CASC PROJECT

Computational Aspects of Statistical Confidentiality

25 July 2001

Strategy for the implementation of individual risk methodology into μ-ARGUS: independent units

Alessandra Capobianchi
Silvia Polettini
Maurizio Lucarelli

ISTAT, MPS/D
Via C. Balbo, 16
00184, Roma
Italy

Deliverable No: 1.2-D1

1. Introduction

In this report the individual risk estimation algorithm is presented, focusing on the main differences with respect to the current version of μ-Argus, and explaining what should be implemented. The next section approaches the algorithm at a general, descriptive level. In section 3 we describe the variables that are needed to implement the risk (key variables, special types variables, etc.). Section 4 explains how to evaluate the frequencies of combinations of key variables in the sample, f_k , and discusses the estimation of these frequencies in the population, $\hat{F}_k$. These two processes will be described also in the presence of missing values. In section 5 estimation of the individual risk is presented. Section 6 contains the flow charts of the algorithms used for risk estimation. In section 7 we show some graphs as examples of what could be useful. Section 8 describes how to tie the μ-Argus suppression strategy with our methodology in order to produce a *safe file*.

2. Algorithm overview

Our approach is based on the need to handle *sample data*: the data file therefore does not include the whole population, but a subset of it, and every unit in the file represents one or more units of the population through the *individual weights*. So, the individuation and treatment of unique (or rare) combinations is no longer adequate in order to make the input file '*safe*', but is necessary to deal with a method that considers the sampling aspect of the data set.

Our method estimates the level of disclosure risk for each unit, defined as the probability of identifying an individual. After the application of the risk calculation algorithm, each record *i* will have associated its own value of the disclosure risk ρ_i. At this point, the user will input a threshold α, that he considers the maximum tolerable risk. This choice should be based on a graph representing the distribution of the individual risk in the file.

Once α has been selected, the algorithm will apply the suppressions only to records *i* such that $\rho_i > \alpha$, following a suppression method similar to the one already implemented in Argus.

After the protection step, the whole risk calculation algorithm should run again, producing the current risk values and another risk graph.

At this stage, the user should judge the gain of safety (i.e. the reduction of risk) attained. He/she has now two choices: a) he is satisfied by the result, and the output file is recorded as *safe file*; b) he/she discards the results, choosing to rollback to the previous risk values, e.g. in order to select another level of α.

A schematic representation of this overview is given in Figure 1.

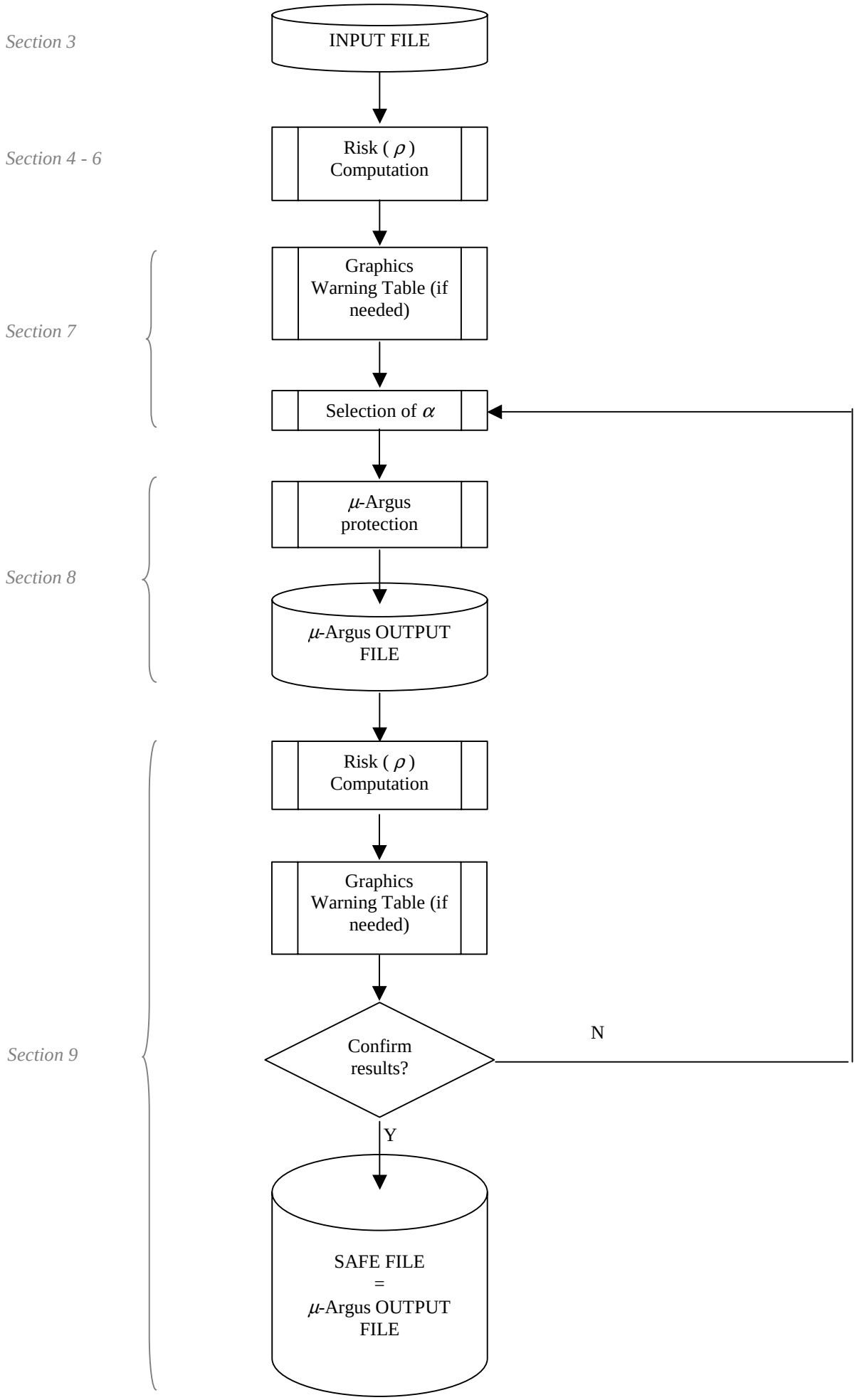


Figure 1: Process Structure

3. Input file

The file acquisition procedure is substantially the same as the one implemented in μ-Argus; it is however indispensable to get further information, additional to that already collected in the first μ-Argus window, in order to calculate the individual risk:

- Individual weights (w_i , always ≥ 1)
- Individual identifier (UnitID)

We wonder at which point of the program it could be better in your opinion to ask for the preferred protection method. This is because our approach does not use some information that, on the contrary, is required by μ-Argus (i.e. identification levels) and we believe it could be useful to drive the user in inserting just the information which is needed.

4. Frequencies calculation

A fundamental step for risk estimation is the computation of the frequencies f_k and $\hat{F}_k$.

First of all, we consider the population as partitioned into K sub-populations ($k = 1, \ldots, K$), defined through all the possible combinations of categories of the key variables.

It must be stressed that in the individuation of these sub-populations we use ***all*** the variables defined by the user as 'key'.

Suppose we have a file composed by 8 units:

HHID	UnitID	Key_Var1	Key_Var2	Key_Var3	Key_Var4	$f_i = f_{k(i)}$	w_i	$\hat{F}_k$
1	1	1	2	5	1	2	18	110
1	2	1	2	1	1	2	45,5	84,5
1	3	1	2	1	1	2	39	84,5
1	4	3	3	1	5	1	17	17
2	5	4	3	1	4	1	541	541
2	6	4	3	1	1	1	8	8
3	7	6	2	1	5	1	5	5
3	8	1	2	5	1	2	92	110

With $k(i) = k$ we denote the sub-population defined by the combination of categories of the key variables (*string*) in the unit i. In our example, there are 6 sub-populations, and unit 1 and 8 belong to the same sub-population identified by the string (1 , 2 , 5 ,1).

With f_k we represent the frequency (count) of units in the k^{th} sub-population that are present in the sample (i.e. in the file). The estimation of these frequencies in the population, $\hat{F}_k$, is given by the sum of the weights associated with the units belonging to that sub-population: $\hat{F}_k = \sum_{i:k(i)=k} w_i$.

In the example above, we get:

$k(1) = k(8) = (1, 2, 5, 1) \Rightarrow \hat{F}_{k(1)} = \hat{F}_{k(8)} = w_1 + w_8 = 18 + 92 = 110$

A problem may arise if there are missing values in the key variables.

Actually, a missing value could stand for any of the possible categories of the variable considered. Thus, in our opinion, computation of the f_k should take this into account. Consider the set of strings or combinations which are '*compatible'* with the one characterising the k^{th} sub-population, i.e. combinations which completely agree, except at most for one or more missing categories. In the presence of missing values, computation of f_k may be pursued by counting the number of units having strings compatible with the k^{th} sub-population. A similar argument can be applied to $\hat{F}_k$.

The table below shows how missing values affect computation of the relevant quantities in the context of the previous example:

HHID	UnitID	Key_Var1	Key_Var2	Key_Var3	Key_Var4	$f_i = f_{k(i)}$	w_i	$\hat{F}_k$
1	1	1	2	5	1	3	18	149
1	2	1	2	1	1	2	45,5	84,5
1	3	1	2	.	1	4	39	194,5
1	4	.	.	1	5	3	17	576
2	5	4	3	1	.	3	541	566
2	6	.	3	1	1	2	8	549
3	7	6	2	1	5	2	5	22
3	8	1	2	5	1	3	92	149

The string (1 , 2 , . , 1), associated whit the UnitID 3, is compatible with the sub-populations identified by the strings (1 , 2 , 5 , 1) and (1 , 2 , 1 , 1), and, in the same way, in each of this two sub-populations it has to be counted also the unit characterised by the string (1 , 2 , . , 1).

So:

$\hat{F}_{k(1)} = \hat{F}_{k(8)} = w_1 + w_8 + w_3 = 18 + 92 + 39 = 149,$

while

$\hat{F}_{k(3)} = w_3 + w_1 + w_8 + w_2 = 39 + 18 + 92 + 45{,}5 = 194{,}5$

5. Base Individual Risk computation

The individual risk, $r_i^{ind} = r_{k(i)}^{ind}$, represents the base individual risk for a unit i having combination $k(i)=k$ of key variables, and is the same for every unit belonging to the same sub-population. It is given by:

$$r_{k(i)}^{ind} = r_k^{ind} = \left(\frac{\hat{p}_k}{1-\hat{p}_k}\right)^{f_k} \left\{ A_0 \left(1 + \sum_{j=0}^{f_k-3} (-1)^{j+1} \prod_{l=0}^{j} B_l \right) + (-1)^{f_k} \log(\hat{p}_k) \right\} \tag{1}$$

where

$$\hat{p}_k = \frac{f_k}{\hat{F}_k} = \frac{f_k}{\sum_{i:k(i)=k} w_i}, \tag{2}$$

and w_i are the individual weights,

while $B_l = \frac{(f_k - 1 - l)^2}{(l+1)(f_k - 2 - l)} \frac{\hat{p}_k^{l+2-f_k} - 1}{\hat{p}_k^{l+1-f_k} - 1}$ and $A_0 = \frac{\hat{p}_k^{1-f_k} - 1}{(f_k - 1)}$. (3)

The above formulation works for $f_k \geq 3$; if $f_k = 1$ we use:

$$r_k = \frac{\hat{p}_k}{1-\hat{p}_k} \log\left(\frac{1}{\hat{p}_k}\right), \tag{3a}$$

while if $f_k = 2$:

$$r_k = \left(\frac{\hat{p}_k}{1-\hat{p}_k}\right) - \left[\left(\frac{\hat{p}_k}{1-\hat{p}_k}\right)^2 \log\left(\frac{1}{\hat{p}_k}\right)\right]. \tag{3b}$$

However, we found the task of evaluating formula (1) exceedingly heavy or even absolutely impossible when observed frequencies are too large. In these cases the introduction of a numerical approximation is convenient. We obtained satisfactory results using:

$$r_k = \frac{\hat{p}_k}{f_k - (1 - \hat{p}_k)} \tag{4}$$

In the flow chart presented in section 6 this approximation is used for frequencies greater than 40. We were forced to set this value because of software limitations: however, use of a higher threshold could increase precision. In the same flow chart are presented solutions for the two cases where the denominator is 0 in the two equations presented in formula (3) – i.e. $f_k = 1$ and $f_k = 2$.

5.1. Final risk

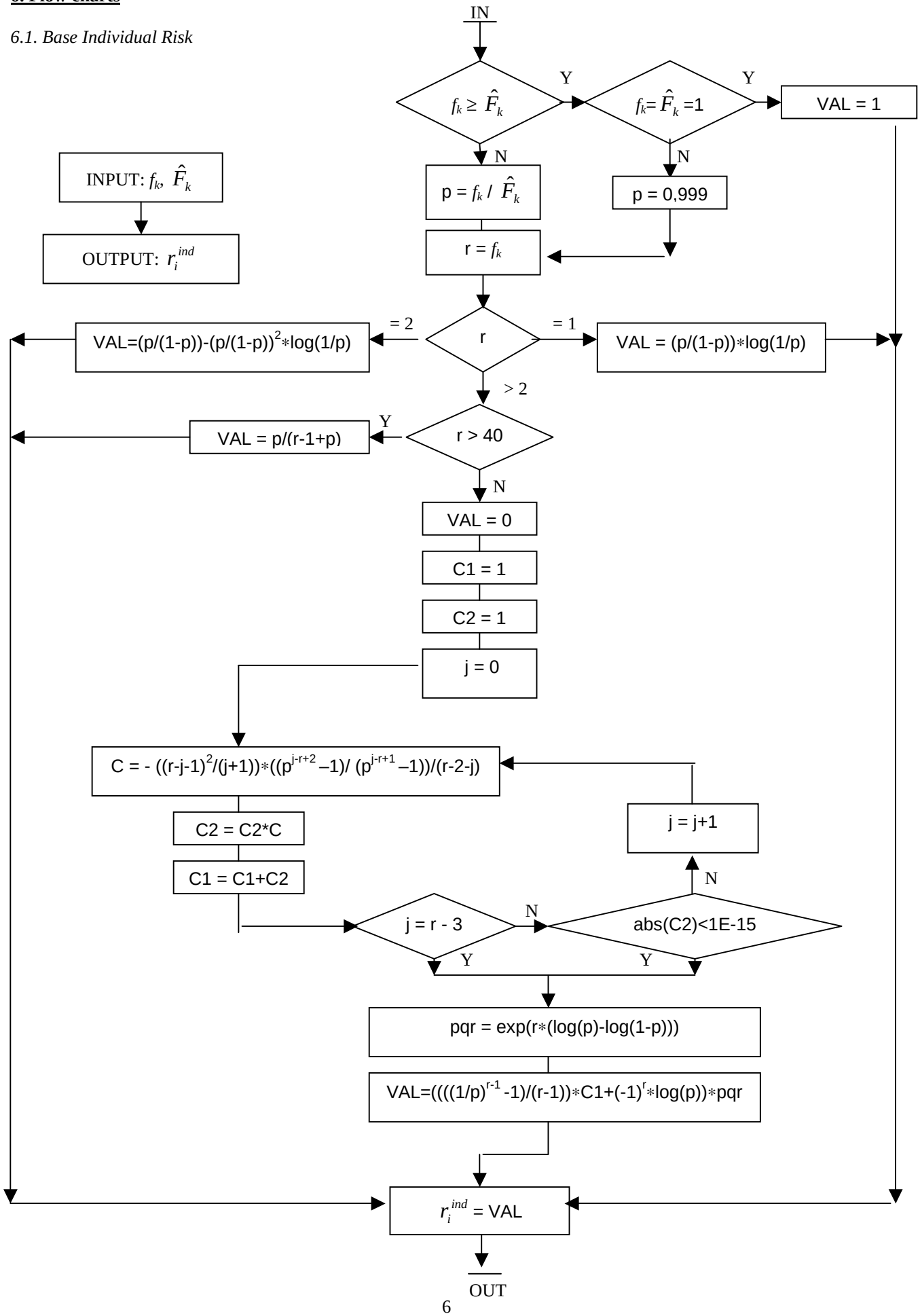
Finally, in order to consider other factors influencing the risk (such as the quality of the key variables, the intruding probability, and so on) we use a multiplying factor π so the final risk formula is given by:

$$\rho_i = \pi * r_{k(i)}^{ind} \tag{5}$$

The factor π, set to 1 as the default, should be requested to the user by an interactive window before the risk computation starts.

6. Flow charts

6.1. Base Individual Risk



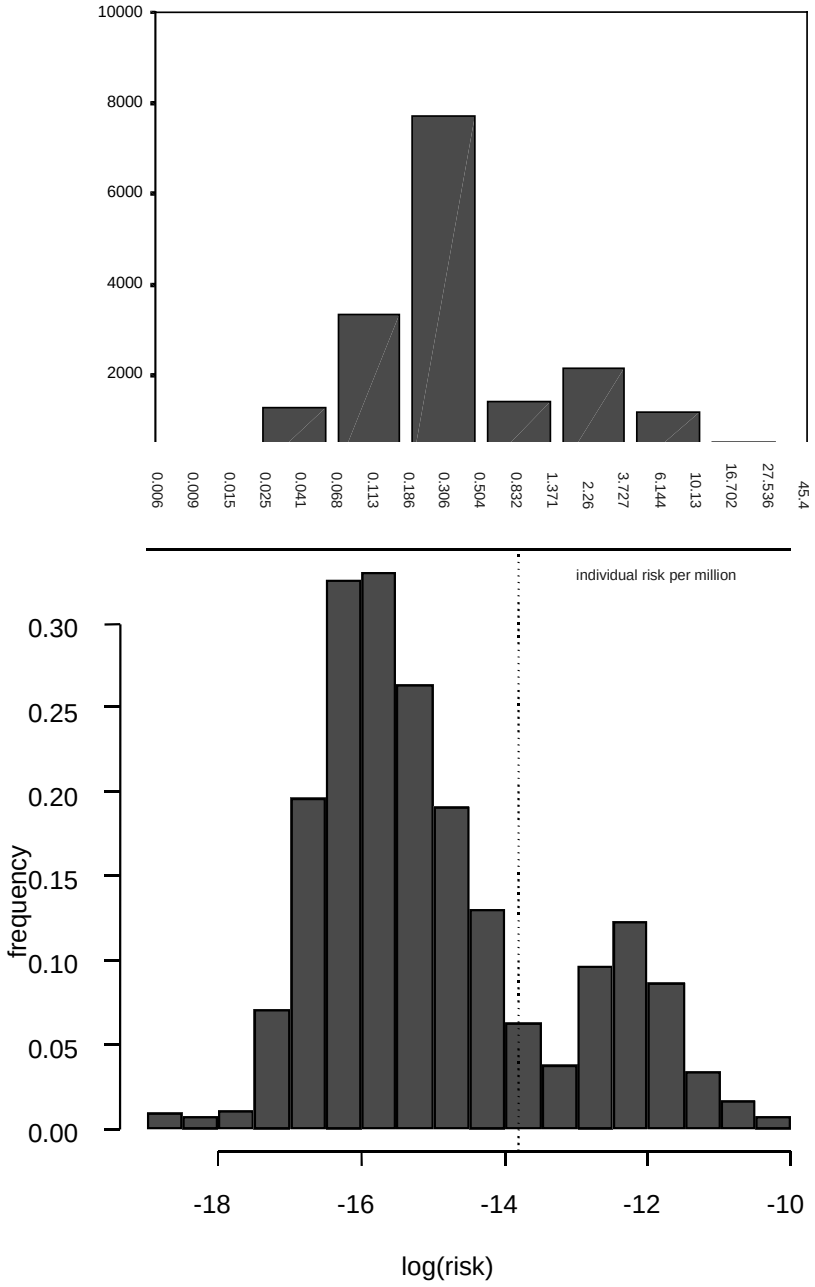
The previous algorithm has to run for each record of the input file.

6.2. Final Risk

The final risk value (ρ_i) is obtained multiplying the output of the previous algorithm (r_i^{ind}) by the π parameter (see formulas (1) in Section 5.) .

7. Graphics

After the evaluation of the final risk ρ_i , the user needs a graphic to fix the threshold α. We were thinking of it as a frequency histogram. By our experience, the graph could be clearer showing a logarithmic scale on the *x* axis (the one with the ρ_i values) or, which is the same, representing $log(\rho_i)$ instead of ρ_i . However, the labels on the axis should still report the corresponding ρ_i value, in order to better evaluate the appropriate α value.
Next we show, as examples, some risk graphs we used, though they do not perfectly correspond to the above description:



It would be useful to have both the graph and the warning message in the same window in which the user chooses the α value, so that as the value of α changes, the vertical line of the threshold shifts on the histograms and the warning message, if any, is refreshed.

8. Application of μ-Argus

After the final risk (ρ_i) has been evaluated for each record and the value of α has been chosen, the protection step follows through the local suppression method.

> As far as we know, in μ-Argus an optimised procedure is implemented, based on minimisation of the suppressions in the unsafe combinations. A combination is considered unsafe if it occurs not more than D_k times in the data set, where D_k is the *threshold* value.
>
> First, the procedure generates the combinations to be inspected following two possible alternatives: a) using the identification levels, b) generating all tables up to a given dimension. Then, after the unsafe combinations have been found, the procedure checks the presence of unsafe combinations in each record and chooses the suppression which minimises the number of suppressions (see 'μ-Argus ver. 2.5 User's Manual'; de Waal – Willenborg: 'Minimizing the Number of Local Suppression in a Microdata Set' - Proj M1-79-589, First Draft, May 31, 1994).

For the implementation of our methodology, we need to introduce some adjustments in μ-Argus protection strategy.

First of all, the identification rule must be changed: a *combination* of key variables is considered *unsafe* if the final risk ρ_i of an individual having that combination of attributes exceeds a given threshold α, which means that the D_k criterion used in μ-Argus is no more adequate.

Second, unsafe combinations are progressively identified via generation of all tables of any dimensions, which must proceed from dimension one up to the highest (*K*, the number of key variables in the data set)[1].

Notice that if a string is found unsafe, any string which contains the latter will be unsafe as well .

After the unsafe strings are singled out, the same protection algorithm already implemented in μ-Argus can be applied, producing the *μ-Argus output file.*

Recall that the final risk is $\rho_i = \pi * r_{k(i)}^{ind}$.

The base individual risk r_i^{ind} and hence ρ_i is nondecreasing in the number of key variables used for identification.

This allows us to apply the checking procedure starting from the *K* univariate contingency tables (step 1). The final risk is evaluated at each category of each of the *K* key variables. If the current value of ρ_i (based on one key variable only) exceeds α for a category, this category is considered unsafe and moreover each combination of key variables containing such category will be unsafe as well. Having selected only the current (step 1) safe strings, the algorithm proceeds in screening pairs of categories of key variables (step 2), identifying the unsafe pairs and so on, adding one dimension a time, up to the highest (step *K*). At each step *k*, the combinations containing a substring judged unsafe at step *k-1* are not screened, as they are certainly unsafe.

Alternatively, instead of the final risk ρ_i, the screening algorithm may check the individual risk r_i^{ind}, and compare it with the threshold α/π.

9. Safe file

Once the suppression algorithm has been applied, the risk calculation algorithm (Section 5) should run again on the output file produced by μ-Argus, in order to produce the new values of the risk after the protection step. Next, the graphics representing the current risk distribution (Section 7) have to be shown.

At this point the user can check the protection level attained, and he has two options:

a) *confirm*: the output file is recorded as the *safe file*;

b) *rollback*: the user is not satisfied by the results. He/she is now presented with different options, which can be applied one by one or in combination. He can: specify a different α value, and/or recode some variable.

[1] To reach this aim with μ-Argus ver. 2.5 we used either the identification levels (specifying for each key variable a different identification level) or the generation of all tables up to a given dimension (the highest).