a way that the decrease of overall sample size is maximised.

In the graph representing the tree, for each node the *level* is the minimum number of arcs connecting that node to the root.



We can represent the algorithm by means of the following steps:

1. initialisation: the node associated to the stratification characterised by a unique stratum, coinciding with the whole population, is the root of the tree (level k = 0), and is set as *generating node*;
2. from the generating node at level k, "child" nodes of level (k+1) are generated, by on turn activating a single value of the vector $\mathbf{v}=[v_1,...,v_M]$ among those not yet activated;
3. at level (k+1), the overall sample size n is calculated with the Bethel-Chromy algorithm for each node in the level. The node with the minimum *n* is set as *generating node*;
4. stopping rule: steps 2 and 3 are repeated until:
   - the maximum acceptable number of strata has been reached (the activation of new values in Xs domains increases the number of resulting strata);
   - the gain in terms of reduction of the overall sample size becomes negligible.

Best solution is then selected by considering the one associated to the generating node of the immediately preceding level.

The above algorithm is different from the one proposed by Benedetti *et al* (2005). In fact, the latter is built with the same rules followed for the creation of *classification* trees, only changing the optimisation criterion: while for pure classification trees it is the minimisation of dependent variable's variance within nodes, here it is the maximisation of sampling size reduction. As in the case of classification trees, splitting is performed analysing possible splits variable by variable and considering the gains. Also in our proposal the optimisation criterion is the maximisation of sampling size reduction, but splitting is performed by considering possible gains value by value for all variables.

4.2 Implementation details

The whole procedure has been developed by using the R system (R Development Core Team 2007). It is composed by four different modules:

1. treeCycle.R, containing the definition of the function "treeCycle";
2. strataTree.R: containing the definition of the function "strataTree";
3. Bethel.R: in this script is contained the definition of the function that implements the Bethel-Chromy algorithm for the solution of the optimal multivariate multidomain allocation problem;
4. makeData.R: this script defines the function makeData, which prepares the input for the Bethel-Chromy algorithm.

The main function ("treeCycle") manages the whole flow of processing in case should it be done iteratively for different domains. The current version allows to handle only one domain type. It requires the following inputs:

- the desired levels of precision (expressed in terms of coefficients of variation) for each target variable Y in each defined domain;
- the list of basic sampling strata, with the indication, for each stratum, of Y's means and standard deviations, population, values of X's variables.

Also in this case it is possible to give an additional strata dataset, containing information on *take-all* strata, that will be handled differently from sampling strata.

In case no domains are given, it is possible to ignore the function "treeCycle" and execute directly the function "strataTree".

General parameters are:

1. the minimum number of units to be allocated in each stratum.
2. the maximum number of strata in the final solution (first stopping rule),
3. a delta value to be compared with the gain in terms of sample size obtained by passing to a deeper level in the tree: if this gain is lower than the delta value, the algorithm ends (second stopping rule).

The first operation consists in reading input data and processing auxiliary X variables to build the solution vector, that in this case is initialised to all 0s.

This first solution, corresponding to the root node of the tree (level 0), is then passed to the "makeData" function. This binary vector is used to process Xs variables in order to aggregate basic strata.

Means and standard deviations of Ys required by Bethel-Chromy algorithm are computed on the basis of input information for basic data, and associated to current stratification. In particular, means and standard deviations in each aggregate stratum can be dynamically calculated starting from means and standard deviations in most detailed strata (*basic* or *atomic strata*), which are the general input for the optimal stratification problem.

If we call *basic strata* the most detailed strata, each given stratum of a solution can be obtained as an aggregation of a subset of basic strata: we indicate the basic strata in this subset as *component basic strata* of the current aggregate stratum..

So, for each aggregate stratum of the current solution, and for each Y, we can calculate:

- Y *total* as the sum of totals in component basic strata;
- Y *variance* as the sum of variances in basic strata, plus the sum of deviances of means related to component basic strata from the mean in aggregate stratum.

Having computed totals and variances for each Y in each stratum, it is possible to derive mean and standard deviations that are usual input to the multivariate allocation problem.

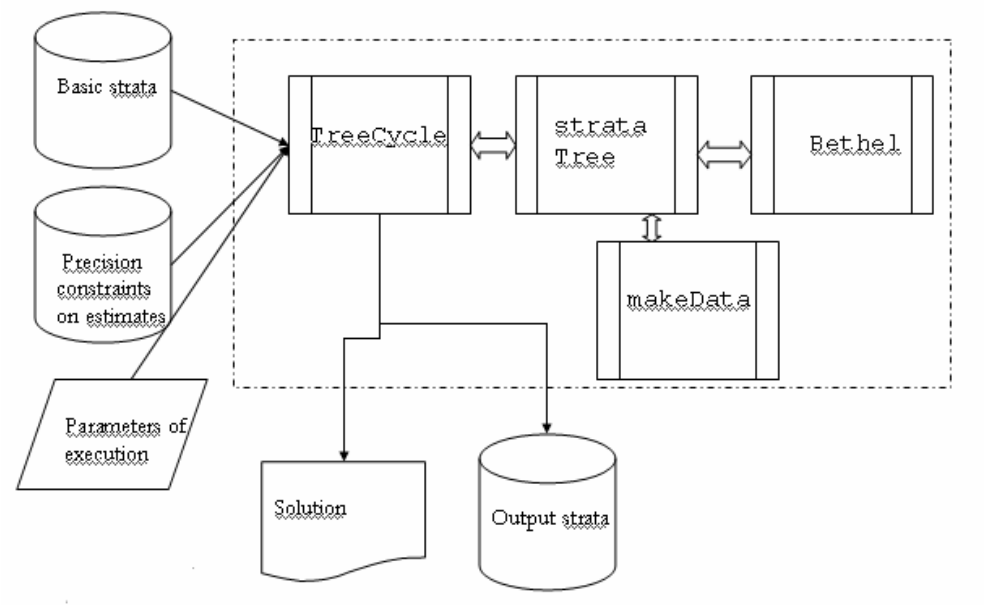
In this way, the information necessary to solve optimal multivariate allocation problem is ready, and passed to function "Bethel"[2], that provides to return a vector of integers, corresponding to units allocated to different strata of the current solution. The total sample size is then computed, and associated to the current node.

This process is repeated for each node in the current level of the tree, in order to choose the generating node.

Generations are carried out until one of the two stopping rules is verified: either the maximum number of strata defined by the corresponding parameter is reached, or the gain in terms of sample size is reduced below the given delta.

This operations can be monitored, and iteration results graphically displayed.

Once stopped, the program outputs the best solution, with its associates number of strata, total sample size, and information on values of Xs utilised to aggregate strata. Finally, a dataset containing aggregate strata corresponding to the best solution are written to an ASCII file, containing also, for each stratum, the number of allocated units.



[2] Bethel function has been implemented in two versions: a first one is entirely in R code, while a second one makes use of FORTRAN compiled code in order to increase computational efficiency in executing the core of the algorithm

5. The genetic algorithm approach

5.1 The genetic algorithm (GA)

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimisation and search problems. Genetic algorithms are a particular class of evolutionary algorithms (also known as *evolutionary computation*) that use techniques inspired by evolutionary bi logy such as *inheritance*, *mutation*, *selection* and *crossover* (also called *recombination*) (Vose 1999) (Schmitt 2001, 2004).

Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called *chromosomes* or the *genotype* or the *genome*) of candidate solutions (called *individuals* or *phenotypes*) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The *evolution* usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (privileging those with higher *fitness*), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

In general, genetic algorithms are better than gradient search methods if search space has many local optima. Since the genetic algorithm traverses the search space using the genotype rather than the phenotype, it is less likely to get stuck on a local high or low.

5.2 Application of GA to optimal stratification problem

A typical genetic algorithm requires two basic elements to be defined:

1. a genetic representation of the solution domain;
2. a fitness function to evaluate the solution domain.

In our case, a solution is given by a particular stratification, i.e. a partition of the population frame. This partition is individuated by the set of variables that have been used to produce it. Each variable is on turn characterised by a domain, i.e. a set of acceptable values.

Given a population frame, all X variables whose values are available for each unit in the frame are candidate for individuating the optimal stratification: for each candidate variable, all its possible values are candidate likewise.

The most disaggregate solution is obtained by considering the complete Cartesian product of the domain sets of all X variables.

The opposite solution is the one consisting in only one stratum, coinciding with the whole population: in this case, no X variable is active in generating strata.

Between these extremes, the candidate solutions can be represented by indicating what values in which variables have to be considered as active in generating strata. Each solution can be represented by the vector $\mathbf{v} = [v_1 \ldots v_M]$ already introduced in the tree-based approach paragraph.

The fitness function is defined over the genetic representation, and measures the *quality* of the represented solution. In our case, a measure of the quality of the solution is measured by related optimal sample size.

Fitness of a given solution is evaluated by solving a multivariate multidomain allocation problem. This is done by considering the following inputs:

1. required precisions for target estimates of variables Ys, for each domain of interest, indicated by means of the coefficients of variations (CVs)
2. means and standard deviations related to Y target variables in each stratum

to which the Bethel algorithm (Chromy version) is applied (Bethel 1985, 1989) (Chromy 1987).

Having defined a suitable representation of candidate solution, and the fitness function to be applied to each solution, in the following we report how GA operates to find the solution.

The first step consists in the initialization of population. First, the program analyses the different values in Xs variables and computes $M = \sum_{k=1}^{t} m_k$, the cardinality of **v** vector. Then, on the basis of the value of the parameter "size of population[3]", *s* different individuals are randomly generated. This generation can be based on an entirely uniform distribution, or conversely it is possible to give indications on the desired proportion between 0s and 1s: a larger incidence of 0s initialises solutions towards the most aggregate extreme, while more 1s initialise population towards the most detailed stratification. We experienced a faster convergence by fixing an initial ratio of 0s with respect to 1s equal to 3 /1.

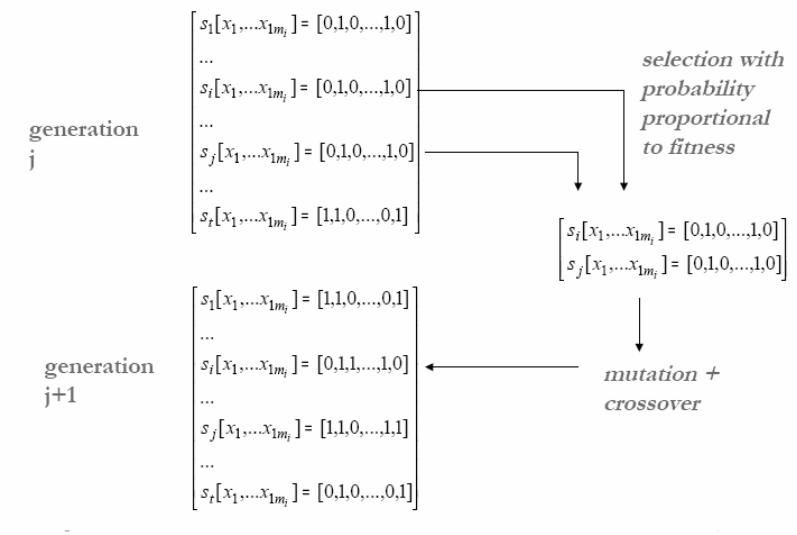
For each individual (solution) in the population, its fitness has to be evaluated, by calculating with the Bethel-Chromy algorithm the associated sample size required to satisfy precision constraints on Ys estimates. As in the case of the tree-based algorithm, means and standard deviations of Y variables in current strata, required as input by Bethel-Chromy algorithm, are computed on the basis of means and standard deviations in basic strata.

Once evaluated the fitness of each individual, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through the *fitness-based* process, where fitter solutions are more likely to be selected, while only a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions.

A second generation population of solutions from those selected is generated through the following genetic operators:

- *crossover*: many crossover techniques exist for GA which use different data structures and different criteria of chromosomes selection, but the general approach is to exchange a subset of chromosomes between two parents.
- *mutation*: given the probability that an arbitrary bit in a genetic sequence will be changed from its original state (*mutation chance*), GA proceeds to draw, for each chromosome in the genoma, a random value to decide if the bit will be changed.

Therefore, for each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool previously selected. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated.



[3] Not to be confused with the size of finite population target of the survey

Usually, the average fitness is increased moving from one generation to the next. The rapidity of this movement towards better solutions greatly depends on the mutation chance. It has to be noticed that a too high value of this parameter can lead to local minima, while a lower value decreases the convergence quickness, but ensures a wider exploration of the solutions space.

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- a solution is found that satisfies minimum criteria;
- fixed number of generations reached;
- the highest ranking solution's fitness is reached or has reached a plateau such that successive iterations no longer produce better results;
- manual inspection;
- combinations of the above.

In our case, the terminating condition is the reached number of iterations.

5.3 Implementation details

The whole procedure has been developed by using the R system, and it resembles very much the one developed for the tree-based algorithm. It is composed by four different modules (the last two have already been introduced in the tree-based approach):

1. genCycle.R, containing the definition of the function "genCycle";
2. strataGenalg.R: containing the definition of the function "strataGenalg";
3. Bethel.R;
4. makeData.R.

As in the tree-based approach, the main function ("genCycle") manages the iteration of the process for different domains. It requires the following inputs:

- desired levels of precision (expressed in terms of coefficients of variation) for each target variable Y in each defined domain of interest;
- the list of basic sampling strata, with the indication, for each stratum, of Ys means and standard deviations, population, values of Xs variables.

It is also possible to give an additional strata dataset, containing information on *take-all* strata, that will be handled differently from sampling strata.

General parameters are:

1. maximum number of strata in the final solution;
2. minimum number of units to be allocated in each stratum.

Package "genalg" is used in order to make use of the Genetic Algorithm (Willighagen, 2005). This package requires the following parameters:

- maximum number of iterations (generations): the stopping rule;
- population size: the number of individuals (solutions) that will be handled at each generation;
- mutation chance: the probability for a chromosome to change value at each iteration;
- ratio between zeroes and ones: the initial desired proportion of 0s and 1s in the individuals of the first generation;
- elitism: the minimum percentage of best individuals that are being carried on from one generation to the next without modifications.

After having stored input data, main program analyses X variables, performs a convenient transformation of their values, then calls the main function of "genalg" package, that is "rbga.bin"[4] (that stands for "R Based Genetic Algorithm – binary"). This function initialises the population accordingly to the population size and the zero-to-one ratio parameters, then it calls the evaluation function for each individual in the population, passing the corresponding binary vector.

[4] This function has been slightly modified in order to take into account the fact that some of the auxiliary variables may not be aggregated, i.e. their values must always be set to '1'. For this reason, it is not possible to directly load package "genalg", but it is necessary to use this modified version

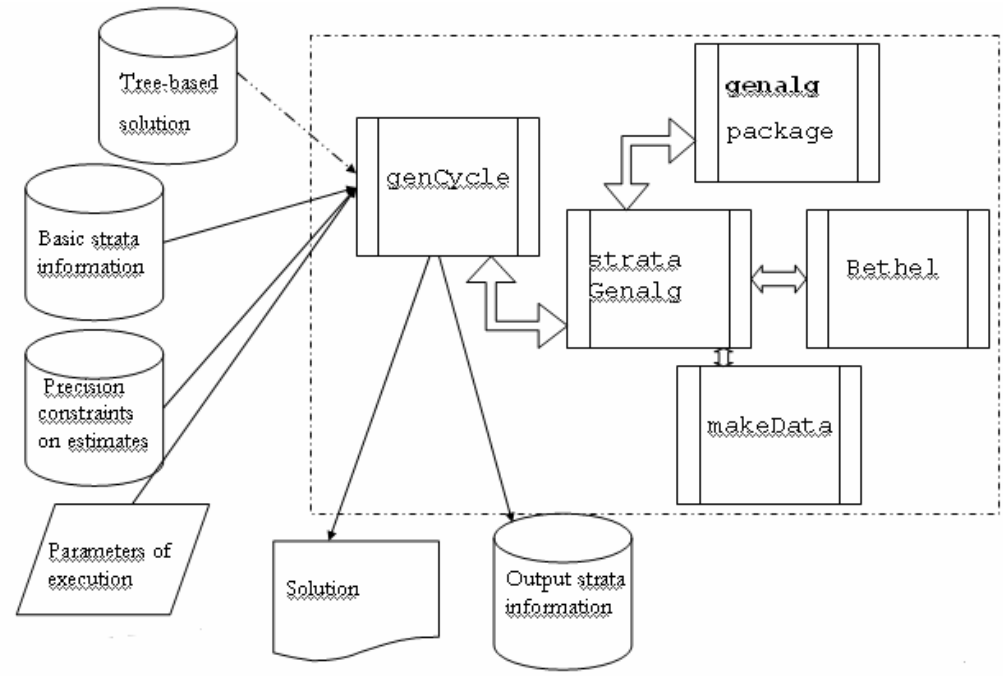
First operation of the evaluation function is to call the "makeData" function, passing the current binary vector.

This binary vector is used to process Xs variable in order to aggregate basic strata. At the same time, all information required by Bethel-Chromy algorithm (means and standard deviations of Ys) is computed and associated to aggregate strata. In this way, information necessary to solve optimal multivariate allocation problem is ready, and passed to function "Bethel", that returns a vector of integers, corresponding to units allocated to different strata of the current solution. The total sample size is then computed, and associated to the current solution as a measure of its fitness.

This process is repeated for each individual in the current population, until a measure of fitness is available for each individual. Then, "rbga.bin" proceeds to select individuals on the basis of their fitness, applying on them crossover and mutation genetic operators, thus generating individuals for the next generation.

This operations can be monitored, and iteration results are graphically displayed.

Once reached the maximum number of iterations, the program outputs the best solution, with its associates number of strata, total sample size, and information on values of Xs utilised to aggregate strata. Finally, a dataset containing aggregate strata corresponding to the best solution are written to an ASCII file, containing also, for each stratum, corresponding allocated units.



6. An application: the Italian Farm Structure Survey

The sampling frame used for the selection of FSS sample contains 2,153,710 farms, each one characterised by the following variables:

1. region (21 different values);
2. provinces (103 different values);
3. legal status (2 values);
4. sector of economical activity (9 values);
5. dimension in terms of production (3 values);
6. dimension in terms of agricultural surface (3 values);
7. dimension in terms of owned cattle (3 values)
8. altimetry class (5 values).

Fourteen different Y variables have been considered as the main target of FSS, on which required precision (in terms of maximum coefficient of variations) has been fixed at regional levels (domains of

interest).

First, we describe the current procedure that was followed in 2003 to choose the most suitable stratification for sample selection.

In the first step a take-all stratum was defined in each region on the base of local characteristics.

In the second step a choice between a stratification based on province or on whole region was made region by region on the basis of local organizational considerations.

In the third step the others six variables were alternatively used in each stratum, among those chosen in the second step, and for each stratification the optimal sample size was computed (the minimum sample size in each stratum had been fixed to 50). The stratification supporting the overall minimum sample size in each region (usually defined on different variables) was considered as the output of this step.

In the fourth step the remaining five variables were used separately to refine the stratification previously obtained. For each of these refined stratifications the optimal sample size was computed considering the same constraints used in step 3.

This stepwise procedure continued refining the best stratification obtained in each step by using still available variables, until the more refined stratifications region by region revealed to be less efficient than the stratification in the previous step.

By doing so, the total amount of planned sample size was fixed to 52.713 units.

Let us now describe the application of the algorithms introduced in previous paragraphs.

Variable "region" determines the 21 domains of interest, so we have to consider as X variables the remaining seven.

The most detailed available partition of the frame consists in 24,454 different strata, 1,787 of whom have been defined as take-all strata. So, the basic strata are given by the 22,667 sampling strata obtained subtracting the 1,787 take-all strata. The latter are collapsed in just one stratum, whose 6,971 units will always be selected for whatever sample.

Desired precision levels (expressed in terms of coefficients of variation) have been set for each one of the fourteen different target variables: they are 5% or 6% for most important variables in each region and 99% for the others. Table 1 contains a complete description of the coefficient of variations used in planning the 2003 FSS survey.

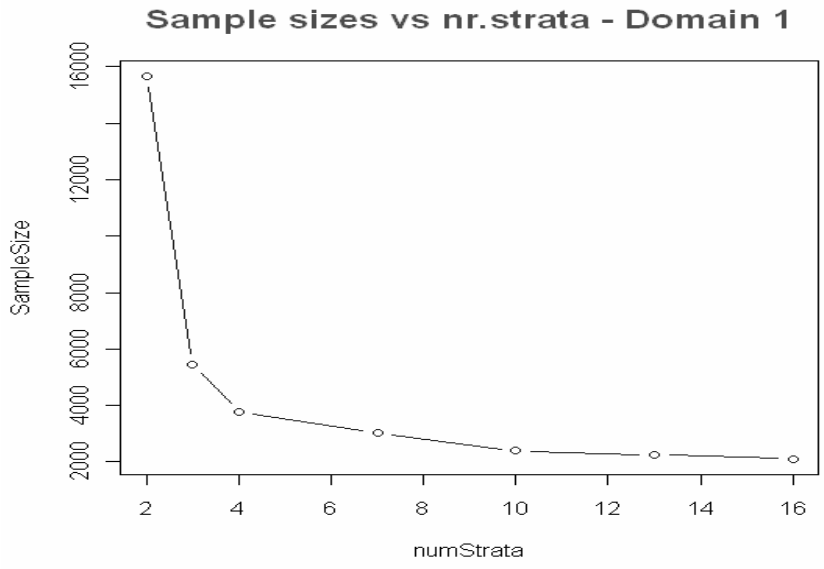
The tree-based algorithm worked iteratively region by region, with the following processing parameters (we report here the first region):

```
--------------------------
 *** Parameters ***
--------------------------
Domain:  1
Maximum number of strata:  25000
Minimum number of units per stratum:  50
Take-all strata (TRUE/FALSE):  TRUE
number of take-all strata :  1
number of sampling strata :  1646
Number of auxiliary variables:  7
...of which, not aggregable :  0 0 0 0 0 0 0
Stopping criterion (delta):  5
Number of target variables:  14
Number of domains:  1
```

Table 1. Coefficients of variation used in planning the 2003 FSS and as input for the algorithms

	cereals	industrial crops	Fresh vegetables	flowers	vineyards	olives	citrus fruit	fruits	bovines	pigs	Sheep	Economic size units	agricultural	Livestock unit
Piemonte	5.0				5.0				5.0			5.0	6.0	6.0
Valle d'A.									5.0			5.0	6.0	6.0
Lombardia	5.0								5.0	5.0		5.0	6.0	6.0
Bolzano								5.0				5.0	6.0	6.0
Trento								5.0				5.0	6.0	6.0
Veneto	5.0				5.0					5.0		5.0	6.0	6.0
Friuli V.G.	5.0											5.0	6.0	6.0
Liguria				5.0								5.0	6.0	6.0
Emilia R.	5.0				5.0			5.0	5.0	5.0		5.0	6.0	6.0
Toscana	5.0				5.0							5.0	6.0	6.0
Umbria						5.0						5.0	6.0	6.0
Marche												5.0	6.0	6.0
Lazio	5.0		5.0		5.0	5.0						5.0	6.0	6.0
Abruzzo						5.0						5.0	6.0	6.0
Molise						5.0						5.0	6.0	6.0
Campania	5.0		5.0			5.0		5.0				5.0	6.0	6.0
Puglia	5.0		5.0		5.0	5.0						5.0	6.0	6.0
Basilicata	5.0											5.0	6.0	6.0
Calabria	5.0					5.0	5.0					5.0	6.0	6.0
Sicilia	5.0		5.0		5.0	5.0	5.0				5.0	5.0	6.0	6.0
Sardegna	5.0										5.0	5.0	6.0	6.0

In the following graph the different solutions evaluated during the tree search for the first region are reported: the relation between the sample sizes and number of strata is clear, and the trend is almost the same for all regions.

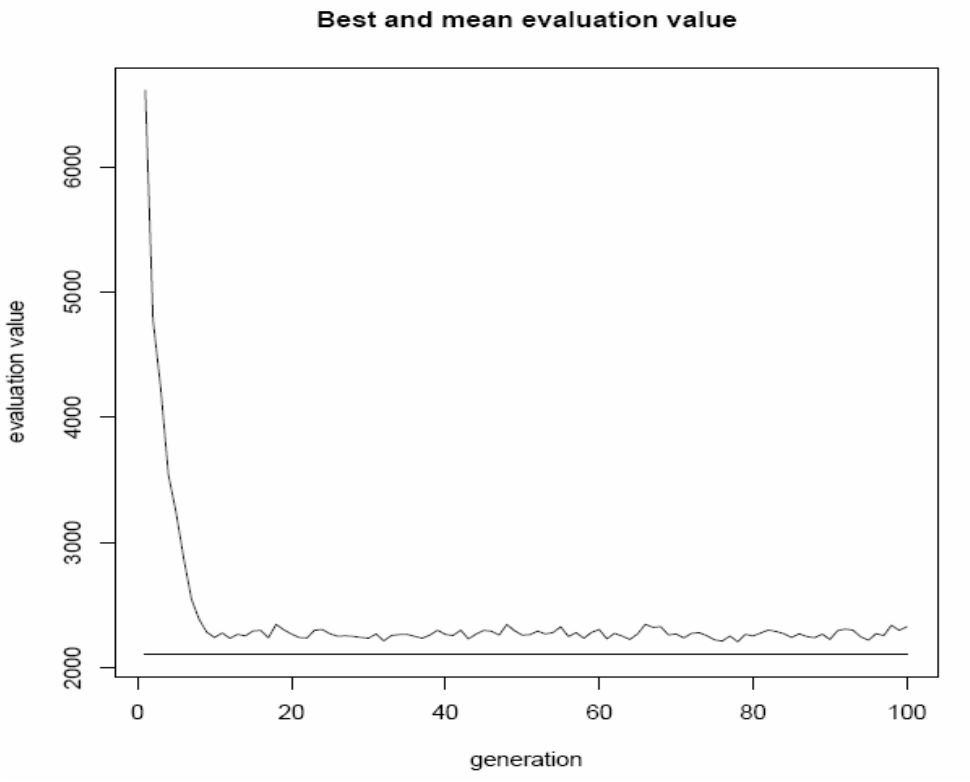


The second step consisted in applying the genetic algorithm in order to understand if the solution found by the tree-base could be improved. To speed GA convergence, the tree-based solution for each particular region was given as a "suggestion" for the GA to generate the first population.

These are the processing parameters for GA (first region):

```
-------------------------------------------
 *** Parameters ***
---------------------------
Domain:  1
Maximum number of strata:  25000
Minimum number of units per stratum:  50
Take-all strata (TRUE/FALSE):  TRUE
number of take-all strata :  1
number of sampling strata :  1646
Number of auxiliary variables:  7
...of which, not aggregable :  0 0 0 0 0 0 0
Number of target variables:  14
Number of domains:  1
Number of GA iterations:  100
Dimension of GA population:  100
Mutation chance in GA generation:  0.05
Initial ratio between 0's and 1's:  3
```

In the next graph, related to the first region, convergence towards the solution is reported. The same behaviour can be observed also for other regions: a quick decrease of sampling size, then a steady move towards the best solution.



In the following table the results of the two algorithmic solutions are compared with the one obtained in 2003 "by hand" and actually used to carry out the FSS survey.

We can observe a dramatic decrease of planned sample size, moving from the actual one (adopted for 2003 FSS) to the tree-based proposed solution: a saving of near 43% on the total, differentiated region by region, with a maximum decrease for Puglia (-64%) and a minimum for Trento.

The genetic algorithm slightly improves the overall tree-based solution (saving 2,7%), but there are significant decreases in some regions, as Liguria and Campania (respectively: -15% and -17%). The final result is an overall size of 28.925, which is only 54% of the adopted sample size in 2003 (52.713).

Table 2. Comparison among the three solutions

	(1) Actual sample size used in 2003 FSS	(2) Tree-based solution	% diff. (2)/(1)	(3) Genetic Algorithm solution	% diff. (3)/(2)
Italia	**52.713**	**29.726**	**-43,61**	**28.925**	**-2,69**
Piemonte	3.560	1.546	-56,57	1.546	0,00
Valle d'Aosta	409	384	-6,11	376	-2,08
Lombardia	5.125	2.237	-56,35	2.237	0,00
Bolzano	687	540	-19,94	540	0,00
Trento	667	638	-4,35	638	0,00
Veneto	3.873	2.299	-40,64	2.138	-7,00
Friuli	1.262	619	-50,95	619	0,00
Liguria	1.327	777	-41,45	657	-15,44
Emilia R.	3.117	1.966	-36,93	1.933	-1,68
Toscana	2.833	1.341	-52,67	1.310	-2,31
Umbria	1.363	858	-37,05	858	0,00
Marche	1.188	508	-57,24	498	-1,97
Lazio	3.710	2.620	-29,38	2.620	0,00
Abruzzo	1.222	950	-22,26	876	-7,79
Molise	1.183	867	-26,71	719	-17,07
Campania	3.163	2.154	-31,90	2.040	-5,29
Puglia	6.595	2.326	-64,73	2.272	-2,32
Basilicata	965	684	-29,12	684	0,00
Calabria	2.846	2.080	-26,91	2.042	-1,83
Sicilia	5.011	3.182	-36,50	3.182	0,00
Sardegna	2.607	1.140	-36,50	1.140	0,00

7. Conclusions and future work

Optimal stratification of a sampling frame can be determined together with optimal sampling size and allocation of units in strata. Information required is much the same than the input required by Bethel algorithm: required precision on target estimates (Ys), and values of mean and standard deviation of corresponding variables in frame strata. Frame strata should be considered at the most detailed level, i.e. the one determined by the Cartesian product of values of all available auxiliary variables.

Two different approaches are possible: the ones based on the tree and the genetic algorithms. The former performs well in terms of computational efficiency, but is more likely to be subject to local

minima, while the latter may obtain better solutions, but can be very expensive in terms of processing time.

Our proposal is to combine the two of them, first by using the tree-based algorithm in order to obtain a (relatively) fast solution, and then to try to improve that solution (given as a "suggestion" to speed up convergence) by using the genetic algorithm.

The application to a real survey showed that this approach performed well: the overall solution revealed to be much better than the one produced by the expert methodologist (who, incidentally, needed one month work to fix it).

Some limitations are still affecting the methods and related software, and need additional work to be overcome.

From a conceptual point of view, a first limit is in the nature of auxiliary variables, that must be strictly categorical. In case of continuous variables, they need to be transformed in ordinal ones, and something should be said about the best way to do it.

A second limit has consequences on the efficiency: when activating values in the same variables, equivalent configurations are generated, and only the first one should be evaluated.

References

Ballin M., Falorsi P.D., Vicard P. (2000). "La strategia di campionamento dell'indagine Long Form del censimento intermedio dell'industria e dei servizi". Atti del convegno SIS: "Verso i censimenti del 2000", Udine 2-4 Giugno 1999, vol.2, 224-234

Benedetti R., Espa G., Lafratta G. (2005), "A Tree-based Approach to Forming Strata in Multipurpose Business Surveys", *Discussion Paper No 5, 2005*, Università degli Studi di Trento – Dipartimento di Economia

Bethel J. (1985), "An Optimum Allocation Algorithm for Multivariate Surveys", in *American Statistical Proceedings of the Survey Research Methods Section*, pp. 209-212

Bethel J. (1989), "Sample Allocation in Multivariate Surveys", *Survey Methodology*, 15 pp. 47-57

Chromy J. (1987), "Design Optimisation with Multiple Objectives", in *American Statistical Proceedings of the Survey Research Methods Section*, pp. 194-199

Cochran, W.G. (1977), *Sampling technique,* John Wiley & Sons, New York

Di Giuseppe R., Giaquinto P., Pagliuca D. (2004). "MAUSS: un software generalizzato per risolvere il problema dell'allocazione campionaria nelle indagini ISTAT". *Collana Contributi Istat n. 7 / 2004*

Falorsi P.D., Ballin M., De Vitiis C., Scepi G. (1998). "Principi e metodi del software generalizzato per la definizione del disegno di campionamento nelle indagini sulle imprese condotte dall'Istat". *Statistica Applicata*, 10, 2, 235-257.

Kozak, M., Verma, M.R., Zielinski, A. (2007). Modern Approach to Optimum Stratification: Review and Perspectives. *Statistics in Transition* 8(2), 223-250.

Lucasius C.B. and Kateman G. (1993). Understanding and using genetic algorithms - Part 1. Concepts, properties and context. *Chemometrics and Intelligent Laboratory Systems 9:1-33*.

Lucasius C.B. and Kateman G. (1994). Understanding and using genetic algorithms - Part 2. Representation, configuration and hybridization. *Chemometrics and Intelligent Laboratory Systems 25:99-145*.

R Development Core Team (2007). "*R: A language and environment for statistical computing*." R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Särndal, C.E., Swensson, B., Wretman, J. (1992), *Model Assisted Survey Sampling. Springer-Verlag.*

Schmitt L. M. (2001), "Theory of Genetic Algorithms*"*, *Theoretical Computer Science* (259), pp. 1-61

Schmitt L. M. (2004), "Theory of Genetic Algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling*"*, *Theoretical Computer Science* (310), pp. 181-231

Vose M. D. (1999), *The Simple Genetic Algorithm: Foundations and Theory*, MIT Press, Cambridge, MA

Willighagen E. (2005), "*genalg: R Based Genetic Algorithm*". R package version 0.1.1. URL http:// cran.r-project.org/

Appendix 1 – Structure of input data and parameters

Both functions genCycle and treeCycle, require the following input data frames:

1. **strati**: this data frame contains information on strata at the most detailed level (one row for each sampling stratum);
2. **cens**: this data frame is optional (active if strcens=TRUE) and has the same structure of the previous one, but is referred to census strata, i.e. strata whose units must all be taken;
3. **errori**: in this data frame the required precision levels for different variables in different domains are indicated (one row for each domain type).

In scripts "genCycle.R" and "treeCycle.R" these data frames are created by reading external files which are "tab delimited", with the first row containing the names of the fields. ***Important: the field name convention must be strictly followed (uppercase and lowercase included).***

strati and **cens** (k1 auxiliary variables, k2 target variables, and k3 domain types)		
Name in first row	**Meaning**	**Possible values**
strato	value of stratum identifier (unique)	alphanumeric string
N	population in stratum	integer
X1	value in stratum of the first auxiliary variable	integer or character
…	values of auxiliary variables	integer or character
Xk1	value in stratum of the last auxiliary variable	integer or character
M1	mean value in stratum of first target variable	real
…	means of target variables	real
Mk2	value in stratum of the last target variable	real
S1	standard deviation in stratum of the first target variable	real
…	standard deviations of target variables	real
Sk2	standard deviation in stratum of the last target variable	real
DOM1	value of first domain type	alphanumeric string
…	values of domain types	alphanumeric string
DOMk3	value of last domain type	alphanumeric string
cens	flag indicating if the stratum is a take-all stratum (1=yes, 0=no)	0,1
cost	cost associated to each interview in the stratum	real

errori (k2 target variables, k3 domain types)		
Name in the first row	**Meaning**	**Possible values**
DOM	indication of domain type	DOM1, … DOMk3
CV1	desired coefficient of variation for the first target variable in current domain type	real (0.00-1.00)
…	desired coefficients of variation	real (0.00-1.00)
CVk2	desired coefficient of variation for the last target variable in current domain type	real (0.00-1.00)

Input parameters can be subdivided in a common set for the two scripts, and in two specific sets. In the following table they are all reported.

Parameter	**Meaning**	**Possible values**	**Default**
Common to genCycle and treeCycle			
maxstrati	maximum number of strata in final solution	integer	Number of rows of strata file
minnumstr	minimum number of units to be allocated in each stratum	integer	2
strcens	flag indicating the presence of take-all strata	TRUE/FALSE	FALSE
nVarX	number of auxiliary variables	integer	
fixedVarX	vector of binary flags indicating which auxiliary variable's values must not be collapsed	vector of 1s and 0s	c(rep(0,nVarX))
Specific treeCycle (tree-base algorithm)			
delta	stopping rule for tree branching: indicates the gain in terms of sampling size reduction from one level to the next	integer	
Specific to genCycle (genetic algorithm)			
iter	number of generations	integer	50
pops	dimension of population (# of chromosomes)	integer	100
mut_chance	mutation chance	real (0.00 -1.00)	0.03
zero_one	ratio between 0s and 1s in the initial population	integer	3
sugg	flag indicating the presence of an initial solution	TRUE/FALSE	FALSE
suggestion	vector solution from a previous run	vector of 1s and 0s	

Appendix 2 – Listings of principal functions

```
#################################################################
# Script: Bethel.R                                              #
# Function BETHEL definition                                    #
# Multivariate optimal allocation for different domains         #
# of interest in stratified sample design                       #
# Extension of Bethel methodology with Chromy Algorithm         #
# see Bethel(1989)"Sample Allocation in Multivariate Surveys"   #
#                  - Survey Methodology                         #
# Version 2.2: 30.5.2008                                        #
# Author: Daniela Pagliuca                                      #
#################################################################
bethel <- function (
          errors,
          stratif,
          minnumstrat=2,
          epsilon=10^(-11),
          maxiter=200,
          printa=FALSE
          )
# Begin body
          {
#-------------------------------------------------------------
# Initialization of parameters. Attribution of initial values
#-------------------------------------------------------------
  iter1<-0
  maxiter1<-25
  val=NULL
  m <- NULL
  s <- NULL
  cv=NULL
#------------------------------
# Initialization of parameters
#------------------------------

  nstrat=nrow(stratif)
  nvar=ncol(errors)-1
  ndom=nrow(errors)
  varloop <- c(1:nvar)
  strloop <- c(1:nstrat)
  domloop <- c(1:ndom)
#-----------------------------------------------------------
# Initial Data Structures - selection from input variables
#-----------------------------------------------------------

# means
  med   <-as.matrix(stratif[,names(stratif) %in%
                    sapply(1:nvar, function(i) paste("M",i,sep=""))])
# variances estimates
  esse  <-as.matrix(stratif[,names(stratif) %in%
                    sapply(1:nvar, function(i) paste("S",i,sep=""))])
# domains
  nom_dom <-sapply(1:ndom, function(i) paste("DOM",i,sep=""))
  dom     <-as.vector(stratif[,names(stratif) %in% nom_dom])
# populations
  N <- as.vector(stratif$N)
# vectors cost
  cost  <- as.vector(stratif$cost)
# vectors cens
  cens  <- as.vector(stratif$cens)
# strata with N < minnumstrata are set to take-all strata
  cens[N<minnumstrat] <- 1
  nocens=1-cens

# numbers of different domains of interest (numbers of modalities/categories
# for each type of domain)
  if (ndom ==1) (nvalues<-nlevels((as.factor(stratif$DOM1))))
  if (ndom>1) {nvalues<-sapply(nom_dom, function(vari)
                  {val<-c(val, nlevels(as.factor(dom[,vari])))})}

#-------------------------------------------------
#  Building means (m) and deviations matrices (s)
#  for different domains of interest
#-------------------------------------------------

#-------------------------------------------------
# disjunctive matrix
#-------------------------------------------------
```

```
crea_disj=function(data,vars){
  out=NULL
  sapply(vars, function(vari){ out<<-
  cbind(out,diag(nlevels(as.factor(data[,vari])))[as.factor(data[,vari]),] ) })
  out }

disj<-crea_disj(stratif,nom_dom)

nc  <- ncol(disj)
#  (m) and (s)
for(i in 1:nc)
   { m <- cbind(m, disj[,i]*med )
     s <- cbind(s, disj[,i]*esse ) }

#-------------------------------------------------------------
# computation of the coefficients of variation CVs
# for different domains of interest
#-------------------------------------------------------------
for (k in domloop)
 {cvx <- as.matrix( errors[k,names(errors) %in%
          sapply(1:nvar, function(i)   paste("CV",i,sep=""))])
     ndomvalues <- c(1:nvalues[k])
     for (k1 in ndomvalues) { cv <- cbind(cv,cvx)  }
    }

#-------------------------------------------------------------
# New definition of initial values
#-------------------------------------------------------------

  nvar_orig<-nvar

  nvar <- ncol(cv)          # new numbers of variables
  varloop <- c(1:nvar)

  NTOT <- c(rep(0,nvar))
  CVfin <- c(rep(0,nvar))
  varfin <- c(rep(0,nvar))
  totm <- c(rep(0,nvar))

#-------------------------------------------------------------
# Calculation of aij - matrix of standardized precision units
#-------------------------------------------------------------
crea_a=function(){
  numA <- (N**2)*(s**2)*nocens

  denA1<-colSums(t(t(N*m)*c(cv)))^2
  denA2<-colSums(N*(s**2)*nocens)

  denA<-denA1+denA2+epsilon

  a<-t(t(numA)/denA)
return(a)
}

#-------------------------------------------------------------
# Computation of alfa's values - Chromy Algorithm Iteration
#-------------------------------------------------------------

chromy=function(alfatot,diffe,iter, alfa, alfanext, x)
{
  while ( diffe > epsilon && iter<maxiter )
  {
  iter <- iter + 1

  den1 =sqrt(rowSums(t( t(a)*c(alfa)) ))
  den2=sum(sqrt(rowSums(t(t(a*cost)*c(alfa)))))

  x<-sqrt(cost)/(den1*den2+ epsilon)

  alfatot <- sum( c(alfa)*(t(a)%*%x)**2 )
  alfanext <- c(alfa)*(t(a)%*%x)**2/alfatot

  diffe <- max(abs(alfanext-alfa))
  alfa <- alfanext
  alfa2 <<- alfanext
  }

# Allocation vector
n <- ceiling(1 / x)
```

```
return(n)
}

a<-crea_a()
n<-chromy(0,999,0,c(rep(1/nvar,nvar)), c(rep(0,nvar)), array(0.1,dim=c(nstrat,1)))

# check n>N
contx<-sum(n>N)
cens[n>N]<-1
nocens=1-cens

#check n<minnumstr
for (i in strloop)
  { if (n[i] < minnumstrat) { n[i] <- min(minnumstrat, N[i])} }


####ITERATIONS###
while (contx>0 && iter1<maxiter1)
{
  iter1=iter1+1

  a<-crea_a()
  n<-chromy(0,999,0,c(rep(1/nvar,nvar)), c(rep(0,nvar)), array(0.1,dim=c(nstrat,1)))

  # check n>N
  contx<-sum(n>N)
  cens[n>N]<-1
  nocens=1-cens

for (i in strloop)
  { if (n[i] < minnumstrat) { n[i] <- min(minnumstrat, N[i])} }
}

#-------------------------
#Definitive best allocation
#-------------------------
n=(nocens*n)+(cens*N)
#-------------------------
#-----------------------------------------------------------
if (printa == TRUE) {
#-----------------------------------------------------------
# Populations in strata for different domains of interest NTOTj
#-----------------------------------------------------------
  for (j in varloop)  { NTOT[j] <- sum((m[,j]>0)*N)  }
#-----------------------------------------------------------
# Computation of the CVs
#-----------------------------------------------------------
varfin=rowSums(t((s*N)**2*(1-round(n)/N)/round(n) ) / NTOT**2)
totm  =rowSums(t(m*N))

CVfin <- round(sqrt(varfin/(totm/NTOT)**2),digits=4)
#-----------------------------------------------------------
# Sensitivity (10%)
g <- 0
for (i in strloop) {
  t <- 0
  for (j in varloop) {
    t <- t + alfa2[j] * a[i,j]
  }
  g <- g + sqrt(cost[i]*t)
}
g <- g**2
sens <- 2 * 0.1 * alfa2 * g
#---------------------
# Printing allocation
#---------------------

for (i in strloop) {
print (paste("Stratum: ",i, "  Population: ",N[i], "  Sample Units: ",round(n[i])))
}
print(paste("Total sample size : ",round(sum(n))))

#---------------------
# Printing CV's
#---------------------
domcard <- c(rep(0,ndom))
for (k in (1:ndom))  {
  statement <- paste(" domcard[",k,"] <- length(levels(as.factor(stratif$DOM",k,")))",sep="")
  eval(parse(text=statement))
  }
```

```
j <- 0
for (k in (1:ndom)) {
  valloop <- c(1:(domcard[k]))
  for (k1 in valloop) {
    for (k2 in 1:(ncol(errors)-1)) {
      j <- j + 1
      if (j == 1) print ("Domain/Var.  Planned CV   Actual CV   Sensitivity 10%")
       print(paste(k,k1,"/",k2,"       ",cv[j],"       ",CVfin[j],"      ",ceiling(sens[j])))
}}}
}
return(n)
# End body
}
#End script
```

```
#####################################################################
# Script: makeData.R                                                #
# Function "makeData" for preparing data input to Bethel function   #
# Version: 30.6.2008                                                #
# Author: Giulio Barcaroli                                          #
#####################################################################

makeData <- function (strati,
                      nvarX,
                      nvar,
                      vett,
                      censiti,
                      fixedVarX,
                      dominio)
{
varXloop <- c(1:nvarX)
varloop <- c(1:nvar)
nummod <- 0
for (j in varXloop) {
  statement <- paste("strati$X",j," <- as.factor(strati$X",j,")",sep='')
  eval(parse(text=statement))
  statement <- paste("numX",j," <- length(levels(strati$X",j,"))",sep='')
  eval(parse(text=statement))
  statement <- paste("levels(strati$X",j,") <- c(1:numX",j,")",sep='')
  eval(parse(text=statement))
  statement <- paste("nummod <- nummod + numX",j,sep='')
  eval(parse(text=statement))
  statement <- paste("strati$X",j," <- as.integer(strati$X",j,")",sep='')
  eval(parse(text=statement))
  }
modloop <- c(1:nummod)
j2 <- 0
for (k1 in varXloop) {
  statement <- paste("X",k1,"m <- strati$X",k1,sep='')
  eval(parse(text=statement))
  statement <- paste("mods <- c(1:numX",k1,")",sep='')
  eval(parse(text=statement))
  for (j1 in mods){
    j2 <- j2 + 1
    statement <- paste("if (vett[j2] == 0) X",k1,"m[X",k1,"m == j1] <- 0",sep='')
    eval(parse(text=statement))
  }
}
for (k1 in varXloop) {
  statement <- paste("X",k1,"m <- as.factor(X",k1,"m)",sep='')
  eval(parse(text=statement))
}
N <- strati$N
varXloop <- c(1:nvarX)
string1 <- ""
string4 <- ""
string7 <- ""
string12 <- ""
for (k in varXloop) {
  string1 <- paste(string1,"X",k,"m,",sep='')
  if (k < nvarX) string4 <- paste(string4,"strwrk$X",k,"m,",sep='')
  if (k == nvarX) string4 <- paste(string4,"strwrk$X",k,"m)",sep='')
  string7 <- paste(string7,"'X",k,"m',",sep='')
  string12 <- paste(string12,"X",k,",",sep='')
  }
string2 <- ""
string3 <- ""
string5 <- ""
string6 <- ""
string8 <- ""
string9 <- ""
string10 <- ""
string11 <- ""
varloop <- c(1:nvar)
for (k1 in varloop) {
statement <- paste("TM",k1," <- strati$M",k1," * strati$N",sep='')
eval(parse(text=statement))
statement <- paste("TVAR",k1," <- strati$S",k1,"**2 * (strati$N - 1)",sep='')
eval(parse(text=statement))
string2 <- paste(string2,"TM",k1,",",sep='')
string3 <- paste(string3,"TVAR",k1,",",sep='')
string5 <- paste(string5,"'TM",k1,"',",sep='')
string6 <- paste(string6,"'TM",k1,"t',",sep='')
string8 <- paste(string8,"'diff",k1,"',",sep='')
```

```
string9 <- paste(string9,"'TVAR",k1,"',",sep='')
string10 <- paste(string10,"M",k1,",",sep='')
string11 <- paste(string11,"S",k1,",",sep='')
}
statement <- paste("strwrk <- data.frame(",string1,string2,string3,"N)",sep='')
eval(parse(text=statement))
statement <- paste("strwrk2 <- aggregate(strwrk[,c(",string5,"'N')],list(",string4,",sum)",sep='')
eval(parse(text=statement))
statement <- paste ("colnames(strwrk2) <- c(",string7,string6,"'Nt')",sep='')
eval(parse(text=statement))
strwrk <- merge(strwrk,strwrk2)
rm(strwrk2)
for (k1 in varloop) {
statement <- paste("strwrk$diff",k1," <- strwrk$N * ((1/strwrk$N)*strwrk$TM",k1," -
(1/strwrk$Nt)*strwrk$TM",k1,"t)**2",sep='')
eval(parse(text=statement))
}
statement <- paste("strwrkagg <-
aggregate(strwrk[,c(",string5,string9,string8,"'N')],list(",string4,",sum)",sep='')
eval(parse(text=statement))
for (k1 in varloop) {
  statement <- paste("M",k1," <- round((strwrkagg$TM",k1," / strwrkagg$N),digits=4)",sep='')
  eval(parse(text=statement))
  statement <- paste("S",k1," <- round(sqrt((1/strwrkagg$N)*(strwrkagg$TVAR",k1," +
strwrkagg$diff",k1,")),digits=4)",sep='')
  eval(parse(text=statement))
}
N <- strwrkagg$N
dimens <- nrow(strwrkagg)
DOM1 <- c(rep(dominio,dimens))
cost <- c(rep(1,dimens))
cens <- c(rep(censiti,dimens))
for (k1 in varXloop) {
  statement <- paste("X",k1," <- as.vector(strwrkagg$Group.",k1,")",sep='')
  eval(parse(text=statement))
}
stringa <- "strato <- paste("
for (k1 in varXloop) {
  if (k1 < nvarX) stringa <- paste(stringa,"X",k1,",'-',",sep='')
  if (k1 == nvarX) stringa <- paste(stringa,"X",k1,sep='')
}
statement <- paste(stringa,",sep='')",sep='')
eval(parse(text=statement))
statement <- paste("strcor <-
data.frame(strato,",string12,string10,string11,"N,DOM1,cost,cens)",sep='')
eval(parse(text=statement))
j <- 1
for (i in 1:nvarX) {
  if (fixedVarX[i] == 1) {
    j <- j+1
    statement <- paste("strcor$DOM",j,"<- as.factor(strcor$X",i,")",sep="")
    eval(parse(text=statement))
  }
}
return(strcor)
#End function
}
#End script
```

```
#############################################################################
# Script: strataGenalg.R                                                    #
# Function: strataGenalg                                                    #
# Version: April 4th, 2008                                                  #
# Author: Giulio Barcaroli                                                  #
# Optimise sampling units multivariate allocation with Genetic Algorithm #
# together with strata determination                                        #
# Indication of X's not to be aggregated (planned domains)                  #
# Suggestions: given by Tree-based solution                                 #
#############################################################################
strataGenalg <- function (errori,
                          strati,
                          cens,
                          strcens = TRUE,
                          nvarX,
                          dominio,
                          fixedVarX,
                          maxstrati,
                          minnumstr,
                          pi,
                          iter,
                          pops,
                          mut_chance,
                          zero_one,
                          highvalue,
                          sugg )
{
if (sugg == TRUE) {
  filesugg <- paste("TreeBasedSolution",dominio,".txt",sep="")
  dv <- read.table(file=filesugg,header=FALSE)
  suggest <- as.vector(dv[1:nrow(dv),])
  }
#-----------------------------------------------------------------------------
# Loading genetic algorithm package and functions (rbga.bin2 is modified)
source("BethelF.R")
source("MakeData.R")
source("rbga.bin2.R")
source("summary.rbga.R")
source("plot.rbga.R")
fileres <- paste("results",dominio,".txt",sep="")
sink(file=fileres)
#-----------------------------------------------------------------------------
nvar=ncol(errori)-1
ndom=nrow(errori)
nstrcamp <- nrow(strati)
if (strcens == TRUE) nstrcens <- nrow(cens)
#-----------------------------------------------------------------------------
cat("\n-----------------------------------------------")
cat("\nOptimal stratification with Genetic Algorithm")
cat("\n-----------------------------------------------")
cat("\n *** Parameters ***")
cat("\n---------------------------")
cat("\nDomain: ",dominio)
cat("\nMaximum number of strata: ",maxstrati)
cat("\nMinimum number of units per stratum: ",minnumstr)
cat("\nTake-all strata (TRUE/FALSE): ",strcens)
if (strcens == TRUE) cat("\nnumber of take-all strata : ",nstrcens)
cat("\nnumber of sampling strata : ",nstrcamp)
cat("\nNumber of auxiliary variables: ",nvarX)
cat("\n...of which, not aggregable : ",fixedVarX)
cat("\nNumber of target variables: ",nvar)
cat("\nNumber of domains: ",ndom)
cat("\nNumber of GA iterations: ",iter)
cat("\nDimension of GA population: ",pops)
cat("\nMutation chance in GA generation: ",mut_chance)
cat("\nInitial ratio between 0's and 1's: ",zero_one)
if (sugg == TRUE) cat("\nSuggestions: ",suggest)
#-----------------------------------------------------------------------------
# Dimension of solutions vector
varXloop <- c(1:nvarX)
varloop <- c(1:nvar)
nummod <- 0
cat("\n---------------------------")
cat("\nTranformation of variables X's")
cat("\n---------------------------")
fixedValuesX <- NULL
k <-0
for (j in varXloop) {
  statement <- paste("strati$X",j," <- as.factor(strati$X",j,")",sep='')
```

```
  eval(parse(text=statement))
  statement <- paste("cat('\nVariable X",j," original values: ',levels(strati$X",j,"))",sep="")
  eval(parse(text=statement))
  statement <- paste("numX <- length(levels(strati$X",j,"))",sep='')
  eval(parse(text=statement))
  statement <- paste("levels(strati$X",j,") <- c(1:numX)",sep='')
  eval(parse(text=statement))
  statement <- paste("cat('\nVariable X",j," transformed values: ',levels(strati$X",j,"))",sep="")
  eval(parse(text=statement))
  statement <- paste("nummod <- nummod + numX",sep='')
  eval(parse(text=statement))
  if (fixedVarX[j] == 0) fixedValuesX <- as.vector(c(fixedValuesX,c(rep(0,numX))))
  if (fixedVarX[j] == 1) fixedValuesX <- as.vector(c(fixedValuesX,c(rep(1,numX))))
  statement <- paste("strati$X",j," <- as.integer(strati$X",j,")",sep='')
  eval(parse(text=statement))
  }
cat("\n---------------------------")
#-------------------------------------------------------------------------
# Preparation of take-all strata
if (strcens == TRUE) {
  vett <- fixedValuesX
  censiti <- 1
  cens <- makeData (cens, nvarX, nvar, vett, censiti, fixedVarX, dominio)
  dimcens <- nrow(cens)
}
#-------------------------------------------------------------------------
# Preparation of suggestions
#numrighe <- round(pops/2)
numrighe <- 1
suggestions1 <- matrix(c(rep(0,numrighe*nummod)),c(numrighe,nummod))
if (sugg == TRUE) {
  for (i in 1:numrighe) {
    for (j in 1:nummod) suggestions1[i,j] <- suggest[j]
  }
}
######################
# Evaluation function
######################
evaluate <- function(indices) {
# Preparazione dati
  soluz <- NULL
  v <- NULL
  dimens <- NULL
  censiti <- 0
  strcor <- makeData (strati, nvarX, nvar, indices, censiti, fixedVarX, dominio)
  dimsamp <- nrow(strcor)
  if (strcens == TRUE) strcor <- rbind(strcor,cens)
  dimens <- nrow(strcor)
#  cat("\nCurrent solution:",indices)
#  cat("\nNumber of input strata:",dimens)
#  cat("\n    ...sampling strata:",dimsamp)
#  cat("\n    ...take-all strata:",dimcens)
#-------------------------------------------------------------------------
# Chiamata funzione allocazione multivariata
  if (dimens < maxstrati)
  soluz <- bethel (
               errori,
               strcor,
               minnumstr,
               printa=FALSE
               )
  sink()
  sink(file=fileres,append=TRUE)
  ntot <- round(sum(soluz))
  if (dimens > (maxstrati-1)) ntot <- highvalue
  cat("\n",ntot)
  return(ntot)
#print(indices)
#print(paste("Dimensione: ",round(ntot),"   Numero strati: ",dimens))
}
#########################
# Monitoring of processing
#########################
monitor <- function(obj) {
    # plot the population
    minEval = min(obj$evaluations);
    plot(obj,type="hist");
#plot(dimens,round(rbga.results$best[iter]),type="b",main = "",col="blue")
#title(main = list("Best sample sizes vs number of strata", cex=1.5,
#                  col="red", font=2))
```

```
}
##############################
# Genetic algorithm execution
##############################
if (sugg == TRUE)
rbga.results <- rbga.bin2(
    size=nummod,
    suggestions=suggestions1,
                monitorFunc=monitor,
                iters=iter,
                popSize=pops,
                zeroToOneRatio=zero_one,
                mutationChance=mut_chance,
                evalFunc=evaluate,
                verbose=TRUE,
                showSetting=TRUE,
    fixed=fixedValuesX)
if (sugg == FALSE)
rbga.results <- rbga.bin2(
    size=nummod,
                monitorFunc=monitor,
                iters=iter,
                popSize=pops,
                zeroToOneRatio=zero_one,
                mutationChance=mut_chance,
                evalFunc=evaluate,
                verbose=TRUE,
                showSetting=TRUE,
    fixed=fixedValuesX)
######################
# Results
######################
plot(rbga.results)
summary(rbga.results,echo=TRUE)
#print(paste("Sample size: ",round(rbga.results$best[iter])))
#cat(" *** Sample size: ",round(rbga.results$best[iter]))
#-----------------------------------------------------------------------
# Writing strata corresponding to optimal solution
#-----------------------------------------------------------------------
v <- rbga.results$population[rbga.results$evaluations==min(rbga.results$evaluations),]
if (class(v) == "matrix") v <- as.vector(v[1,])
censiti <- 0
strcor <- makeData (strati, nvarX, nvar, v, censiti, fixedVarX, dominio)
if (strcens == TRUE) strcor <- rbind(strcor,cens)
dimens <- nrow(strcor)
soluz <- bethel (
                errori,
                strcor,
                minnumstr,
                printa=F
                )
sink()
sink(file=fileres,append=TRUE)
cat("\n *** Sample size: ",sum(soluz))
cat(paste("\n *** Number of strata: ",dimens))
risulta <- cbind(strcor,soluz)
fileout <- paste("outstrata",dominio,".txt",sep="")
write.table(risulta,file=fileout,sep="\t",row.names=FALSE,col.names=TRUE,quote=FALSE)
cat("\n...written output to",fileout)
#-----------------------------------------------------------------------
# Decoding of zero values in X's
#-----------------------------------------------------------------------
statement <- NULL
cat("\nDecoding zero values in X's")
cat("\n---------------------------")
for (j in varXloop) {
  statement <- paste("x <- levels(risulta$X",j,")",sep="")
  eval(parse(text=statement))
  x <- x[x!="0"]
  statement <- paste("if (length(x) > 0 ) {cat('\n'); cat('Variable X",j," = 0 <=> X",j," <>', x)
}",sep="")
  eval(parse(text=statement))
  }
cat("\n---------------------------")
cat("\n")
proc.time()
sink()
# End function
}
#End script
```

```
############################################################
# Script: genCycle.R                                       #
# Function: genCycle                                       #
# Main program for                                         #
# GA-based optimal stratification and optimal allocation   #
# Version: 4.9.2008                                        #
# Author: Giulio Barcaroli                                 #
############################################################
# Reading data
errori <- read.table("cv.txt",header=TRUE,sep="\t")
strati <- read.table("strata.txt",header=TRUE,sep="\t")
cens <- read.table("takeallstrata.txt",header=TRUE,sep="\t")
# Function definition
genCycle <- function (
    errori,
    strati,
    cens,
    strcens = TRUE,
    maxstrati = 25000,
    minnumstr = 50,
    nvarX = 7,
    fixedVarX,
    pi = 0.1,
    iter = 25,
    pops = 100,
    mut_chance = 0.05,
    zero_one = 3,
    highvalue = 100000,
    sugg = FALSE )
{
sink("results.txt")
#-------------------------------------------------------------------------
# Parameters
#-------------------------------------------------------------------------
source("strataGenalg.R")
source("BethelF.R")
#-------------------------------------------------------------------------
if (missing(fixedVarX)) fixedVarX <- rep(0,nvarX)
erro <- split(errori,list(errori$region))
stcamp <- split(strati,list(strati$DOM1))
if (strcens == TRUE) stcens <- split(cens,list(cens$DOM1))
ndom <- length(levels(as.factor(strati$DOM1)))
fixedVarX <- rep(0,nvarX)
for (i in 1:ndom) {
  erro[[i]] <- erro[[i]][,-16]
  strataGenalg (errori=erro[[i]],
                strati=stcamp[[i]],
                cens=stcens[[i]],
                strcens,
                nvarX,
                dominio=i,
                fixedVarX,
                maxstrati,
                minnumstr,
                pi,
                iter,
                pops,
                mut_chance,
                zero_one,
                highvalue,
                sugg )
  }
outstrata <- read.delim("outstrata1.txt")
if (ndom > 1) {
  for (i in 2:ndom) {
    statement <- paste('out<-read.delim("outstrata',i,'.txt")',sep="")
    eval(parse(text=statement))
    outstrata <- rbind(outstrata,out)
    }
  }
dimens <- sum(outstrata$soluz)
write.table(outstrata,file="outstrata.txt",sep="\t",row.names=FALSE,  col.names=TRUE,quote=FALSE)
cat("\n *** Sample size : ",dimens)
cat("\n *** Number of strata : ",nrow(outstrata))
cat("\n---------------------------")
cat("\n...written output to outstrata.txt")
sink()
#End function
}
```

```
genCycle(errori,strati,cens)
#End script
```

```
#############################################################
# Script: strataTree.R                                      #
# Function: strataTree                                      #
# Version: 2.7.2008                                         #
# Author: Giulio Barcaroli                                  #
# This function implements the tree-based approach for the  #
# determination of the best stratification of a sampling    #
# frame under accuracy constraints of sampling estimates    #
#############################################################

strataTree <- function (errori,
              strati,
              cens,
              strcens,
              nvarX,
              dominio,
              fixedVarX,
              maxstrati,
              minnumstr,
              epsilon,
              delta)
{
fileres <- paste("results",dominio,".txt",sep="")
sink(file=fileres)
#-----------------------------------------------------------------------------
#options(echo=FALSE)
#-----------------------------------------------------------------------------
# Functions
source("BethelF.R")
source("MakeData.R")
#-----------------------------------------------------------------------------
nvar <- ncol(errori)-1
ndom <- nrow(errori)
nstrcamp <- nrow(strati)
if (strcens == TRUE) nstrcens <- nrow(cens)
#-----------------------------------------------------------------------------
cat("\n---------------------------")
cat("\n *** Parameters ***")
cat("\n---------------------------")
cat("\nDomain: ",dominio)
cat("\nMaximum number of strata: ",maxstrati)
cat("\nMinimum number of units per stratum: ",minnumstr)
cat("\nTake-all strata (TRUE/FALSE): ",strcens)
if (strcens == TRUE) cat("\nnumber of take-all strata : ",nstrcens)
cat("\nnumber of sampling strata : ",nstrcamp)
cat("\nNumber of auxiliary variables: ",nvarX)
cat("\n...of which, not aggregable : ",fixedVarX)
cat("\nStopping criterion (delta): ",delta)
cat("\nNumber of target variables: ",nvar)
cat("\nNumber of domains: ",ndom)
#-----------------------------------------------------------------------------
# First call to Bethel
#-----------------------------------------------------------------------------
# Preparation of data for Bethel
#-----------------------------------------------------------------------------
SampleSize <- NULL
numStrata <- NULL
dimens <- NULL
varXloop <- c(1:nvarX)
varloop <- c(1:nvar)
nummod <- 0
cat("\n---------------------------")
cat("\nTranformation of stratification variables X's")
cat("\n---------------------------")
fixedValuesX <- NULL
for (j in varXloop) {
  statement <- paste("strati$X",j," <- as.factor(strati$X",j,")",sep='')
  eval(parse(text=statement))
  statement <- paste("cat('\nVariable X",j," original values: ',levels(strati$X",j,"))",sep="")
  eval(parse(text=statement))
  statement <- paste("numX <- length(levels(strati$X",j,"))",sep='')
  eval(parse(text=statement))
  statement <- paste("levels(strati$X",j,") <- c(1:numX)",sep='')
  eval(parse(text=statement))
  statement <- paste("cat('\nVariable X",j," transformed values: ',levels(strati$X",j,"))",sep="")
  eval(parse(text=statement))
  statement <- paste("nummod <<- nummod + numX",sep='')
  eval(parse(text=statement))
  if (fixedVarX[j] == 0) fixedValuesX <- as.vector(c(fixedValuesX,c(rep(0,numX))))
```

```
  if (fixedVarX[j] == 1) fixedValuesX <- as.vector(c(fixedValuesX,c(rep(1,numX))))
  statement <- paste("strati$X",j," <- as.integer(strati$X",j,")",sep='')
  eval(parse(text=statement))
  }
cat("\n---------------------------")
sink()
#vett <- c(rep(0,nummod))
vett <- fixedValuesX
censiti <- 0
strcor <- makeData (strati, nvarX, nvar, vett, censiti, fixedVarX, dominio)
#vett <- c(rep(1,nummod))
if (strcens == TRUE) {
  censiti <- 1
  cens <- makeData (cens, nvarX, nvar, vett, censiti, fixedVarX, dominio)
  strcor <- rbind(strcor,cens)
}
dimensione <- nrow(strcor)
#-----------------------------------------------------------------------
# Bethel
#-----------------------------------------------------------------------
soluz <- NULL
soluz <- bethel(
              errori,
              strcor,
              minnumstr,
              epsilon=10^(-11),
                         printa=F
              )
sink(file=fileres,append=TRUE)
cat("\nLevel : 0")
cat("\nSample size : ",round(sum(soluz)))
oldn <- sum(soluz)
TreeLevel <- c(0)
SampleSize <- cbind(SampleSize,oldn)
numStrata <- cbind(numStrata,dimensione)
plot(numStrata,SampleSize,type="b",main = "",col="blue")
tit <- paste("Sample sizes vs nr.strata - Domain ",dominio,sep="")
title(main = list(tit, cex=1.5, col="red", font=2))
#-----------------------------------------------------------------------
# Treatment of auxiliary variables
nummod <- 0
for (j in varXloop) {
  statement <- paste("strati$X",j," <- as.factor(strati$X",j,")",sep='')
  eval(parse(text=statement))
  statement <- paste("numX",j," <- length(levels(strati$X",j,"))",sep='')
  eval(parse(text=statement))
  statement <- paste("levels(strati$X",j,") <- c(1:numX",j,")",sep='')
  eval(parse(text=statement))
  statement <- paste("nummod <- nummod + numX",j,sep='')
  eval(parse(text=statement))
  statement <- paste("strati$X",j," <- as.integer(strati$X",j,")",sep='')
  eval(parse(text=statement))
  }
#oldv <- c(rep(0,nummod))
oldv <- fixedValuesX
n <- c(rep(999999,nummod))
diff <- 999999
dimensione <- 0
k <- 0
# Inizio loop livello albero (k)
#while (k < 1) {
while (diff > delta && dimensione < maxstrati) {
  k <- k + 1
  v <- oldv
  modloop <- c(1:(nummod))
#print(v)
# Inizio loop sugli strati atomici
  for (j in modloop) {
    if (v[j] != 1)  {
        v[j] <- 1
        j2 <- 0
        for (k1 in varXloop) {
            statement <- paste("X",k1,"m <- strati$X",k1,sep='')
            eval(parse(text=statement))
            statement <- paste("mods <- c(1:numX",k1,")",sep='')
            eval(parse(text=statement))
                 for (j1 in mods){
                        j2 <- j2 + 1
                        statement <- paste("if (v[j2] == 0) X",k1,"m[X",k1,"m == j1] <- 0",sep='')
                        eval(parse(text=statement))
```

```
                        }
                    }
#---------------------------------------------------------------------------
# Preparation of data to bethel
#---------------------------------------------------------------------------
        censiti <- 0
        strcor <- makeData (strati, nvarX, nvar, v, censiti, fixedVarX, dominio)
        if (strcens == TRUE) strcor <- rbind(strcor,cens)
        dimens[j] <- nrow(strcor)
#---------------------------------------------------------------------------
# Bethel
#---------------------------------------------------------------------------
        soluz <- bethel(
               errori,
               strcor,
               minnumstr,
               epsilon=10^(-11),
               maxiter=200,
                          printa=F
               )
       n[j] <- round(sum(soluz))
#cat("\n",n[j])
# Fine loop if v[j] != 0
    }
    v <- oldv
# Fine loop modalità (j)
  }
  j3 <- 0
  while (j3 < nummod) {
    j3 <- j3 + 1
    if (n[j3] == min(n))  {
      oldv[j3] <- 1
      dimensione <- dimens[j3]
      numStrata <- cbind(numStrata,dimensione)
      j3 <- nummod + 1
      }
  }
  diff <- oldn - min(n)
  oldn  <- min(n)
sink()
sink(file=fileres,append=TRUE)
cat("\n---------------------------")
cat ("\nLevel : ",k)
cat ("\nBest current solution : ",oldv)
cat ("\nSample size : ",oldn)
TreeLevel <- cbind(TreeLevel,k)
SampleSize <- cbind(SampleSize,oldn)
plot(numStrata,SampleSize,type="b",main = "",col="blue")
tit <- paste("Sample sizes vs nr.strata - Domain ",dominio,sep="")
title(main = list(tit, cex=1.5, col="red", font=2))
cat("\n---------------------------")
# Fine loop livello (k)
}
#---------------------------------------------------------------------------
# Writing output
#---------------------------------------------------------------------------
strcor <- makeData (strati, nvarX, nvar, oldv, censiti, fixedVarX, dominio)
if (strcens == TRUE) strcor <- rbind(strcor,cens)
soluz <- bethel(
               errori,
               strcor,
               minnumstr,
               epsilon=10^(-11),
               maxiter=200,
                          printa=T
               )
risulta <- cbind(strcor,soluz)
soluzione <- paste("TreeBasedSolution",dominio,".txt",sep="")
write.table(oldv,file=soluzione,sep="\t",row.names=FALSE,col.names=FALSE,quote=FALSE)
fileout<-paste("outstrati",dominio,".txt",sep="")
write.table(risulta,file=fileout,sep="\t",row.names=FALSE,col.names=TRUE,quote=FALSE)
cat("\n *** Domain : ",dominio)
cat("\n *** Sample size : ",sum(soluz))
cat("\n *** Number of strata : ",nrow(strcor))
cat("\n---------------------------")
cat("\n...written best solution to TreeBasedSolution.txt")
cat("\n...written output to outstrati.txt")
#---------------------------------------------------------------------------
# Decoding 0's in X's
#---------------------------------------------------------------------------
```

```
statement <- NULL
cat("\nDecoding zero values in X's")
cat("\n---------------------------")
for (j in varXloop) {
  statement <- paste("x <- levels(risulta$X",j,")",sep="")
  eval(parse(text=statement))
  x <- x[x!="0"]
  statement <- paste("if (length(x) > 0 ) {cat('\n'); cat('Variable X",j," = 0 <=> X",j," <>', x)
}",sep="")
  eval(parse(text=statement))
  }
cat("\n---------------------------")
cat("\n")
proc.time()
sink()
#End function
}
```

```
#############################################################
# Script: treeCycle.R                                       #
# Function: treeCycle                                       #
# Main program for                                          #
# tree-based optimal stratification and optimal allocation  #
# Version: 4.9.2008                                         #
# Author: Giulio Barcaroli                                  #
#############################################################
# Reading data
errori <- read.table("cv.txt",header=TRUE,sep="\t")
strati <- read.table("strata.txt",header=TRUE,sep="\t")
cens <- read.table("takeallstrata.txt",header=TRUE,sep="\t")
# Functiion definition
treeCycle <- function (
    errori = errori,
    strati = strati,
    cens = cens,
    strcens = TRUE,
    maxstrati = 25000,
    minnumstr = 50,
    nvarX = 7,
    fixedVarX,
    epsilon=0.00000000001,
    delta=5 )
{
#-----------------------------------------------------------------------------
source("strataTree.R")
source("BethelF.R")
#-----------------------------------------------------------------------------
if (missing(fixedVarX)) fixedVarX <- rep(0,nvarX)
erro <- split(errori,list(errori$region))
stcamp <- split(strati,list(strati$DOM1))
if (strcens == TRUE) stcens <- split(cens,list(cens$DOM1))
ndom <- length(levels(as.factor(strati$DOM1)))
ndomvals <- length
for (i in 1:ndom) {
  erro[[i]] <- erro[[i]][,-16]
  strataTree (errori=erro[[i]],
              strati=stcamp[[i]],
              cens=stcens[[i]],
              strcens,
              nvarX,
              dominio=i,
              fixedVarX,
              maxstrati,
              minnumstr,
              epsilon,
              delta)
  }
outstrata <- read.delim("outstrata1.txt")
if (ndom > 1) {
for (i in 2:ndom) {
  statement <- paste('out<-read.delim("outstrata',i,'.txt")',sep="")
  eval(parse(text=statement))
  outstrata <- rbind(outstrata,out)
    }
  }
dimens <- sum(outstrata$soluz)
write.table(outstrata,file="outstrata.txt",sep="\t",row.names=FALSE,col.names=TRUE,quote=FALSE)
sink("results.txt")
cat("\n *** Sample size : ",dimens)
cat("\n *** Number of strata : ",nrow(outstrata))
cat("\n--------------------------")
cat("\n...written output to outstrata.txt")
sink()
#End function
}
treeCycle(errori,strati,cens)
#End script
```

Contributi ISTAT(*)

1/2004 – Marcello D'Orazio, Marco Di Zio e Mauro Scanu – *Statistical Matching and the Likelihood Principle: Uncertainty and Logical Constraints*

2/2004 – Giovanna Brancato – *Metodologie e stime dell'errore di risposta. Una sperimentazione di reintervista telefonica*

3/2004 – Franco Mostacci, Giuseppina Natale e Elisabetta Pugliese – *Gli indici dei prezzi al consumo per sub popolazioni*

4/2004 – Leonello Tronti – *Una proposta di metodo: osservazioni e raccomandazioni sulla definizione e la classificazione di alcune variabili attinenti al mercato del lavoro*

5/2004 – Ugo Guarnera – *Alcuni metodi di imputazione delle mancate risposte parziali per dati quantitativi: il softaware* Quis

6/2004 – Patrizia Giaquinto, Marco Landriscina e Daniela Pagliuca – *La nuova funzione di analisi dei modelli implementata in Genesees v. 3.0*

7/2004 – Roberto Di Giuseppe, Patrizia Giaquinto e Daniela Pagliuca – *MAUSS (Multivariate Allocation of Units in Sampling Surveys): un software generalizzato per risolvere il problema dell'allocazione campionaria nelle indagini Istat*

8/2004 – Ennio Fortunato e Liana Verzicco – *Problemi di rilevazione e integrazione della condizione professionale nelle indagini sociali dell'Istat*

9/2004 – Claudio Pauselli e Claudia Rinaldelli – *La valutazione dell'errore di campionamento delle stime di povertà relativa secondo la tecnica Replicazioni Bilanciate Ripetute*

10/2004 – Eugenio Arcidiacono, Marina Briolini, Paolo Giuberti, Marco Ricci, Giovanni Sacchini e Giorgia Telloli – *Procedimenti giudiziari, reati, indagati e vittime in Emilia-Romagna nel 2002: un'analisi territoriale sulla base dei procedimenti iscritti nel sistema informativo Re.Ge.*

11/2004 – Enrico Grande e Orietta Luzi – *Regression trees in the context of imputation of item non-response: an experimental application on business data*

12/2004 – Luisa Frova e Marilena Pappagallo – *Procedura di now-cast dei dati di mortalità per causa*

13/2004 – Giorgio DellaRocca, Marco Di Zio, Orietta Luzi, Emanuela Scavalli e Giorgia Simeoni – *IDEA (Indices for Data Editing Assessment): sistema per la valutazione degli effetti di procedure di controllo e correzione dei dati e per il calcolo degli indicatori SIDI*

14/2004 – Monica Pace, Silvia Bruzzone, Luisa Frova e Marilena Pappagallo – *Review of the existing information about death certification practices, certificate structures and training tools for certification of causes of death in Europe*

15/2004 – Elisa Berntsen – *Modello Unico di Dichiarazione ambientale: una fonte amministrativa per l'Archivio delle Unità Locali di Asia*

16/2004 – Salvatore F. Allegra e Alessandro La Rocca – *Sintetizzare misure elementari: una sperimentazione di alcuni criteri per la definizione di un indice composto*

17/2004 – Francesca R. Pogelli – *Un'applicazione del modello "Country Product Dummy" per un'analisi territoriale dei prezzi*

18/2004 – Antonia Manzari – *Valutazione comparativa di alcuni metodi di imputazione singola delle mancate risposte parziali per dati quantitativi*

19/2004 – Claudio Pauselli – *Intensità di povertà relativa: stima dell'errore di campionamento e sua valutazione temporale*

20/2004 – Maria Dimitri, Ersilia Di Pietro, Alessandra Nuccitelli e Evelina Paluzzi – *Sperimentazione di una metodologia per il controllo della qualità di dati anagrafici*

21/2004 – Tiziana Pichiorri, Anna M. Sgamba e Valerio Papale – *Un modello di ottimizzazione per l'imputazione delle mancate risposte statistiche nell'indagine sui trasporti marittimi dell'Istat*

22/2004 – Diego Bellisai, Piero D. Falorsi, Annalisa Lucarelli, Maria A. Pennucci e Leonello G. Tronti – *Indagine pilota sulle retribuzioni di fatto nel pubblico impiego*

23/2004 – Lidia Brondi – *La riorganizzazione del sistema idrico: quadro normativo, delimitazione degli ambiti territoriali ottimali e analisi statistica delle loro caratteristiche strutturali*

24/2004 – Roberto Gismondi e Laura De Sandro – *Provisional Estimation of the Italian Monthly Retail Trade Index*

25/2004 – Annamaria Urbano, Claudia Brunini e Alessandra Chessa – *I minori in stato di abbandono: analisi del fenomeno e studio di una nuova prospettiva d'indagine*

26/2004 – Paola Anzini e Anna Ciammola – *La destagionalizzazione degli indici della produzione industriale: un confronto tra approccio diretto e indiretto*

27/2004 – Alessandro La Rocca – *Analisi della struttura settoriale dell'occupazione regionale: 8° Censimento dell'industria e dei servizi 2001 7° Censimento dell'industria e dei servizi 1991*

28/2004 – Vincenzo Spinelli e Massimiliano Tancioni – *I Trattamenti Monetari non Pensionistici: approccio computazionale e risultati della sperimentazione sugli archivi INPS-DM10*

29/2004 – Paolo Consolini – *L'indagine sperimentale sull'archivio fiscale modd.770 anno 1999: analisi della qualità del dato e stime campionarie*

1/2005 – Fabrizio M. Arosio – *La stampa periodica e l'informazione on-line: risultati dell'indagine pilota sui quotidiani on-line*

2/2005 – Marco Di Zio, Ugo Guarnera e Orietta Luzi – *Improving the effectiveness of a probabilistic editing strategy for business data*

3/2005 – Diego Moretti e Claudia Rinaldelli – *EU-SILC complex indicators: the implementation of variance estimation*

4/2005 – Fabio Bacchini, Roberto Iannaccone e Edoardo Otranto – *L'imputazione delle mancate risposte in presenza di dati longitudinali: un'applicazione ai permessi di costruzione*

5/2005 – Marco Broccoli – *Analisi della criminalità a livello comunale: metodologie innovative*

6/2005 – Claudia De Vitiis, Loredana Di Consiglio e Stefano Falorsi – *Studio del disegno campionario per la nuova rilevazione continua sulle Forze di Lavoro*

7/2005 – Edoardo Otranto e Roberto Iannaccone – *Continuous Time Models to Extract a Signal in Presence of Irregular Surveys*

(*) ultimi cinque anni

8/2005 – Cosima Mero e Adriano Pareto – *Analisi e sintesi degli indicatori di qualità dell'attività di rilevazione nelle indagini campionarie sulle famiglie*

9/2005 – Filippo Oropallo – *Enterprise microsimulation models and data challenges*

10/2005 – Marcello D' Orazio, Marco Di Zio e Mauro Scanu – *A comparison among different estimators of regression parameters on statistically matched files through an extensive simulation study*

11/2005 – Stefania Macchia, Manuela Murgia, Loredana Mazza, Giorgia Simeoni, Francesca Di Patrizio, Valentino Parisi, Roberto Petrillo e Paola Ungaro – *Una soluzione per la rilevazione e codifica della Professione nelle indagini CATI*

12/2005 – Piero D. Falorsi, Monica Scannapieco, Antonia Boggia e Antonio Pavone – *Principi Guida per il Miglioramento della Qualità dei Dati Toponomastici nella Pubblica Amministrazione*

13/2005 – Ciro Baldi, Francesca Ceccato, Silvia Pacini e Donatella Tuzi – *La stima anticipata OROS sull'occupazione. Errori, problemi della metodologia attuale e proposte di miglioramento*

14/2005 – Stefano De Francisci, Giuseppe Sindoni e Leonardo Tininini – *Da Wînci/MD: un sistema per data warehouse statistici sul Web*

15/2005 – Gerardo Gallo e Evelina Palazzi – *I cittadini italiani naturalizzati: l'analisi dei dati censuari del 2001, con un confronto tra immigrati di prima e seconda generazione*

16/2005 – Saverio Gazzelloni, Mario Albisinni, Lorenzo Bagatta, Claudio Ceccarelli, Luciana Quattrociocchi, Rita Ranaldi e Antonio Toma – *La nuova rilevazione sulle forze di lavoro: contenuti, metodologie, organizzazione*

17/2005 – Maria Carla Congia – *Il lavoro degli extracomunitari nelle imprese italiane e la regolarizzazione del 2002. Prime evidenze empiriche dai dati INPS*

18/2005 – Giovanni Bottazzi, Patrizia Cella, Giuseppe Garofalo, Paolo Misso, Mariano Porcu e Marianna Tosi – *Indagine pilota sulla nuova imprenditorialità nella Regione Sardegna. Relazione Conclusiva*

19/2005 – Fabrizio Martire e Donatella Zindato – *Le famiglie straniere: analisi dei dati censuari del 2001 sui cittadini stranieri residenti*

20/2005 – Ennio Fortunato – *Il Sistema di Indicatori Territoriali: percorso di progetto, prospettive di sviluppo e integrazione con i processi di produzione statistica*

21/2005 – Antonella Baldassarini e Danilo Birardi – *I conti economici trimestrali: un approccio alla stima dell'input di lavoro*

22/2005 – Francesco Rizzo, Dario Camol e Laura Vignola – *Uso di XML e WEB Services per l'integrazione di sistemi informativi statistici attraverso lo standard SDMX*

1/2006 – Ennio Fortunato – *L'analisi integrata delle esigenze informative dell'utenza Istat: Il contributo del Sistema di Indicatori Territoriali*

2/2006 – Francesco Altarocca – *I design pattern nella progettazione di software per il supporto alla statistica ufficiale*

3/2006 – Roberta Palmieri – *Le migranti straniere: una lettura di genere dei dati dell'osservatorio interistituzionale sull'immigrazione in provincia di Macerata*

4/2006 – Raffaella Amato, Silvia Bruzzone, Valentina Delmonte e Lidia Fagiolo – *Le statistiche sociali dell'ISTAT e il fenomeno degli incidenti stradali: un'esperienza di record linkage*

5/2006 – Alessandro La Rocca – *Fuzzy clustering: la logica, i metodi*

6/2006 – Raffaella Cascioli – *Integrazione dei dati micro dalla Rilevazione delle Forze di Lavoro e dagli archivi amministrativi INPS: risultati di una sperimentazione sui dati campione di 4 province*

7/2006 – Gianluca Brogi, Salvatore Cusimano, Giuseppina del Vicario, Giuseppe Garofalo e Orietta Patacchia – *La realizzazione di Asia Agricoltura tramite l'utilizzo di dati amministrativi: il contenuto delle fonti e i risultati del processo di integrazione*

8/2006 – Simonetta Cozzi – *La distribuzione commerciale in Italia: caratteristiche strutturali e tendenze evolutive*

9/2006 – Giovanni Seri – *A graphical framework to evaluate risk assessment and information loss at individual level*

10/2006 – Diego Bellisai, Annalisa Lucarelli, Maria Anna Pennucci e Fabio Rapiti – *Feasibility studies for the coverage of public institutions in sections N and O*

11/2006 – Diego Bellisai, Annalisa Lucarelli, Maria Anna Pennucci e Fabio Rapiti – *Quarterly labour cost index in public education*

12/2006 – Silvia Montagna, Patrizia Collesi, Florinda Damiani, Danila Fulgenzio, Maria Francesca Loporcaro e Giorgia Simeoni – *Nuove esperienze di rilevazione della Customer Satisfaction*

13/2006 – Lucia Coppola e Giovanni Seri – *Confidentiality aspects of household panel surveys: the case study of Italian sample from EU-SILC*

14/2006 – Lidia Brondi – *L'utilizzazione delle surveys per la stima del valore monetario del danno ambientale: il metodo della valutazione contingente*

15/2006 – Carlo Boselli – *Le piccole imprese leggere esportatrici e non esportatrici: differenze di struttura e di comportamento*

16/2006 – Carlo De Greogorio – *Il nuovo impianto della rilevazione centralizzata del prezzo dei medicinali con obbligo di prescrizione*

1/2007 – Paolo Roberti, Maria Grazia Calza, Filippo Oropallo e Stefania Rossetti – *Knowledge Databases to Support Policy Impact Analysis: the EuroKy-PIA Project*

2/2007 – Ciro Baldi, Diego Bellisai, Stefania Fivizzani, e Marina Sorrentino – *Production of job vacancy statistics: coverage*

3/2007 – Carlo Lucarelli e Giampiero Ricci – *Working times and working schedules: the framework emerging from the new Italian lfs in a gender perspective*

4/2007 – Monica Scannapieco, Diego Zardetto e Giulio Barcaroli – *La Calibrazione dei Dati con R: una Sperimentazione sull'Indagine Forze di Lavoro ed un Confronto con GENESEES/SAS*

5/2007 – Giulio Barcaroli e Tiziana Pellicciotti – *Strumenti per la documentazione e diffusione dei microdati d'indagine: il Microdata Management Toolkit*

6/2007 – AA.VV. – *Seminario sulla qualità: l'esperienza dei referenti del sistema informativo SIDI - 1ª giornata*

7/2007 – Raffaella Cianchetta, Carlo De Gregorio, Giovanni Seri e Giulio Barcaroli – *Rilevazione sulle Pubblicazioni Scientifiche Istat*

8/2007 – Emilia Arcaleni, e Barbara Baldazzi – *Vivere non insieme: approcci conoscitivi al Living Apart Together*

9/2007 – Corrado Peperoni e Francesca Tuzi – *Trattamenti monetari non pensionistici metodologia sperimentale per la stima degli assegni al nucleo familiare*

10/2007 – AA.VV – *Seminario sulla qualità: l'esperienza dei referenti del sistema informativo SIDI - 2ª giornata*

11/2007 – Leonello Tronti – *Il prototipo (numero 0) dell'Annuario di statistiche del Mercato del Lavoro (AML)*

12/2007 – Daniele Frongia, Raffaello Martinelli, Fernanda Panizon, Bruno Querini e Andrea Stanco – *Il nuovo Sistema informatico Altri Servizi. Progetto di reingegnerizzazione dei processi produttivi delle indagini trimestrali di fatturato degli altri servizi*

1/2008 – Carlo De Gregorio, Stefania Fatello, Rosanna Lo Conte, Stefano Mosca, Francesca Rossetti – *Sampling design and treatment of products in Istat centralised CPI surveys*

2/2008 – Mario Abissini, Elisa Marzilli e Federica Pintaldi – *Test cognitivo e utilizzo del questionario tradotto: sperimentazioni dell'indagine sulle forze di lavoro*

3/2008 – Franco Mostacci – *Gli aggiustamenti di qualità negli indici dei prezzi al consumo in Italia: metodi, casi di studio e indicatori impliciti*

4/2008 – Carlo Vaccari e Daniele Frongia – *Introduzione al Web 2.0 per la Statistica*

5/2008 – Antonio Cortese – *La conta degli stranieri: una bella sfida per il censimento demografico del 2011*

6/2008 – Carlo De Gregorio, Carmina Munzi e Paola Zavagnini – *Problemi di stima, effetti stagionali e politiche di prezzo in alcuni servizi di alloggio complementari: alcune evidenze dalle rilevazioni centralizzate dei prezzi al consumo*

7/2008 – AA.VV. – *Seminario: metodi per il controllo e la correzione dei dati nelle indagini sulle imprese: alcune esperienze nel settore delle statistiche strutturali*

8/2008 – Monica Montella – *La nuova matrice dei margini di trasporto*

9/2008 – Antonia Boggia, Marco Fortini, Matteo Mazziotta, Alessandro Pallara, Antonio Pavone, Federico Polidoro, Rosabel Ricci, Anna Maria Sgamba e Angela Seeber – *L'indagine conoscitiva della rete di rilevazione dei prezzi al consumo*

10/2008 –Marco Ballin e Giulio Barcaroli – *Optimal stratification of sampling frames in a multivariate and multidomain sample design*