WP. 35
ENGLISH ONLY

UNITED NATIONS STATISTICAL COMMISSION and ECONOMIC COMMISSION FOR EUROPE
CONFERENCE OF EUROPEAN STATISTICIANS

EUROPEAN COMMISSION
STATISTICAL OFFICE OF THE EUROPEAN COMMUNITIES (EUROSTAT)

Joint UNECE/Eurostat work session on statistical data confidentiality
(Bilbao, Spain, 2-4 December 2009)

Topic (vi): Case studies

TECHNIQUES FOR USING τ-ARGUS MODULAR ON SETS OF LINKED TABLES

Supporting Paper

Prepared by Sarah Giessing (Statistisches Bundesamt), Germany

Techniques for Using τ-Argus Modular on Sets of Linked Tables

Sarah Giessing
Statistisches Bundesamt, 65180 Wiesbaden, Germany, Email: Sarah.Giessing@destatis.de

Abstract: The software package τ-ARGUS offers very efficient algorithms for secondary cell suppression. In practice, statistical agencies release multiple tabulations based on the same dataset. Usually these tables are linked through certain linear constraints. In such a case cell suppressions must obviously be coordinated between tables. For the near future the release of a version of τ-ARGUS is planned that can handle cell suppression for linked tables. However, it has to be expected that – at least for the medium term future – the capacity of the software to deal with a large number of tables, or a large number of different table dimensions may still be limited. This paper explains a working solution that handles sets of linked tables by applying the τ-ARGUS modular method to each single table as a module within an iterative procedure. Two real life case studies are used to demonstrate the methodology, which is implemented prototypically in SAS.

1 Introduction

Some cells of the tabulations released by official statistics contain information that chiefly relates to single, or very few respondents which may often be easily identifiable. Therefore, traditionally, statistical agencies suppress a part of the data, hiding some table cells from publication. An efficient algorithm for secondary cell suppression is the modular optimization algorithm (a.k.a "HiTaS", see De Wolf, 2002) offered by the software package τ-ARGUS, see (Hundepool et.al, 2009a).
When tables are linked through simple linear constraints, cell suppressions must obviously be coordinated between tables. The most typical case is when tables share common cells (usually marginal totals), i.e., when they are linked through constraints saying literally that cell X of table A is identical to cell Y of table B. (DeWolf and Giessing, 2009) presents a collection of several alternative approaches for this coordination problem, one of which is now getting implemented in τ-ARGUS within the framework of a current joint European cooperation project[1]. Another one, referred to as "traditional method" will be explained in detail in the present paper. The "traditional method" can be used with the current version of τ-ARGUS modular method, and also for sets of linked tables that are too complex (involving f.i. too many classification variables, or too many different tables) to be handled even by the upcoming extended version of τ-ARGUS. However, we assume here that the tables are the result of a planned, co-ordinated process. For tables generated 'on the fly' on end user request the traditional method will usually be less suitable.

[1] See http://neon.vb.cbs.nl/casc/ESSNetindex.htm

The paper is organized as follows. Section 2 provides the methodological background for the "traditional method" and outlines the technical concept of a prototypical implementation. In section 3 we study the method at the example of two real-life applications. The paper concludes with a summary and comments.

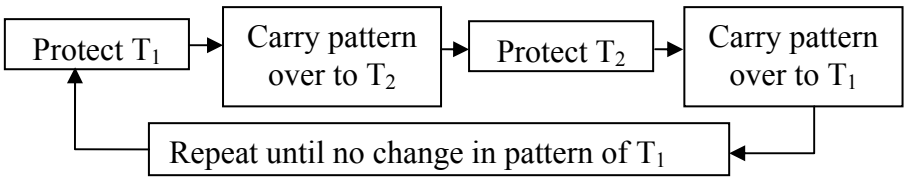
An extended version of this paper will eventually be made available on the ESSNET project website (Giessing, 2009), along with some of software codes discussed in section 2, examples for file formats, τ-ARGUS meta-data and control files. Note that for the remainder of this paper we assume that the reader has some basic familiarity with τ-ARGUS as well as with the basic concepts of automated tabular data protection as discussed f.i. in (Hundepool et al., 2009b).

2 The traditional method to co-ordinate cell suppression in a set of linked tables

The 'traditional' method is based on the idea of applying τ-ARGUS modular to each table within a set of linked tables separately within a backtracking procedure.
In case of two linked tables T_1 and T_2, the approach would be as follows:

1. Protect table T_1 on its own;
2. Each cell in T_2 that is also present in T_1 will get the status (i.e., suppressed or not-suppressed) of the cell in the protected table T_1;
3. Table T_2, with the additional suppressions carried over in step 2, is protected on its own.
4. Each cell in T_1 that is also present in T_2 will get the status of the cell in the protected table T_2;
5. Repeat step 1 – 4 until no changes occur in protecting table T_1 nor in protecting T_2.

Graphically the approach would look like:



Technically the τ-ARGUS package supports changing cell status by its <APRIORI> functionality. The functionality allows to turn cells status for a given list of cells (the so called "history file") into the status specified in such a history file.

2.1 Content of the history file

According to our concept for the traditional linked tables procedure the history file will generally consist of three parts:

Part 1: primary suppressions in the current table.
Part 2: Old secondary suppressions, i.e. all secondary suppressions in the current table resulting from previous protection steps (including those carried over from other tables)
Part 3: New secondary suppressions, i.e. all cells belonging to the current table which became suppressed in another table where they are also present since the current table was processed last time.

Note that the part 2 of the history file is important to speed up the procedure: in this way the suppression pattern of the previous run is re-used in the current run[2].
Regarding the construction of part 3, it is practical to build it 'on the fly', e.g. to store for every table in the set a list of new secondary suppressions. Whenever a cell that is also present in a particular table T_i is suppressed (when processing another table T_j), a line corresponding to this cell is added to the part 3 file for table T_i. Just before processing T_i for the next time, the complete history file for T_i is compiled by appending to the parts 1 and 2 all the cells in part 3 that have been collected since the previous run of T_i. After running T_i, we delete the corresponding part 3 file.

2.2 Which control options to choose for τ-ARGUS ?

First of all, in a procedure where tables are processed repeatedly, we make use of the option of supplying the data to τ-ARGUS in tabular format.
When looping table protection, it is essential to make sure that τ-ARGUS modular accepts a suppression pattern assigned in a previous step without adding new suppressions.
When protecting a table which has already been protected(!) in previous steps and where part 3 of the history file is empty, the current suppression step should not result in any new secondary suppressions. This can be achieved using the following control options

(1) Inactivate all *singleton options* of the modular method[3],
(2) Set protection levels for all cells of the history file to 1 (which is the smallest possible value). This can be achieved in the following way:
   - In advance of applying the history file information set the status of all cells of the current table to "safe",
   - Set the "manual range" parameter to 0
   - Do not specify any primary protection rule
(3) Ensure that the smallest absolute cell value in the table is larger than 1[4] (this can be achieved for instance by multiplying each cell in the table by a suitable constant).

[2] When protecting a table for the first time in the process we omit parts 1 and 2
[3] See (Hundepool et al., 2009a) for explanation on "singleton options"
[4] Otherwise a cell value 1 cell will be considered "too small" to protect even cells selected in a previous step to protect this cell value 1 cell.

When processing a table for the first time, but with history file information involving a non-empty part 3
(4) Inactivate the singleton/multiple option of the modular method

2.3 A general looping scheme for the 'traditional' method

Adding more tables to the set of linked tables, i.e., considering $\{T_1, T_2, \ldots, T_n\}$, $n > 2$, adds some complexity to this procedure. In that case several schemes can be thought of. The outcome of the procedure will depend on this scheme, e.g. the particular sequence in which the tables will be processed (eventually repeatedly)[5].
In practice, developing a table protection concept for a set of linked tables foreseen for a publication involves two crucial issues: one has to establish an adequate balance between the amount of detail in the tables on one hand, and the information loss caused by protection of the cell-suppressions of each single table in the other tables of the set on the other hand. The second issue is to establish a suitable sequence for processing the tables. It usually requires some experimentation to come up with a satisfactory concept. Especially during this experimentation stage it is crucial to record and evaluate information loss information. We need to observe information like: how many secondary suppressions in a table A are caused by protecting secondary suppressions of a table B in iteration step k? How often do we have to loop linked tables A, B, C until no additional suppressions occur?
In practice the situation is often that some tables are fixed – their design must not be changed, and they are considered highly important. Others could eventually be re-designed, if necessary. They are less relevant. Some tables have stronger links, i.e. suppressions in one table lead to many suppressions in another, some have weaker links, i.e. even though the tables may share cells that are logical identical, most of these cells usually remain unsuppressed anyway.
Out of these considerations, especially when a large number of tables is foreseen to be subject to the disclosure control procedure, it is sometimes natural to form groups of tables for the linked tables processing. In the following subsection we outline a software concept which takes into account such table-groupings.

2.3.1 Looping scheme for table groups

In the following, we denote as '*simple linked tables sequence'* for a set of linked tables $T_1, T_2, \ldots, T_n$ a sequence according to the following scheme outlined at the example of n=3:

[5] One must also bear in mind that this iterative approach could lead to a loop that keeps repeating until all cells of all tables are suppressed. In our experience, it is essential to apply this methodology only when tables are not too detailed. The experience is that detailed tables with a very high proportion of sensitive cells tend to cause too many loops of the procedure. Each loop adds new secondary suppressions, i.e. adds to the loss of data utility. It may then be better to think about relaxing the table structure (f.i. by recoding some of the tables). This means to sacrifice some low-level aggregated information for the sake of having a smaller number of suppressions in important high-level aggregated cells.

Protect T_1, carry pattern over, protect T_2, carry pattern over, protect T_3, carry pattern over, repeat until no changes are added.

Note that the step "carry pattern over" after protecting a table T_i involves the following two actions:

- for any newly assigned secondary suppression in T_i, add a line to all the part 3 files of those tables where this cell is also present[6] (c.f. outline of setting up part 3 files in sec. 2.1.1),
- create the history file for the subsequent table T_{i+1}[7] by appending the three parts to each other.

Let us assume now we are given with k groups of tables G_1 to G_k. Each group consists of n_j tables Tj1,Tj2,…,Tjnj. The looping scheme for table groups foresees to loop simple linked tables sequences in the following way:
Step 1: Carry out the simple looping sequence for G_1;
…
Step k (k > 2): (1) Carry out the simple looping sequence for G_k;
(2) Carry pattern over;
(3) Repeat step k-1;
(4) Carry pattern over;
Repeat (1) to (4) until no change in patterns of the tables in G_1 to G_k

2.4 Data preparation and work-flow

The first important step is to ensure that tables are defined properly. Logical identical categories of the same (explanatory) variable must have identical codes in every table. One should also seek to avoid disclosure by differencing problems: If by taking the difference between two non-sensitive (unsuppressed) cells sensitive information could be obtained, a table has to be introduced into the set of tables, which contains these differences[8].

[6] With respect to the algorithm for groups of tables outlined below, with "tables where this cell is also present" we also refer to tables of any of the other groups
[7] Re. the algorithm for groups of tables, note that Ti+1 may refer to the first table of the next group
[8] For example, assume there is a detailed-geography table presenting the number of cows, and (eventually another) table presenting the number of cows for milk production. Then by differencing the number of non-milk-production cows can be computed, whenever neither milk-production-cows nor total-cows is suppressed. If it is likely that this is for some geography cases (like small villages) sensitive information (perhaps relating to a single farm only) we need a table in the system of tables where one of the table relations expresses: milk-production cows + non-milk-production cows = Total cows.

2.4.1 Data preparation

For our prototypical software implementation in SAS, we require the user to supply initially one or more *basic SAS files* (say: S_1 *to* S_m) which serve as database and (jointly) contain all cells of every table to be protected by the method.

In our implementation we require the basic SAS files to contain explanatory variables Var_1 to Var_n for each of n table dimensions, the (usually quantitative) variable which should be the response variable for the τ-ARGUS suppression procedure, the number of respondents, eventually the largest contributor information used to assess cell sensitivity, cell status (after primary suppression) and protection level. This information can be obtained for instance by doing a preparatory run of τ-ARGUS on the table which carries out primary protection and saves the data in τ-ARGUS intermediate format. It is of course essential to define Var_1 to Var_n consistently across SAS files S_1 to S_m.

In addition to these SAS files the procedure requires a second set of SAS files, defining the table structures. For every table in the set there has to be such a *table-definition file* listing the cross-combinations of explanatory variables Var_1 to Var_n which define the cells of this table (i.e. one record per cell).

Finally a third set of SAS files will be needed to define the linked tables structure. This means for each pair of linked tables (T_i,T_j) we expect a *link-definition file* for this pair. This link-definition file can easily be obtained by merging the two separate table-definition files and creating output only of those records contained in both separate files.

Apart from those SAS files, the process has to be supplied with suitable τ-ARGUS meta-data files, e.g. the *.rda files with the file-description* of the τ-ARGUS input file, the *.hrc files explaining the table relations* (i.e. the logical relations between the categories of the explanatory variables) and the *.arb (batch) control files* for τ-ARGUS.

2.4.2 SAS macros

Actually, our current implementation consists of several SAS macros, which will be briefly sketched in the following. The general outline is that in a particular protection step for a particular table we extract a suitable τ-ARGUS input file from the initial SAS files S_1 to S_m, then create the τ-ARGUS history file outlined in 2.3.1, run , read back the τ-ARGUS output to SAS, create/update a SAS file that contains the results of all the previous protection steps for this table and, if a newly suppressed cell is also present in other tables, append a record to the part3 files (denoted in the following as SAS *P3 files*) for these tables.

%sas2tab extracts the data of a table T from one of the SAS files S_1 to S_m, by merging the table definition file of this table with the corresponding SAS file S_j. Output is created only for records contained in both data sets. The code supports several options for a suitable file format, i.e. which variables to be written to the output. The output file format is a suitable input format for τ-ARGUS.

%sas2hst creates parts 1 and 2 of the history file by selecting from the SAS results file for this table those records which are primary suppressions, or became secondary suppressions in any of the previous protections steps. Note, when a table is processed for the first time, parts 1 and 2 are empty.
To the parts 1 and 2 file we append the part 3 (i.e. SAS P3 file for this table). The macro creates from the resulting SAS file output in the τ-ARGUS history file format. Finally we delete the SAS P3 file for this table.

%ARGUS contains the call to τ-ARGUS[9]: `tauargus &arb test.tauargus.log`, where the SAS macro variable &arb provides (at run time) the name of the τ-ARGUS batch command file[10].

%csv2sas reads a τ-ARGUS .csv output file format into a SAS file.

%mergeResults merges this file with the SAS results file for the current table, adding (each time) a new variable to the SAS-file which contains the current cell status. Conditioning on cells that according to this variable were selected as secondary suppression in the current step and merging with each link-definition file (one by one), we create data sets with new suppressions from this step for those tables where these newly suppressed cells are also present. These data sets are eventually appended to the SAS P3 files - if already existing - for those tables.

%simple_linked_tables_sequence is the central control macro which sets the parameters for the other macros and runs them. It will carry out the sequence of macros for each table within a given group, and keeps looping until the P3 SAS file (with the new suppressions carried over from the other tables) is empty for every table in the group.

The main program loops the *%simple_linked_tables_sequence* macro across the groups of tables G_1 to G_k according to the following simple recursive algorithm:

%simple_linked_tables_sequence(G_k)
%Rep_proc(k)

where *%Rep_proc(l)* is defined as:

Do while $\bigcup_{\{j=1,\ldots,l\}} \bigcup_{\{i=1,\ldots,n_j\}} P3_{j,i}$ is non - empty:
%Rep_proc(l-1)
%simple_linked_tables_sequence(G_l)
end

---

[9] In our current implementation a special construct is needed. Because in our environment SAS is running on a Unix-server, the Windows software τ-ARGUS has to run on another machine. Therefore this macro contains facilities to copy the τ-ARGUS input data (table-file, history-file and meta-data files) to the Windows computer, initiate the τ-ARGUS run there, and – after completion – copy the τ-ARGUS output back to the SAS-server.

[10] See [Hundepool, sec. 5 for further information.

For *l*=1 *%Rep_proc* is defined as the *%simple_linked_tables_sequence* for the first group of tables (G_1).

3 Applications

This paragraph describes two real life applications for the methodology of section 2. The first instance is rather simple. It refers to a set of tables published for the German business tax statistics. The second instance with data from agriculture statistics is more complex, although the tables involved are much smaller than in the first instance.

3.1 Business tax statistics

The tabulations of business tax statistics data in the official publication that we protect with the methodology of the traditional method are specified on the basis of 3 explanatory variables: NACE[11], GEO[12] and SIZE. Respondents are grouped according to a (turnover) size class variable SIZE with 15 categories at 3 hierarchy levels.

Tables display sums of various indicators. Disclosure control was based on a p%-rule to identify sensitive cells. It was carried out only on tables displaying one specific lead indicator, because mostly for this indicator the risk is rather high that an enterprise dominating a cell total according to this indicator could be identified by this property (of being the 'largest' enterprise). The suppression pattern resulting from this process was then 'copied' to the tabulations of the other indicators.

The most important table T_1 is given by a combination of NACE down to the 5-digit level (:= NACE_5) and GEO down to the district level (:= GEO_distr, ca. 470 cat.). Another important table is T_2, given by a combination of NACE_5, GEO down to the state level (:= GEO_state, 17 cat.) and SIZE_1 (2 levels, 9 cat.). After some testing, it was decided also to include a third table into the linked-tables procedure. This table T_3 is given as cross-classification of NACE_sect[13]. x GEO_distr*[14] x SIZE_1[15].

[11] The NACE classification used here involves all sectors (A to O) in a breakdown to the 5-digit level (more than 1600 categories across a 7-level hierarchy)

[12] The most detailed breakdown by geography GEO considered here is down to the district level (more than 450 categories at 4 hierarchy levels).

[13] NACE_sect. denotes the simple breakdown of NACE to the section level (2 levels, 16 cats.)

[14] GEO_distr* a breakdown of GEO down to the district level for those German states who want a suppression pattern for T_3, and down to the state level for the other states. The background is that it has been observed that protecting T_3 leads to considerable increase in suppressions in T_1 especially on the highest level of NACE which is considered unacceptable for some states.

[15] Note, that there are more tables in that statistic that have to be protected. However, on those tables we use a more complex methodology which cannot be discussed in this paper. C.f. (Giessing, 2007)

It is practical to save the data for these 3 tables in a single basic SAS file S_1 (see 2.4.1)[16]. The 3 tables could be successfully processed by our methodology, considering them as one group (G_1) of tables in the sequence T_3, T_2 ,T_1, e.g. processing the two 3-dimensional tables first. Dealing with only one group we had to carry out *%simple_linked_tables_sequence(G_1)* only once. After processing T_3 and T_2 for the second time the process finished successfully: all P3 files were empty. While computing times were about 2 and 3.5 hours for tables T_3 and T_2 in the first run, the repetition runs took only 3 and 8 minutes.

3.2 Cattle register data

Tabulations to be published on the basis of the German cattle register involve a larger number of explanatory variables than the business tax statistics tables of 3.1. While they also involve the variable GEO, the most important explanatory variable is variable TYPE (4 levels, 11 cat.). TYPE is a breakdown of cattle into age/production type groups (with categories like f.i. "cows for milk-production", or "cattle under age 1"). For the 5 most important categories of TYPE separate herd size-classes are defined, e.g. Variables S_1 to S_5 (each at 2 levels, up to 8 cat.). In addition to that we have variables RACE (3 levels, 26 cat.) and SEX (2 levels, 3 cat.). All tables present numbers of animals (which is the response variable). See figure 1 for the definition of the 9 tables T_1 to T_9 processed within a looping scheme considering 4 groups of tables.

Table	Group	Table-Specification
T_1	1	GEO_distr x TYPE x SEX
T_2	1	GEO_distr x RACE
T_3	1	GEO_Nuts2[17] x TYPE x SEX x RACE[18]
T_4,T_5	2	GEO_distr x S1, GEO_distr x S2
T_6	3	GEO_comm[19] x TYPE_1
T_7, T_8, T_9	4	GEO_distr x S3, GEO_distr x S4, GEO_distr x S4

Figure 1: Table specifications and groups

[16] This SAS file could for instance be obtained by generating the table NACE x SIZE_1 x GEO_distr with ARGUS. Note that we do not do disclosure control for this (big!) table because it yields too many suppressions in table T_1.

[17] Nuts2: geography level just above district

[18] According to this definition, T_3 would be a 4-dimensional table. However, by dropping all cells which are not in the margin of SEX, except when the cell is also on the lowest level of TYPE, the table can be transformed into a 3-dimensional table, where SEX is only an additional hierarchy level of TYPE

[19] Community level: ca. 10000 cat.

The main issues when grouping the tables was to collect - during processing - information loss data. We wanted to know which of the less important tables results in how many suppressions in the most important table T_1.
The outcome of the recursive algorithm outlined at the end of 2.4.2 was a sequence of *%simple_linked_tables_sequence* runs for those groups in the following order: G_1, G_2, G_1, G_3, G_4, G_1. Each of these *%simple_linked_tables_sequence* runs finished after the first loop. In fact, T_1 is the only table, for which –after processing it for the first time – we obtained non-empty P3 files.

4 Summary and Final Comments

This paper has described the 'traditional method' to co-ordinate cell suppression when protecting set of linked tables by using the Modular method which currently cannot be applied directly to sets of linked tables, or tables with non-nested hierarchies. While the approach is straightforward, in practice applications can be tedious, especially when dealing with larger sets of tables. In this paper a prototypical implementation of this method has been described. Section 3 has illustrated the outcome of the procedure at the example of two real-life applications. This implementation consists of a set of macros written in SAS. The idea was to design a tool that can be applied to sets of linked tables from a wide range of statistics without requiring adaptation (except from further development, of course).
Necessary extensions of our software tool concern the production of τ-ARGUS meta-data and control-files and the collection of information loss information during the process.

Acknowledgements

This research was partially supported by the European Commission under grant agreement No 25200.2005.003-2007.670: 'ESSnet on statistical disclosure control'

References

De Wolf, P.P. and Giessing, S. (2008), '*How to make the τ-ARGUS Modular Method Applicable to Linked Tables*', in *Privacy in Statistical Databases*, Domingo-Ferrer and Saygin (Eds.), Springer LNCS 5262, pp 37-50

De Wolf, P.P. (2002), '*HiTaS: A Heuristic Approach to Cell Suppression in Hierarchical Tables*', In: '*Inference Control in Statistical Databases*' Domingo-Ferrer (Ed.), Springer (Lecture notes in computer science; Vol. 2316)

Giessing, S. (2009), '*Report: Techniques for using τ-ARGUS Modular on Sets of Linked Tables,* to appear at http://neon.vb.cbs.nl/casc/handbook.htm#casestudies

Hundepool, A., van de Wetering, A., Ramaswamy, R., de Wolf, P.P., Giessing, S., Fischetti, M., Salazar, J.J., Castro, J. and Lowthian, P. (2009a), τ-ARGUS *user's manual, version 3.3*, available at http://neon.vb.cbs.nl/casc/tau.htm

Hundepool, Anco, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Rainer Lenz, Jane Longhurst, Eric Schulte Nordholt, Giovanni Seri and Peter-Paul De Wolf (2009b), *ESSNET handbook on Statistical Disclosure Control,* ESSNET-SDC project, available at http://neon.vb.cbs.nl/casc/handbook.htm