

τ

ARGUS

version 2.1



User's Manual

Document: 4.2-D1
Project: CASC-project
Date: February 2002

Statistics Netherlands
P.O. Box 4000
2270 JM Voorburg
The Netherlands
email: ahnl@krypton.vb.cbs.nl

Contributors: Anco Hundepool and Aad van de Wetering.
Peter-Paul de Wolf (Hitas), Sarah Giessing (GHMiter)
Matteo Fischetti, Juan-José Salazar and Alberto Caprara (Optimisation)

Contents

Preface.....	3
About the name ARGUS.....	3
Contact	4
Acknowledgments	4
1. Introduction.....	6
2. Producing safe tables	6
2.1 Sensitive cells in magnitude tables.....	6
2.2 Sensitive cells in frequency count tables.....	7
2.3 Table redesign.....	8
2.4 Secondary cell suppression.....	8
2.5 Information loss in terms of cell weights	8
2.6 Series of tables.....	9
2.7 The Hypercube/GHMITER method	9
2.7.1 The method	9
2.8 The ARGUS implementation of GHMITER.....	9
2.8.1 Confirmed errors: The warning file.....	10
2.8.2 An illustrative example	11
2.8.3 References on GHMiter	11
2.9 Hitas.....	12
2.10 Functional design of τ -ARGUS.....	14
3. A tour of τ -ARGUS	15
3.1 Preparation.....	15
3.1.1 The Metafile.....	16
3.1.2 Specification.	18
3.2 The process of disclosure control.....	19
3.2.1 Table protection	20
3.3 Saving the safe table.....	23
4. Description of the Menu Items	24
4.1 Main Window	24
4.2 The File menu	25
4.2.1 File Open Microdata.....	25
4.2.2 File Exit.....	25
4.3 Specify	25
4.3.1 Specify Metafile	25
4.3.2 Specify Tables.....	27
4.4 Modify	29
4.4.1 Modify Select Table.....	29
4.4.2 Modify View Table	29
4.4.2.1 Recoding a non-hierarchical variable	30
4.4.2.2 Recoding a hierarchical variable.....	31
4.5 Output	34
4.5.1 Output Save Table.....	34
4.5.2 Output Report.....	34
4.6 Help.....	35
4.6.1 Help Contents.....	35
4.6.2 Help About	35

Preface

This is the user's manual of τ -ARGUS version 2.1. τ -ARGUS is a software tool designed to assist a data protector in producing safe tables. This version is the first release of τ -ARGUS in the CASC-project. With respect to the τ -ARGUS version from the previous project we have made a major step forward and τ -ARGUS has now facilities to protect hierarchical tables.

The purpose of τ -ARGUS is to protect tables against the risk of disclosure, i.e. the accidental or deliberate disclosure of information related to individuals from a statistical table. This is achieved by modifying the table so that it contains less, or less detailed, information. τ -ARGUS allows for several modifications of a table: a table can be redesigned, meaning that rows and columns can be combined; sensitive cells can be suppressed and additional cells to protect these can be found in some optimum way (secondary cell suppression).

τ -ARGUS is one of a twin set of disclosure control packages. Within the CASC-project a tool for microdata - called μ -ARGUS - is also being developed, which is the twin brother of τ -ARGUS.¹ This is manifest not only when one looks at the (user inter)faces of both packages, but also when one would look at the source code: the bodies of the twins are so much combined that they in fact are like Siamese twins!

About the name ARGUS

Somewhat jokingly the name ARGUS can be interpreted as the acronym of 'Anti-Re-identification General Utility System'². As a matter of fact, the name ARGUS was inspired by a myth of the ancient Greeks. In this myth Zeus has a girl friend named Io. Hera, Zeus' wife, did not approve of this relationship and turned Io into a cow. She let the monster Argus guard Io. Argus seemed to be particularly well qualified for this job, because it had a hundred eyes that could watch over Io. If it would fall asleep only two of its eyes were closed. That would leave plenty of eyes to watch Io. Zeus was eager to find a way to get Io back. He hired Hermes who could make Argus fall asleep by the enchanting music on his flute. When Hermes played his flute to Argus this indeed happened: all its eyes closed, one by one. When Hermes had succeeded in making Argus fall asleep, Argus was decapitated. Argus' eyes were planted onto a bird's tail - a type of bird that we now know under the name of peacock. That explains why a peacock has these eye-shaped marks on its tail. This also explains the picture on the cover of this manual. It is a copper-plate engraving of Gerard de Lairesse (1641-1711) depicting the process where the eyes of Argus are being removed and placed on the peacock's tail.³

Like the mythological Argus, the software is supposed to guard something, in this case data. This is where the similarity between the myth and the package is supposed to end, as we believe that the package is a winner and not a loser as the mythological Argus is.

¹ See Anco Hundepool et al., 2002, μ -ARGUS version 3.1 user's manual, Statistics Netherlands, Voorburg, The Netherlands.

² This interpretation is due to Peter Kooiman, and dates back to around 1992 when the first prototype of ARGUS was being built by Wil de Jong.

³ The original copy of this engraving is in the collection of 'Het Leidsch Prentenkabinet' in Leiden, The Netherlands.

Contact

Feedback from users will help improve future versions of τ -ARGUS and is therefore greatly appreciated. The authors of this manual can be contacted directly for suggestions that may lead to improved versions of τ -ARGUS in writing or otherwise; e-mail messages can also be sent to argus@cbs.nl.

Acknowledgments

τ -ARGUS has been developed as part of the CASC project that was partly sponsored by the EU under contract number IST-2000-25069. This support is highly appreciated. The CASC (Computational Aspects of Statistical Confidentiality) project is part of the Fifth Framework of the European Union. The main part of τ -ARGUS has been developed at Statistics Netherlands by Aad van de Wetering (who wrote the kernel) and Anco Hundepool (who wrote the interface). However this software would not have been possible without the contributions of several others, both partners in the CASC-project and outsiders.

The German partners Statistisches Bundesamt (Sarah Giessing and Dietz Repsilber) have contributed the GHMITER software, which offers a solution for secondary cell suppression based on hypercubes. Peter-Paul de Wolf has build a search algorithm based on the non-hierarchical optimal solutions. This algorithm will break down a large hierarchical table into small non-hierarchical subtables, which will then be protected. The optimisation routines have been developed by JJ Salazar cs. of the University La Laguna Tenerife, Spain.

For solving these optimisation problems τ -ARGUS uses commercial LP-solvers. Traditionally we use Xpress as an LP-solver. This package is kindly made available for users of τ -ARGUS at a special agreement between the τ -ARGUS-team and DASH-optimisation, the developers of Xpress. Alternatively τ -ARGUS can also use the Cplex-package. Also with CPLEX there is an arrangement for the use of τ -ARGUS. It is the choice of the users. However users having a licence for one of these packages can use their current licence for τ -ARGUS as well.

The CASC-project

The CASC project on the one hand can be seen as a follow up of the SDC-project of the 4th Framework. It will build further on the achievements of that successful project. On the other hand it will have new objectives. It will concentrate more on practical tools and the research needed to develop them. For this purpose a new consortium has been brought together. It will take over the results and products emerging from the SDC-project. One of the main tasks of this new consortium will be to further develop the ARGUS-software, which has been put in the public domain by the SDC-project consortium and is therefore available for this consortium. The main software developments in CASC are μ -ARGUS, the software package for the disclosure control of microdata while τ -ARGUS handles tabular data.

The CASC-project will involve both research and software development. As far as research is concerned the project will concentrate on those areas that can be expected to result in practical solutions, which can then be built into (future version of) the software. Therefore the CASC-project has been designed round this software twin ARGUS. This will make the outcome of the research readily available for application in the daily practice of the statistical institutes.

CASC-partners

At first sight the CASC-project team had become rather large. However there is a clear structure in the project, defining which partners are working together for which tasks. Sometimes groups working closely together have been split into independent partners only for administrative reasons.

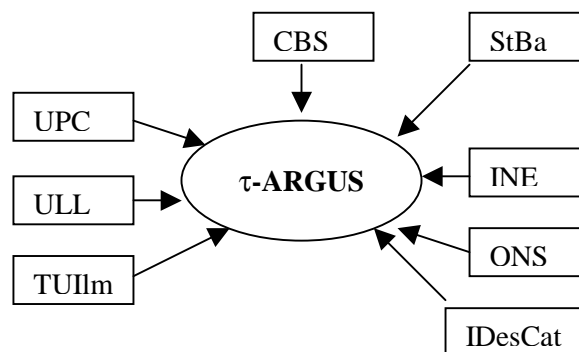
Institute	Short	Country
1. Statistics Netherlands	CBS	NL
2. Istituto Nazionale di Statistica	ISTAT	I
3. University of Plymouth	UoP	UK
4. Office for National Statistics	ONS	UK
5. University of Southampton	SOTON	UK
6. The Victoria University of Manchester	UNIMAN	UK
7. Statistisches Bundesamt	StBA	D
8. University La Laguna	ULL	ES
9. Institut d'Estadística de Catalunya	IDESCAT	ES
10. Institut Nacional de Estadística	INE	ES
11. TU Ilmenau	TUilm	D
12. Institut d'Investigació en Intel·ligència Artificial-CSIC	CIS	ES
13. Universitat Rovira i Virgili	URV	ES
14. Universitat Politècnica de Catalunya	UPC	ES

Although Statistics Netherlands is the main contractor, the management of this project is a joint responsibility of the steering committee. This steering committee constitutes of 5 partners, representing the 5 countries involved and also bearing a responsibility for a specific part of the CASC-project:

CASC Steering Committee

Institute	Country	Responsibility
Statistics Netherlands	Netherlands	Overall manager Software development
Istituto Nazionale di Statistica	Italy	Testing
Office for National Statistics	UK	
Statistisches Bundesamt	Germany	Tabular data
Universitat Rovira i Virgili	Spain	Microdata

The CASC tabular data team



1. Introduction

The growing demands from researchers, policy makers and others for more and more detailed statistical information leads to a conflict. The statistical offices collect large amounts of data for statistical purposes. The respondents are only willing to provide the statistical offices with the required information if they can be certain that these statistical offices will treat their data with the utmost care. This implies that their confidentiality must be guaranteed. This imposes limitations on the amount of detail in the publications. Practice and research have generated insights into how to protect tables, but the problem is certainly not definitively settled.

Before we go into more details, the basic ideas on which τ -ARGUS is based, we give a sketch of the general ideas. At first sight one might find it difficult to understand that information presented in tabular form presents a disclosure risk. After all one might say that the information is presented only in aggregate form.

2. Producing safe tables

Safe tables are produced from unsafe ones by applying certain SDC measures to these tables. In the current section these SDC measures - as far as they are implemented in τ -ARGUS - are discussed in the present section. Some key concepts such as sensitive cells, information loss and the like are discussed as well.

2.1 Sensitive cells in magnitude tables

The well-known dominance rule is often used to find the sensitive cells in tables, i.e. the cells that can not be published as they might reveal information on individual records. More particularly, this rule states that a cell of a table is unsafe for publication if a few (n) major contributors to a cell are responsible for a certain percentage (p) of the total of that cell. The idea behind this rule is that in that case at least the major contributors themselves can determine with great precision the contributions of the other contributors to that cell. The choice $n=3$ and $p=70\%$ is not uncommon, but τ -ARGUS will allow the users to specify their own choice.

As an alternative the prior-posterior rule has been proposed. The basic idea is that a contributor to a cell has better chances to estimate the competitors in a cell than an outsider and also that these kind of intrusions can occur rather often. The precision with which a competitor can estimate is a measure of the sensitivity of a cell. The worst case is that the second largest contributor will be able to estimate the largest contributor. If this precision is more than $p\%$ the cell is considered unsafe. An extension is that also the global knowledge about each cell is taken into account. In that case we assume that each intruder has a basic knowledge of the value of each contributor of $q\%$

With these rules as a starting point it is easy to identify the sensitive cells, provided that the tabulation package has the facility not only to calculate the cell totals, but also to calculate the number of contributors and the n individual contributions of the major contributors. Tabulation packages like ABACUS (from Statistics Netherlands) and the Australian package SuperCross have that capacity. In fact τ -ARGUS not only stores the sum of the n major contributions for each cell, but the individual contributions themselves. The reason for this is that this is very handy in case rows and columns etc. in a table are combined. By merging and sorting the sets of individuals contributions of the cells to be combined, one can quickly determine the major contributions of the new cell, without going back to the original file. This implies that one can quickly apply the dominance rule to the combined cells. Combining rows and columns (table redesign) is one of the major instruments to reduce the number of unsafe cells.

This too is the reason why τ -ARGUS only read microdata files. However it is to be expected that in future versions of τ -ARGUS we will include options for reading ready made tables, with the restriction that the options for table redesign cannot be used any more.

A problem, however, arises when also the marginals of the table are published. It is no longer enough to just suppress the sensitive cells, as they can be easily recalculated using the marginals. Even if it is not possible to exactly recalculate the suppressed cell, it is possible to calculate an interval that contains the suppressed cell. This is possible if some constraints are known to hold for the cell values in a table. A common found constraint is that the cell values are all nonnegative.

If the size of such an interval is rather small, then the suppressed cell can be estimated rather precisely. This is not acceptable either. Therefore it is necessary to suppress additional information to achieve that the intervals are sufficiently large.

Several solutions are available to protect the information of the sensitive cells:

- Combining categories of the spanning variables (table redesign). Larger cells tend to protect the information about the individual contributors better.
- Suppression of additional (secondary) cells to prevent the recalculation of the sensitive (primary) cells.

The calculation of the optimal set (with respect to the loss of information) of secondary cells is a complex OR-problem. τ -ARGUS will be build around this solution and takes care of the whole process. A typical τ -ARGUS session will be one in which the users will first be presented with the table containing only the primary unsafe cells. The user can then choose how to protect these cells. This can be the combining of categories, equivalent to the global recoding of μ -ARGUS . The result will be an update of the table with presumably less unsafe cells (certainly not more). At a certain stage the user requests the system to solve the remaining unsafe cells by finding secondary cells to protect the primary cells.

At this stage the user can choose between several options to protect the primary sensitive cells. Either he chooses the hypercube method or the optimal solution. In this case he also has to select the solver to be used, Xpress or Cplex. After this the table can be stored and can be published.

2.2 Sensitive cells in frequency count tables

In its simplest way sensitive cells in frequency count tables are defined as those cells that contain a frequency that is below a certain threshold value. This threshold value is to be provided by the data protector. This way of identifying unsafe cells in a table is the one that is implemented in the current version of τ -ARGUS It should be remarked, however, that this is not always the adequate way to protect a frequency count table.⁴ Yet it is one that is applied a lot, and one that follows from applying the dominance rule to frequency count tables. Rather than mechanically applying a dominance rule or a p% rule one should think about possible disclosure risks that a frequency count table poses and possible disclosure scenarios in order to simulate the behaviour of an intruder. Such an analysis would probably come up with different insights than using a simple thresholding rule, e.g. like the one sketched in the reference just mentioned.

⁴ See for instance Leon Willenborg and Ton de Waal, 1996, Statistical disclosure control in practice, Springer-Verlag, New York, Section 6.3.

2.3 Table redesign

In case in a table a great many sensitive cells appear to exist, it might be an indication that the spanning variables of the table are too detailed. In that case one could consider the possibility of combining certain rows and columns in the table. (This might not always be possible, from a publication policy point of view.) Otherwise the amount of secondary cell suppressions might just be too enormous. The situation is comparable to the case of microdata containing a lot of unsafe combinations. Rather than eliminating them with local suppressions one can remove them by using global recodings. In case of tables we use the phrase “table redesign” to denote an operation analogous to global recoding in microdata sets. The idea of table redesign is to combine rows, columns etc., by adding the cell contents of corresponding cells from the different rows, columns etc. It is a property of the dominance rule that a joint cell is more safe than any of the individual cells. So as a result of this operation the number of unsafe cells is reduced. One can try to eliminate all unsafe combinations in this way, but that might lead to an unacceptably high information loss. Instead, one could stop at some point, and eliminate the remaining unsafe combinations by using other techniques such as cell suppression.

2.4 Secondary cell suppression

Once the sensitive cells in a table - either of magnitude or a frequency count type - have been identified and there are not too many of these it might be a good idea to suppress these values. In case no constraints on the possible values in the cells of a table exist this is easy: one simply removes the cell values concerned and the problem is solved. In practice, however, this situation hardly ever occurs. Instead one has constraints on the values in the cells due to the presence of marginals and lower bounds for the cell values (typically 0). The problem then is to find additional cells that should be suppressed in order to protect the sensitive cells. The additional cells should be chosen in such a way that the interval of possible values for each sensitive cell value is sufficiently large. What is “sufficiently large” is to be specified by the data protector by specifying the protection intervals.

In general the secondary cell suppression problem turns out to be a hard problem, provided the aim is to retain as much information in the table as possible, which, of course, is a quite natural requirement. The optimisation problems that will then result are quite difficult to solve and require expert knowledge in the area of combinatorial optimisation.

2.5 Information loss in terms of cell weights

In case of secondary cell suppression it is possible that a data protector might want to differentiate between the candidate cells for secondary suppression. It is possible that he would like to preserve the content of certain cells as much as possible and is willing to sacrifice the values of certain other cells instead. A mechanism that can be used to make such a distinction between cells in a table is that of cell weights. In τ -ARGUS it is possible to associate different weights with the cells in a table. The higher the weight the more important the corresponding cell value is considered and the less likely it will be suppressed. We shall interpret this by saying that the cells with the higher associated weights have a higher information content. The aim of secondary cell suppression can be summarised by saying that a safe table should be produced from an unsafe one, by minimising the information loss, expressed as the sum of the weights associated with the cells that have secondarily been suppressed.

τ -ARGUS offers several ways to compute these weights. The first option is to compute these weights as the sum of the contributions to a cell. Secondly this weight can be the frequency of the contributors to a cell, and finally each cell can be weighted as one, minimising the number of suppressed cells.

2.6 Series of tables

In τ -ARGUS it is possible to specify a series of tables that will be protected one by one, and independently of each other. It is more efficient to choose this option since τ -ARGUS requires only a single run through the microdata in order to produce the tables. But also for the user it is often more attractive to specify a series of tables and let τ -ARGUS protect them in a single session, rather than have several independent sessions.

2.7 The Hypercube/GHMITER method⁵

In order to ensure tractability also of big applications, τ -ARGUS interfaces with the GHMITER hypercube method of R. D. Repsilber of the Landesamt für Datenverarbeitung und Statistik in Nordrhein-Westfalen/Germany, offering a quick heuristic solution. The method has been described in depth in [1] and [2], for a briefer description see [3].

2.7.1 The method

The approach builds on the fact that a suppressed cell in a simple n-dimensional table without substructure cannot be disclosed exactly if that cell is contained in a pattern of suppressed, nonzero cells, forming the corner points of a hypercube.

The algorithm subdivides n-dimensional tables with hierarchical structure into a set of n-dimensional sub-tables without substructure. These sub-tables are then protected successively in an iterative procedure that starts from the highest level. Successively, for each primary suppression in the current sub-table, all possible hypercubes with this cell as one of the corner points are constructed.

For each hypercube, a lower bound is calculated for the width of the suppression interval for the primary suppression, that would result from the suppression of all corner points of the particular hypercube. To compute that bound, it is not necessary to implement the time consuming solution to the Linear Programming problem. If it turns out that the bound is sufficiently large, the hypercube becomes a feasible solution. For any of the feasible hypercubes, the loss of information associated with the suppression of its corner points is calculated. The particular hypercube that leads to minimum information loss is selected, and all its corner points are suppressed.

After all sub-tables have been protected once, the procedure is repeated in an iterative fashion. Within this procedure, when cells belonging to more than one sub-table are chosen as secondary suppressions in one of these sub-tables, in further processing they will be treated like sensitive cells in the other sub-tables they belong to.

It should be mentioned here that the ‘hypercube criterion’ is a sufficient but not a necessary criterion for a ‘safe’ suppression pattern. Thus, for particular subtables the ‘best’ suppression pattern may not be a set of hypercubes – in which case, of course, the hypercube method will miss the best solution and lead to some overprotection. So, there is some tendency for over-suppression connected to this method.

2.8 The ARGUS implementation of GHMITER

The following issues are important to note:

- In the implementation offered by ARGUS, GHMITER makes sure that a single respondent cell will never appear to be corner point of one hypercube only, but of two hypercubes at least. Otherwise it could happen that a single respondent, who often can be reasonably assumed to know that he is the only respondent, could use his knowledge on the amount of his own contribution to recalculate the value of any other suppressed corner point of this hypercube.
- For tables presenting magnitude data, ARGUS will ensure that GHMITER selects secondary suppressions that protect the sensitive cells properly. It is assumed that users of the table can

⁵ The section on GHMITER has been contributed by Sarah GIESSING, *Federal Statistical Office of Germany 65180 Wiesbaden E-mail: sarah.giessing@destatis.de*

estimate any cell value to within $\pm q$ % of its actual value in advance of the publication, when the disseminator uses the Prior/Posterior-rule with parameters p and q , and to within ± 100 % when a (n,k)-dominance rule is used. GHMITER will select secondary suppressions in such a way that a user of the resulting protected table when using, apart from the assumed *a priori* information, only information provided by the data of the protected table would normally not be able to derive any bounds for the contribution of any respondent to a particular sensitive cell close enough to disclose this contribution according to the primary sensitivity rule in use.

- The standard information loss measure of GHMITER (in the following also referred to as ‘costs’ for suppressing a cell) is proportional to the logarithm of the cell value. This measure is used in combination with some heuristic approaches to distinct between several categories of cells, such as suppressed and (so far) unsuppressed cells, and single contributor cells. The costs for suppressing a hypercube that contains at least one unsuppressed cell, for instance, will at any rate exceed the costs for a hypercube containing only cells that are already suppressed. The ARGUS implementation makes use of the GHMITER option to assign high additional costs to cells in the margins of a subtable to avoid their suppression to the extent possible.
- All cell values must be positive. The current ARGUS implementation does not support the capability of GHMITER to process tables containing negative cell values.

2.8.1 Confirmed errors: The warning file

GHMITER checks the feasibility of each hypercube, one by one. The method is unable to ‘add’ the protection given by multiple hypercubes. In certain situations it is not possible to protect a particular sensitive cell by suppression of one single hypercube (for illustration, see example below). In such a case, GHMITER is unable to confirm that this cell has been protected properly. It will then create a warning file⁶. Note, that even when this file was created, there is still a chance that suppression pattern may be valid because the primary suppressions in question are protected properly by a combination of multiple hypercubes in the suppression pattern.

The user is requested to study the warning file carefully. The file lists those suppressions the protection of which could not be confirmed. These ‘target’ suppressions are referred to in the warning file as ‘PIVOT’. For any PIVOT the protection of which could not be confirmed, the file successively lists all candidate suppression hypercubes, together with some information why this candidate does not seem to give sufficient protection. The first line that refers to a particular candidate hypercube displays the cell values of the PIVOT and the complementary suppressions (X^{th} QUAW). Note that due to rounding errors, displayed values may not exactly match the actual values. When checking the warning file, the user will usually find that for each of the candidate hypercubes either

- the range ratio (displayed as QUOTE in the warning file) is below the minimum feasible value as displayed in the ARGUS report (GHMITER uses a ‘range ratio’ as measure for the protection given by a suppression hypercube), or
- the flag AB is different from zero, indicating that the candidate was rejected because it contains a
 - zero value cell (AB=1.),
 - single respondent/statistical unit cell (AB=2. or 3.), or
 - pre-published cell (cell value assumed to be known in advance of this publication) (AB=4.)

⁶ The file will not be produced for tables with more than four dimensions or more than 50 000 cells.

2.8.2 An illustrative example

Figure 1	Total	A	B
Total	136	20	116
I	26	10	16
II	110	10	100

For primary protection of the data presented in figure 3 a p%-rule with parameter $p = 15$ has been used, resulting in the two primary suppressions shaded grey in the figure. GHMITER will in that case use a range ratio of 30 %. That means GHMITER will consider as feasible suppression hypercubes only those that give a protection range of 30 % to any primary suppression.

Let us assume now, the value 116 for the second column total 'B' were already published, and therefore not available for secondary suppression. When this information is passed to GHMITER, the program will not be able to find a valid suppression pattern. In figure 1 the cell values for the (single) hypercube that results in the largest range ratio are printed in italics. As mentioned above, it is generally assumed, that users of the table are able to estimate any cell to within ± 100 %. So, if the table were published with only the cell values in italics suppressed, the users could estimate the value for cell (II,B) to within $[0;32]$ and use these estimates to deduce an interval of $[94;126]$ for the confidential value of the second row total (II). The width of this interval is $126 - 94 = 32$, which is 29.1 % of 110 . As 29.1 is still below 30, GHMITER rejects this solution. Find below a representation of those lines in the warning file referring to the discussed figure 1 suppression pattern.

```

W: 110. 100. 26. 16. -210.01 0.00
-- PIVOT 1.QUA 2.QUAW 3.QUAW AB 1 GRENZE 2
GRENZE QUOTE SUMME K: -295121.-193192. -765938. -656113. 0. 16.
16. 0.291 0.00E+00 R:0.00E+00 -3.00E+05 -3.30E+05 -3.30E+05 0.
0 0. 0.000 0.00E+00 I: 0.00E+00 -3.00E+05 -3.30E+05 -3.30E+05 0
0. 0. 0.000 0.00E+00 G: 0.00E+00 -3.00E+05 -3.30E+05 -3.30E+05 0.
0. 0. 0.000 0.00E+00

```

2.8.3 References on GHMiter

- [1] Repsilber, R. D. (1994), 'Preservation of Confidentiality in Aggregated data', paper presented at the Second International Seminar on Statistical Confidentiality, Luxemburg, 1994
- [2] Repsilber, D. (1999), 'Das Quaderverfahren' - in *Forum der Bundesstatistik, Band 31/1999: Methoden zur Sicherung der Statistischen Geheimhaltung*, (in German)
- [3] Giessing, S. and Repsilber, D. (2002), 'Tools and Strategies to Protect Multiple Tables with the GHQUAR Cell Suppression Engine', in *'Inference Control in Statistical Databases'* Domingo-Ferrer (Editor), Springer Lecture Notes in Computer Science Vol. 2316, to appear.

2.9 *Hitas*

HiTaS is a heuristic approach to cell suppression in hierarchical tables. Hierarchical tables are specially linked tables: at least one of the spanning variables exhibits a hierarchical structure, i.e., contains (many) sub-totals.

In Fiscetti and Salazar (1998) a theoretical framework is presented that should be able to deal with hierarchical and generally linked tables. In the sequel this will be called the mixed integer approach. In that framework, additional constraints to a linear programming problem are generated. The number of added constraints however, grows rapidly when dealing with hierarchical tables, since many dependencies exist between all possible (sub-)tables containing many (sub-)totals. The implemented heuristic approach (HiTaS) deals with a large set of (sub-)tables in a particular order. A non hierarchical table can be considered to be a hierarchical table with just one level. In that case, the approach reduces to the original mixed integer approach and hence provides the optimal solution. In case of a hierarchical table, the approach will provide a sub-optimal solution that minimises the information loss per sub-table, but not necessarily the global information loss of the complete set of hierarchically linked tables.

In the following section, a short description of the approach is given. For a more detailed description of the method, including some examples, see e.g., De Wolf (2002).

HiTaS deals with cell suppression in hierarchical tables using a top-down approach. The first step is to determine the primary unsafe cells in the base-table consisting of all the cells that appear when crossing the hierarchical spanning variables. This way all cells, representing a (sub-)total or not, are checked for primary suppression. Knowing all primary unsafe cells, the secondary cell suppressions have to be found in such a way, that each (sub-)table of the base-table is protected and that the different tables cannot be combined to undo the protection of any of the other (sub-)tables. The basic idea behind the top-down approach is to start with the highest levels of the variables and calculate the secondary suppressions for the resulting table. The suppressions in the interior of the protected table is then transported to the corresponding marginal cells of the tables that appear when crossing lower levels of the two variables. All marginal cells, both suppressed and not suppressed, are then 'fixed' in the calculation of the secondary suppressions of that lower level table, i.e., they are not allowed to be (secondarily) suppressed. This procedure is then repeated until the tables that are constructed by crossing the lowest levels of the spanning variables are dealt with.

A suppression pattern at a higher level only introduces restrictions on the marginal cells of lower level tables. Calculating secondary suppressions in the interior while keeping the marginal cells fixed, is then independent between the tables on that lower level, i.e., all these (sub-)tables can be dealt with independently of each other. Moreover, added primary suppressions in the interior of a lower level table are dealt with at that same level: secondary suppressions can only occur in the same interior, since the marginal cells are kept fixed.

However, when several empty cells are apparent in a low level table, it might be the case that no solution can be found if one is restricted to suppress interior cells only. Unfortunately, backtracking is then needed.

Obviously, all possible (sub)tables should be dealt with in a particular order, such that the marginal cells of the table under consideration have been protected as the interior of a previously considered table. To that end, certain groups of tables are formed in a specific way (see De Wolf (2002)). All tables within such a group are dealt separately, using the mixed integer approach.

The number of tables within a group is determined by the number of parent-categories the variables have one level up in the hierarchy. A parent-category is defined as a category that has one or more sub-categories. Note that the total number of (sub-)tables that have to be considered thus grows rapidly.

Singletons

Singleton cells should be treated with extra care. The single respondent in this cell could easily undo the protection if no extra measures were taken. The most dangerous situation is that there are only two singletons in a row, or one singleton and one other primary unsafe cell. These singletons could easily disclose the other cell.

In the current implementation we have made sure that at least two singletons in one row or column cannot disclose each other information. For this we will increase the protection margins of these singletons such that the margin of the largest is greater than the cell-value of the smallest.

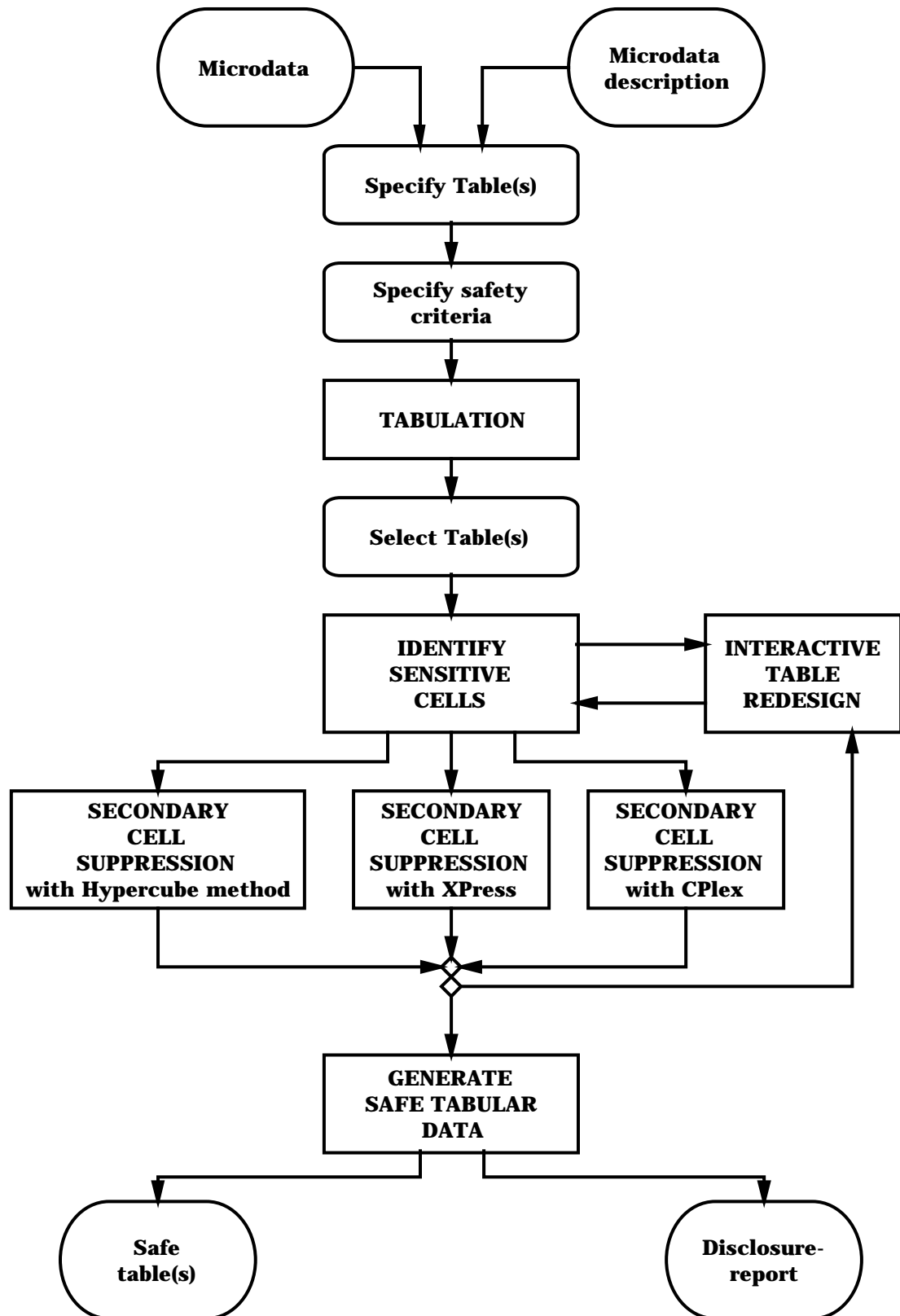
References on HITAS

Fischetti, M. and J.J. Salazar-González (1998). *Models and Algorithms for Optimizing Cell Suppression in Tabular Data with Linear Constraints*. Technical Paper, University of La Laguna, Tenerife.

P.P. de Wolf (2002). *HiTaS: a heuristic approach to cell suppression in hierarchical tables*. Proceedings of the AMRADS meeting in Luxembourg (2002).

Additional reading on the optimisation models can be found at the website (<http://webpages.ull.es/casc>)

2.10 Functional design of τ -ARGUS

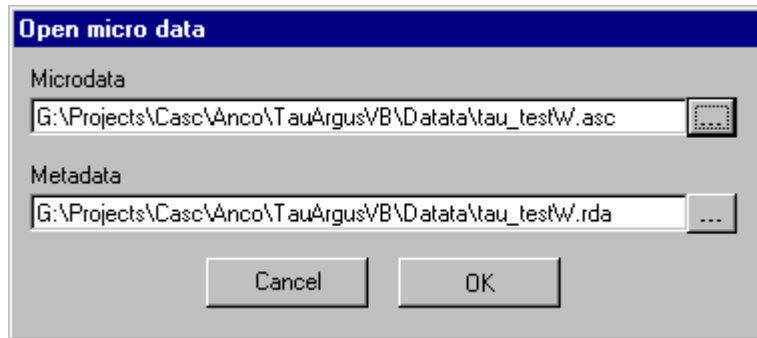


3. A tour of τ -ARGUS

This section will give the reader an introduction through all phases of the use of τ -ARGUS. Some Windows experience is assumed. In section 4 a more systematic description of the different parts of τ -ARGUS will be given.

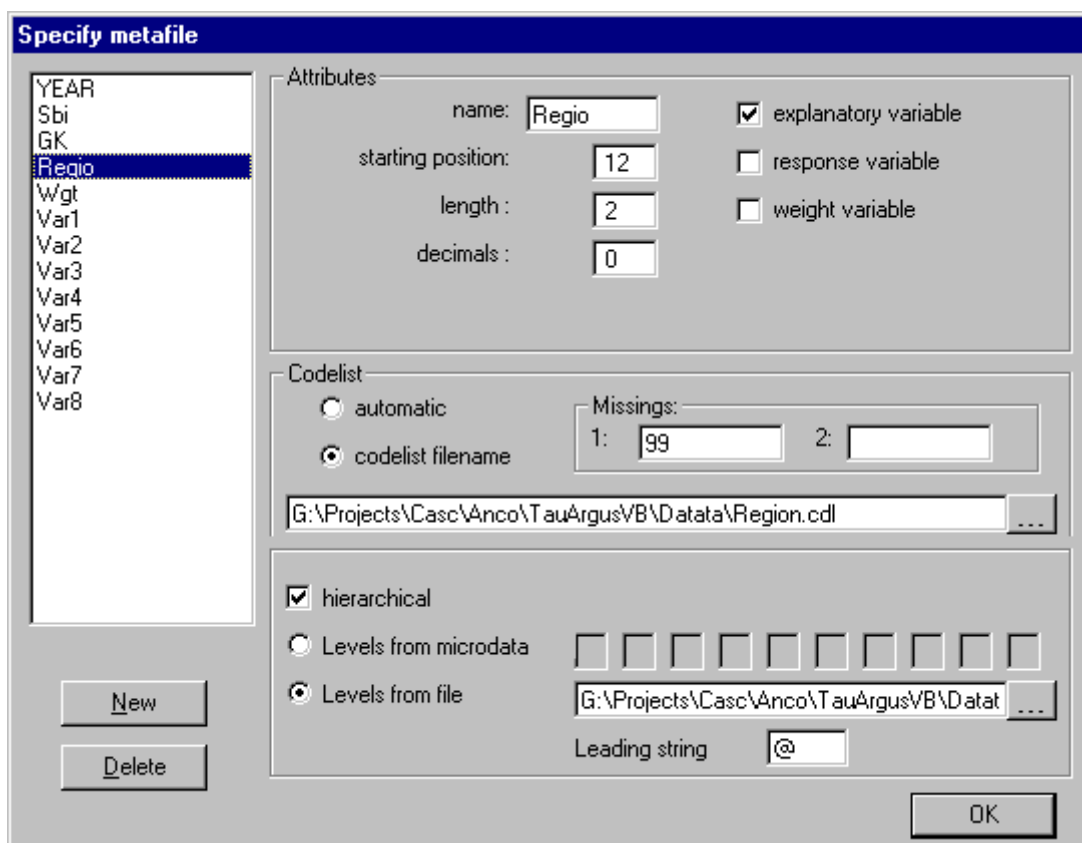
3.1 Preparation

To start the disclosure control with τ -ARGUS you need to have a microdata file and the metadata describing this microdata file. The microdata file must be a fixed format ASCII file. If you click (File|Open Microdata) you can specify the name of the microdata file and the name of the file containing the metadata.



The program assumes the extension .ASC for the datafile and .RDA for the metadata, but you can use your own extensions. The metadata is stored in a separate file. If the name of the metadata file is the same as the datafile, except for the extension, τ -ARGUS will fill in this file automatically. If no metadata file is specified, the program has the facility to let the user specify the metadata interactively via the menu option (Specify|Metafile). This is also the place to make changes to the metadata. In subsection 3.1.1 we will give a description of the metadata file for τ -ARGUS.

When you enter or change the metadata interactively using τ -ARGUS the option (Specify|Metafile) will bring you to this screen:



The left pane shows the names of the variables. Besides some information on the position of the variables you can indicate whether a variable can be used as explanatory variable in a table or as a response variable. It is also possible to indicate that a variable can be used as a weight variable. In a weighted table an adapted version of the dominance rule is applied to identify the primary unsafe cells.

In the bottom half of the window all kind of details on the codelist can be stored.

- The codelists: Always τ -ARGUS will explore the datafile itself and build the codelist of the explanatory variables. This is the automatic option, However additionally the user can specify a codelist file. This codelist will be used to present more meaningful labels attached to the codes in some of the screens of τ -ARGUS.
- Missing values. τ -ARGUS needs to know which missing values are attached to a codelist.
- Hierarchical codes. As τ -ARGUS has now the facilities to protect hierarchical tables, you can instruct τ -ARGUS here on the nature of these hierarchical codelists. There are basically two options.
 1. The hierarchy can be derived from the digits of the individual codes. Each digit denotes a level and if required some digits can be grouped together.
 2. A file containing the hierarchical structure is specified. In this file the level of the nesting is indicated by a special character/string. In the example above the @ had been selected.

3.1.1 The Metafile

The metafile describes the variables in the microdata file both the record layout and some additional information necessary to perform the SDC-process. Each variable is specified on one main line followed by one or more option lines.

1. The first line gives the name of the variable followed by the starting positions, the width of the field and one or two missing values.
2. The following lines specify specific characteristics of the variable:

• <RECODEABLE>	This variable can be recoded and used as an explanatory variable in
----------------	---

	a table
• <CODELIST>	This explanatory variable has a codelist. The name of the codelist file follows this keyword. Default extension .CDL
• <NUMERIC>	This variable can be used as cell-item.
• <DECIMALS>	The number of decimal position for this variable
• <WEIGHT>	This variable contains the weighting scheme
• <HIERARCHICAL>	This variable is hierarchical
• <HIERLEVELS>	The hierarchy is derived from the digits of the codes itself. The specification is followed by a list of integers denoting the width of each level. The sum of these integers should be the width of the total code
• <HIERCODELIST>	The name of the file describing the hierarchical structure. Default extension .HRC
• <HIERLEADSTRING>	The string/character that is used to indicate the depth of a code in the hierarchy

An example of a metadata file

```

YEAR 1 2 99
  <RECODEABLE>
Sbi 4 5 99999
  <RECODEABLE>
  <HIERARCHICAL>
  <HIERLEVELS> 3 1 1 0 0
GK 9 2 99
  <RECODEABLE>
Regio 12 2 99
  <RECODEABLE>
  <CODELIST> Region.cdl
  <HIERCODELIST> Region2.hrc
  <HIERLEADSTRING> @
  <HIERARCHICAL>
Wgt 14 4 9999
  <NUMERIC>
  <DECIMALS> 1
  <WEIGHT>
Var1 19 9 999999999
  <NUMERIC>
Var2 28 10 9999999999
  <NUMERIC>
  <DECIMALS> 2
.....
.....

```

An example of a codelist file (Region.CDL)

```

1,Groningen
2,Friesland
3,Drenthe
4,Overijssel
5,Flevoland
6,Gelderland
7,Utrecht
8,Noord-Holland
9,Zuid-Holland
10,Zeeland
11,Noord-Brabant
12,Limburg
Nr,North

```

```
Os, East
Ws, West
Zd, South
```

Example of a file with the hierarchical structure (Regio.HRC)

```
Nr
@ 1
@ 2
@ 3
Os
@ 4
@ 5
@ 6
@ 7
Ws
@ 8
@ 9
@10
Zd
@11
@12
```

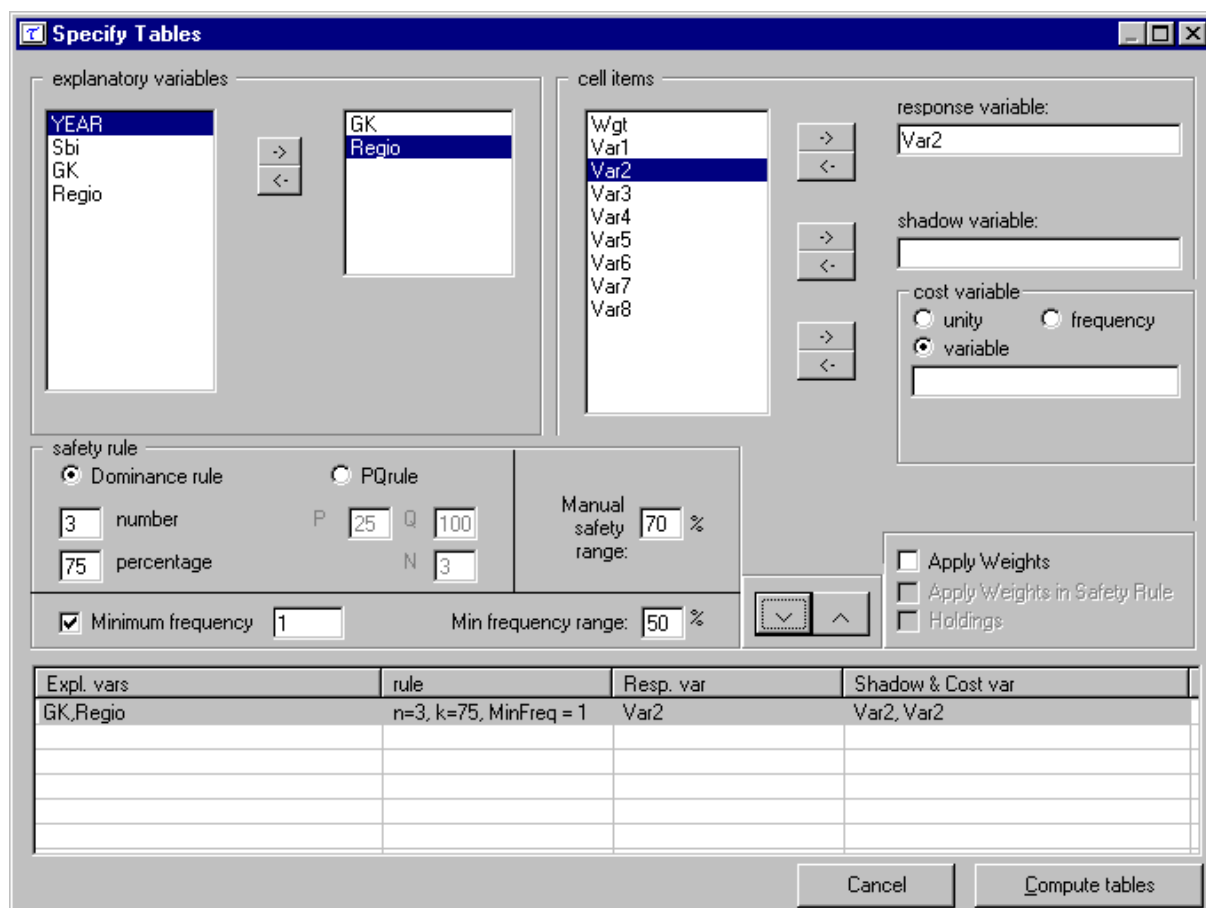
3.1.2 Specification.

When the metadata is ready, the user can specify the tables he wants to protect. Via Specify|Tables you come in a window to specify the tables. In the current version of τ -ARGUS you can specify more than one table, but each table is protected separately. In the upper part of the window you will find two panes, the one showing the explanatory variables and one showing the response variables. Up to four explanatory variables can be selected. The response variable is the variable that will be used to calculate the cell totals. The cost variable is used to calculate the minimum information loss; see section 2.5. This variable is not required as the default will be the response variable. As an alternative also the frequency of the cell can be used as a cost-function, or even a '1' for each cell. The frequency-option will minimise the number of contributors to be minimised, while the unity-option will simply minimise the number of suppressed cells.

The shadow variable is the variable that will be used to apply the dominance rule. In most cases this will be the response variable, but sometimes an other variable can be used. E.g. the turn-over (as a proxy for the size of the enterprises) can be a suitable variable to apply the dominance rule, although the table is constructed with an other variable.

At the bottom of the window the parameters for the dominance rule (see section 2.1) can be specified. Not only the parameters (N,p%) for the dominance rule can be specified, but also the minimum number of records contributing to a cell can be specified. Alternatively the prior-posterior rule (P-Q-rule) can be used to identify the primary unsafe cells.

When a cell is set manually unsafe (an option to discussed later), τ -ARGUS cannot calculate safety-ranges itself. So the user must supply a safety-percentage manually. Also when the minimum frequency rule is used, cells that violate this rule have no theoretical safety range. So also a user-specified range is needed here.



When the tables have been specified- more than one is possible, but we keep it simple in this introduction- you press the ‘Compute tables’ button. The result will be that τ -ARGUS will go twice through the datafile. In the first round the codelist of the explanatory variables are computed and in the second round the tables itself are computed.

Everything is now ready for the actual table protection.

3.2 The process of disclosure control

When the table(s) have been calculated, the main-window of τ -ARGUS will show again with an overview of all the unsafe cells per variable (over all the tables). If you had specified more than one table you have to choose a table with ‘Modify|Select table’. In this example you can go directly to ‘Modify|View table’.

The main menu for τ -ARGUS, showing the number of unsafe combinations per variable

Variable	dim 1	dim 2
GK	1	12
Regio	0	12

Code	Label	Freq	dim 1	dim 2
	Total	42713	1	0
Nr	North	11385	0	2
.1	Groningen	6112	0	2
.2	Friesland	3788	0	0
.3	Drenthe	1485	0	0
Os	East	10227	0	2
.4	Overijssel	538	0	2
.5	Flevoland	1597	0	0
.6	Gelderland	5446	0	2
.7	Utrecht	2646	0	0
Ws	West	10054	0	0
.8	Noord-Holla...	1182	0	0
.9	Zuid-Holland	8018	0	0
10	Zeeland	854	0	0
Zd	South	11047	0	1
11	Noord-Brab...	7511	0	1
12	Limburg	3536	0	0
99		0	0	0

3.2.1 Table protection

Via 'Modify/View table' you will come to the table-window, the heart of τ -ARGUS :

	tot	2	4	5	6	7	8	9	9
tot	16847261.84	20.00	25.00	2711808.00	2320534.00	2505042.58	2799074.26	6510758.00	
Nr	4373279.00	5.00	5.00	719049.00	659680.00	688962.00	756529.00	1549049.00	
1	1986129.00	5.00	5.00	398062.00	348039.00	354711.00	418778.00	466529.00	
2	1808861.00	-	-	223990.00	221332.00	241913.00	258233.00	863393.00	
3	578289.00	-	-	96997.00	90309.00	92338.00	79518.00	219127.00	
Os	3703896.00	15.00	5.00	642238.00	515003.00	534147.00	620392.00	1392096.00	
4	124336.00	5.00	-	36311.00	32132.00	25770.00	18150.00	11968.00	
5	526279.00	-	-	93589.00	94957.00	110930.00	81799.00	145004.00	
6	2234995.00	10.00	5.00	345803.00	251358.00	251188.00	303377.00	1083254.00	
7	818286.00	-	-	166535.00	136556.00	146259.00	217066.00	151870.00	
Ws	4576115.84	-	-	648972.00	543570.00	663896.58	775132.26	1944545.00	
8	485326.00	-	-	63767.00	75442.00	87305.00	59953.00	198859.00	
9	3664559.84	-	-	537911.00	430851.00	515019.58	643762.26	1537016.00	
10	426230.00	-	-	47294.00	37277.00	61572.00	71417.00	208670.00	
Zd	4193971.00	-	15.00	701549.00	602281.00	618037.00	647021.00	1625068.00	
11	2752743.00	-	15.00	488613.00	392395.00	363490.00	402925.00	1105305.00	
12	1441228.00	-	-	212936.00	209886.00	254547.00	244096.00	519763.00	
99	-	-	-	-	-	-	-	-	

From here all the other actions can be performed.

The options at the bottom of the table:

Via 'Change view' you can transpose the table. You simply indicate which variable will be the row-variable and which will be the column variable and the table will be transposed. If more than two explanatory variables had been selected the other variables will be in the layer and shown as combo-boxes on the top.

Table-summary will give you an overview of the number of safe and unsafe cells.

Explan. Var
GK
Regio

Status	Freq
Safe	102
Safe (Manual)	0
Unsafe	13
Unsafe (Manual)	0
Protected	0
Secondary	0
Secondary (Manual)	0
Empty	47

Respons Var:

Shadow Var:

Cost Var:

Not yet protected

OK

The checkbox 3-dig separator will optionally show this separator.

The checkbox output view will alternatively show the table in full or will all the primary and secondary cells replaced by 'X' as the table will eventually be published.

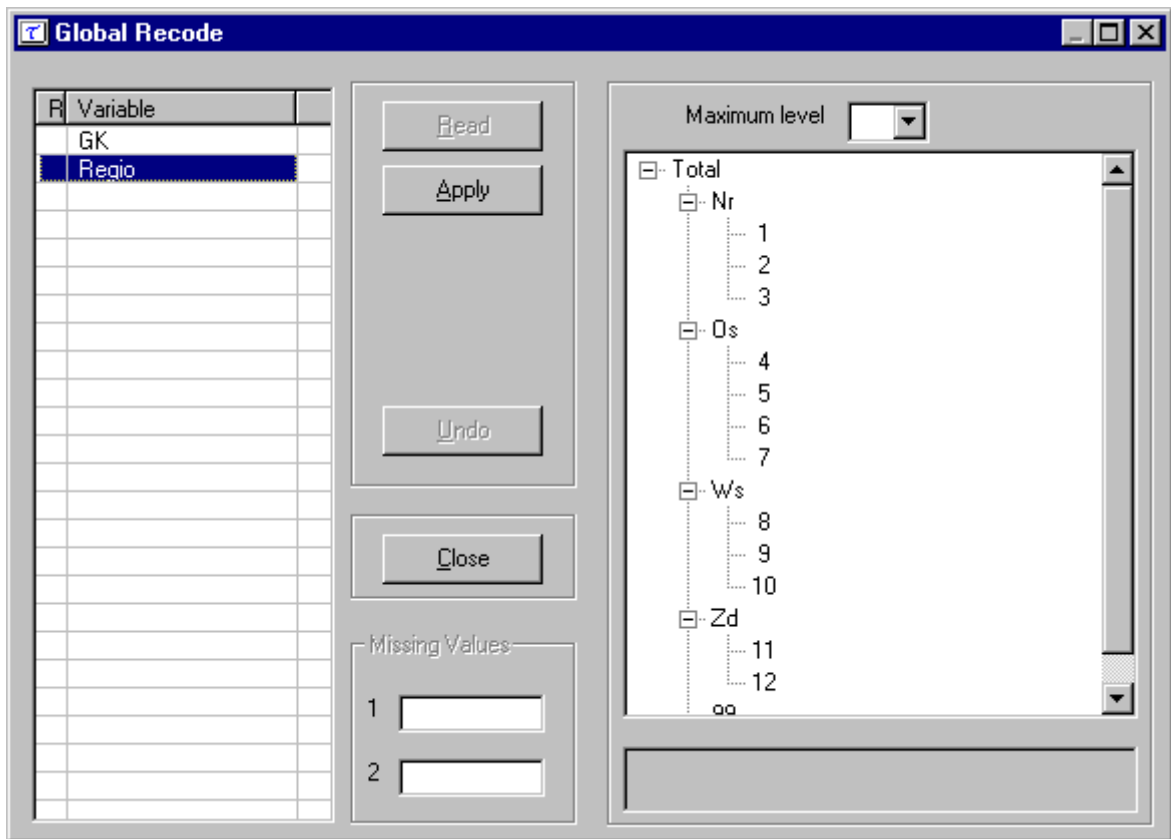
The **Cell-information pane** will show the details of the selected cell.

The **Change-status pane** will allow you to manually change the status of a cell. A safe cell can be made unsafe and vice versa, if you want to overrule the primary safety rule for some reason. E.g because the contributors to a cell have allowed you to publish their information. Also you could change the status from safe to protected. This will cause that a cell will never be selected as a secondary suppression.

Recoding

An important option of τ -ARGUS is the recoding of explanatory variables. Combining categories is an efficient way to protect cells as larger cells tend to be safer. Pressing the recode-button will bring you to this window:

Specifying and selecting recodings



In the above example we have selected the Region. This window will behave differently whether the variable selected is hierarchical or not. In the hierarchical case the codes are shown in a hierarchical tree. The standard windows facilities to manipulate trees can be applied here as well. Folding and unfolding of branches is showing or omitting codes from the table. In this version of τ -ARGUS we have chosen that this is the only allowable recoding of a hierarchical codelist.

Pressing the 'Apply' button will actually apply the selecting recoding. The undo-button is possible to go back to the original recoding scheme.

In the case of a non-hierarchical codelist the right hand pane will be an editbox. In this editbox a recoding scheme can be specified. The syntax is as follows:

A recode groups together categories as they appear in the original microdata and makes each group a new category with a new code. The syntax of the recode file is as follows: each line in the file corresponds with one new category. The code of the new category is placed before a colon (:). The old categories to be grouped into the new category are placed behind the colon. Single categories are separated by commas (,) and if a hyphen (-) is placed between two categories it refers to all subsequent categories between and including these two categories. If a hyphen is only placed before or after a category, this refers to all categories before or after and including this category respectively.

Example: The line "7:-4,6,8-10,13-" means that the categories 0, 1, 2, 3, 4, 6, 8, 9, 10, 13, 14, ... (if present) are recoded as 7.

Via the read-button you can read an existing recode scheme.

In the non-hierarchical case you could change the codes for missing values here, if you wish and indicate a new codelist.

The suppress-button.

This is the most important button. It will activate the modules for computing the necessary secondary suppressions. You have in principle three options here.

- Hypercube
- Opt/Xpress
- Opt/Cplex

The hypercube method (see section 2.7) will calculate a suppression pattern without a LP-optimisation module. So it can be used without any additional licence. The other two are two different implementation of HITAS, the search-algorithms based on JJ Salazar's optimal solutions, see section 2.9.

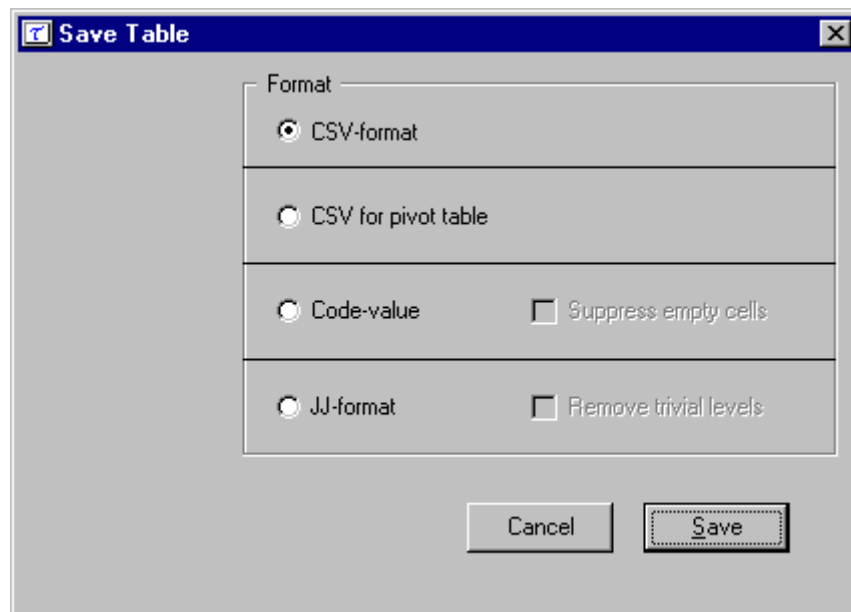
In either case you will in principle end with a safe table, indicated by the additional blue (secondary) cells.

You are now ready to store the table (with the 'write table') button. This is the same option as 'Output|Save table'

3.3 Saving the safe table

When the table is safe it can be written to the hard disk of the computer. You have three options:

1. CSV-format; a format that can easily be read by Excel.
2. CVS-format for pivot table. Nice for manipulating the table in Excel.
3. Code-Value



And of course a report file is written

4. Description of the Menu Items

In this section we will give a description of the program by menu-item. The information in this section is the same as the information shown when the help-facility of τ -ARGUS is invoked.

4.1 Main Window

Variable	dim 1	dim 2
GK	1	12
Regio	0	12

Code	Label	Freq	dim 1	dim 2
	Total	42713	1	0
Nr	North	11385	0	2
.1	Groningen	6112	0	2
.2	Friesland	3788	0	0
.3	Drenthe	1485	0	0
Os	East	10227	0	2
.4	Overijssel	538	0	2
.5	Flevoland	1597	0	0
.6	Gelderland	5446	0	2
.7	Utrecht	2646	0	0
Ws	West	10054	0	0
.8	Noord-Holla...	1182	0	0
.9	Zuid-Holland	8018	0	0
10	Zeeland	854	0	0
Zd	South	11047	0	1
11	Noord-Brab...	7511	0	1
12	Limburg	3536	0	0
99		0	0	0

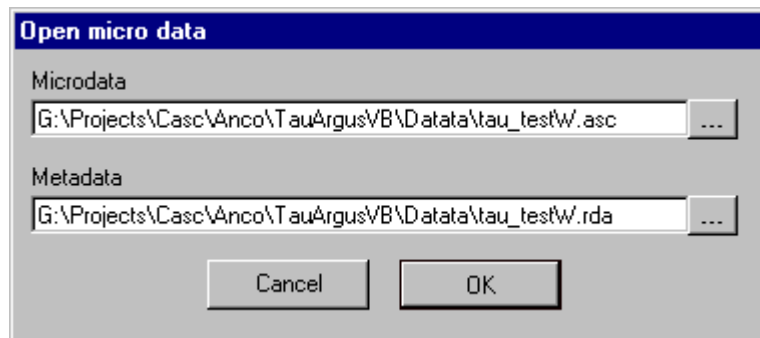
Overview of the menu-items

<u>F</u> ile	<u>S</u> pecify	<u>M</u> odify	<u>O</u> utput	<u>H</u> elp
<u>O</u> pen	<u>M</u> etafile	<u>S</u> elect Table	<u>S</u> ave table	<u>C</u> ontents
<u>E</u> xit	<u>T</u> ables	<u>V</u> iew Table	<u>V</u> iew Report	<u>O</u> ptions
				<u>A</u> bout


4.2 The File menu

4.2.1 File|Open Microdata

The File|Open microdata menu allows you to specify the Microdata file and the meta data file



In this dialog box you can select the microdata-file and the corresponding metafile.

- By default the microdata-file has extension .asc and the metafile .rda.
- When you click on  you get an open file dialog box. In this box you can search for the files you want to use. You can choose other file-types when you click on the file-types listbox. When you have selected the microdata-file a suggestion for the metafile (with the same name but with the extension .rda) is given but only when this file exists.
- Before you click **OK** you must have filled in the name of the microdata-file.

4.2.2 File|Exit

Terminates the τ -ARGUS-session.

4.3 The Specify menu

4.3.1 Specify|Metafile

In this dialog box all attributes of the variables can be specified. In section 3.1.1 we already have explained the layout of this file.

If under File|Open Microdata a .rda file has been specified, this dialog box shows the contents of this file. If no .rda file has been specified the information can be specified in this dialog box after pushing the New button. As default "newvar" is substituted. Apart from defining a new variable, an existing one can be modified or deleted.

The following attributes can be specified:

- name of the variable
- its first position in the data file

- its length and the number of decimals.

Furthermore the kind of variable can be specified: explanatory variable, response variable or weight variable. An explanatory variable can be used as a spanning variable in the row or the column of the table, a response (numerical) variable can be used as cell-item. A weight variable specifies the weight of the record and is based on the (post)sampling design used.

Codelist

As this version of τ -ARGUS is the first version that can handle hierarchical codelists, this information should be made known to τ -ARGUS.

In the non hierarchical case τ -ARGUS will always explore the datafile itself and make the codelist. This is called the automatic option. Additionally the user can specify a codelist, but that is only a list containing the labels attached to the codes. These labels are only used to enhance the information by τ -ARGUS on the screen, but τ -ARGUS will work only with the codes, that it has found when it explored the datafile

In the hierarchical case there are two options.

1. The hierarchy is derived from the digits in the codes. Each digit of the code denotes a level in the hierarchy. This is the traditional system for often used code lists like industry code (NACE) etc. Additionally in τ -ARGUS you can specify that sometimes two or more digits together denote one level.
2. If the coding system used does not contain itself the information about the hierarchy this information should be supplied to τ -ARGUS in an extra file. The layout of this file has been explained already in section 3.1.1. In this file the tree structure of the hierarchy is stored. A special character is used to indicate the depth in the hierarchy of a code. In this example an “@” has been chosen.

Additional for each code the missing values (at least one) should be specified. In many surveys two missing values are used e.g. one for *don't know* and one for *refusal*.

4.3.2 Specify|Tables

In this dialog box you can specify the tables you want to protect. In one run of τ -ARGUS more than one table can be specified, but the tables will be protected separately.

Also you have to specify the parameters for the dominance rule and the minimum of records in a cell. At this moment τ -ARGUS only allows for 4-dimensional tables, but to the capacities of the LP-solver used (Xpress or CPLEX) and the complexity of the optimisations involved these 4-dim tables can only be protected by the hypercube method, see section 2.7

Expl. vars	rule	Resp. var	Shadow & Cost var

The explanatory variables

On the left is the listbox with the explanatory variables

When you click on '>' or '<' you transport the selected variable to the next box. From the left box with explanatory variables you can select the variables that will be used in the row or the column of the table.

The response variable

From the list of response variables you can select a variable as response variable (the cell-item). This is the variable for which the table to be protected is calculated.

The shadow variable

The shadow variable is the variable that is used to apply the safety rule. By default this is the response variable, but it is possible to select another variable. The safety rules are build on the principle of the characteristics of the largest contributors to a cell. If an other variable than the response variable is a better indicator for the size of a company this variable can be used here.

The cost variable

This variable describes the cost of each cell. These costs are minimised when the secondary suppressed cells are calculated. By default this is the response variable but another choice is possible. It is also possible to use the frequency of the cells as a cost-function, This will minimise the number of records contributing to the cells to be suppressed. A third option is that the number of cells to be suppressed in minimised, irrespective of the size of their contributions.

Weight

If the data file has a sample weight, specified in the meta data, the table can be computed taking this weights into account. For this purpose the dominance rule has been extended.

The safety rule

The concept of safety rules is explained in section 2.1 On the left side of the window the type of rule can be selected and the value of the parameters. Additional the minimum number of contributors can be chosen.

For the dominance rule and the pq-rule safety ranges can be derived automatically. The theory gives formulas for the upper limit only, but for the lower limit we have chosen a symmetric range. However the theory does not provide any safety range for the frequency rule and of course not for the cells that have been made unsafe by manual intervention of the user. So in these two cases the user must provide a safety-range percentage.

When you have filled in everything you click ‘√’ to transport all the specified parameters to the listwindow on the bottom. You can specify as many tables as you want, but as the size of the memory of a computer is still restricted you should not overdo it. If you want to modify a already made table you press the ‘^’ button.

Pressing the ‘Compute tables’ button will invoke τ -ARGUS to actually compute the tables requested and you are ready to start the process of disclosure control. τ -ARGUS will come back with the main window showing you the number of unsafe cells per variable per dimension.

4.4 The Modify menu

4.4.1 Modify|Select Table

In this dialog box you can select the table you want to see. If you have specified only one table, this table will be selected automatically and this option cannot be accessed..

4.4.2 Modify|View Table

This window shows the table you have selected with Modify|View Table. On the left side the table itself is shown in a spreadsheet view. Safe cells are black, unsafe cells are red, secondary suppressed cells are blue and empty cells have a hyphen (-). The two check-boxes on the left-bottom give you some control over the layout.

- The 3-digit separator will show the cell-values with this separator. The separator is chosen according to the general Windows settings.
- The Output view is meant to show the table, where all the suppressed cells are replaced by an 'X'; this is how the safe table will be published.

When you click on a cell, information about this cell is visible in Cell Information pane. You can see the following information:

1. The cell-value
2. The cell status
3. The number of contributors to a cell

4. The largest contributors.

Status is the status of the cell, this can be:

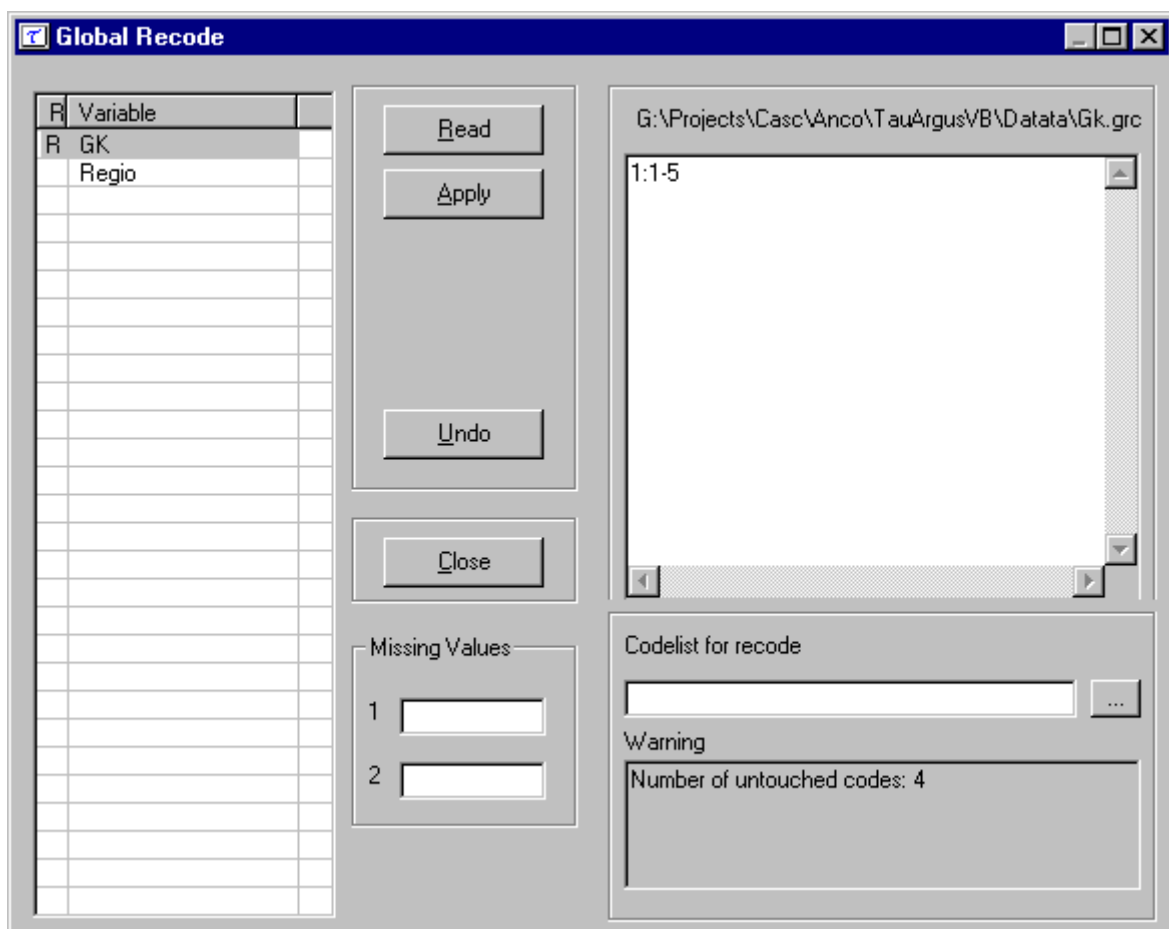
- Safe: Does not violate the safety rule
- Safe (from manual): manually made safe during this session
- Unsafe: According to the safety rule
- Unsafe (from manual): manually made unsafe during this session
- Suppressed: Made unsafe by the secondary cell suppression
- Protected: Cannot be selected as a candidate for secondary cell suppression
- Zero: Value is zero and cannot be suppressed.
- Empty: No records contributed to this cell and the cell cannot be suppressed.

The second pane on the right will allow you to change the cell-status. Only a few (logical) transitions are allowed. Changing the cell status can be useful if an unsafe cell may be published because of external reasons (permissions to publish). Changing to protect will prevent the system to select this cell as a secondary suppression.

The recode button will bring you to the recoding system. Recoding is a very powerful method of protecting a table. Collapsed cells tend to have more contributors and therefore tend to be much safer.

4.4.2.1 Recoding a non-hierarchical variable

There is a big difference in recoding a hierarchical variable compared to a non-hierarchical variable.



In the non-hierarchical case you can specify a global recoding manually. Either you enter the recoding described below manually or you read it from a file. The default extension for this file is .GRC.

There are some rules how you have to specify a recode scheme. All the codelists are treated as alphanumeric codes. This means that the codelists are not restricted to numerical codes only. However this implies that the codes '01' and '1' are considered different codes and also 'aaa' and 'AAA' are different. In a recoding scheme you can specify individual codes separated by a comma (,) or ranges of codes separated by a hyphen (-). The range is determined by treating the codes as strings and using the standard string comparison. E.g. '0111' < '11' as the '0' precedes the '1' and 'ZZ' < 'a' as the uppercase 'Z' precedes the lowercase 'a'. Special attention should be paid when a range is given without a left or right value. This means every code less or greater than the given code. In the first example the new category 1 will contain all the codes less than or equal to 49 and code 4 will contain everything larger than or equal to 150.

Example:

for a variable with the categories 1,...,182 a possible recode is then:

```
1: - 49
2: 50 - 99
3: 100 - 149
4: 150 -
```

for a variable with the categories 01 till 10 a possible recode is:

```
1: 01 , 02
2: 03 , 04
3: 05 - 07
4: 08 , 09 , 10
```

Don't forget the colon (:) if you forget it the recode will not work.

The recode 3: 05-07 is the same as 3: 05,06,07 you can choose what you like best.

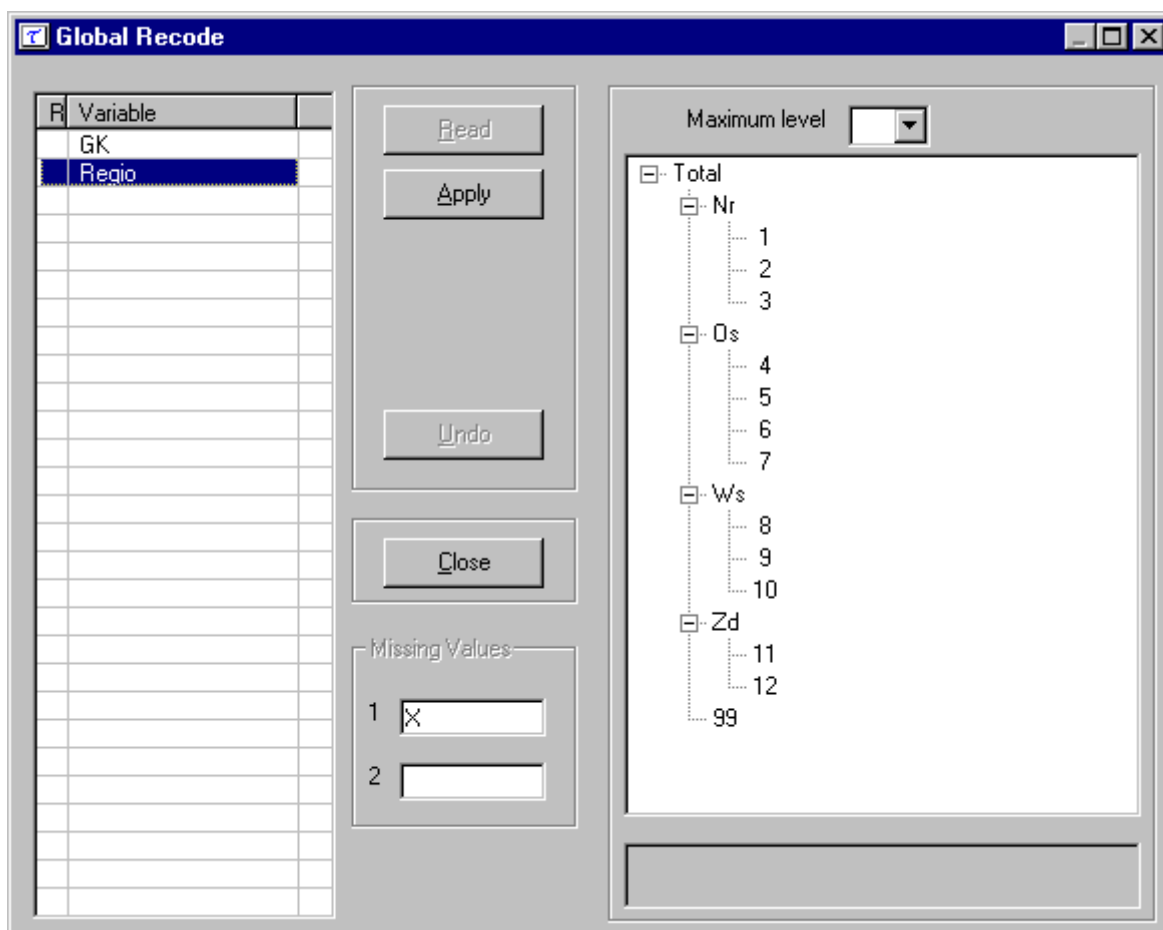
Additional you can change the coding for the missing values by entering these codes in the relevant textboxes. And you can specify the name of a new codelist with the labels for the new coding scheme.

Pressing the apply button will actually restructure your table. And if required you can always undo a recoding.

If you apply a recoding τ -ARGUS will present you with the results. This can be that certain codes could not be found or that you did not follow the above described syntax. In that case an error message will be shown. Alternatively a warning could be issued. E.g. if you did not recode all original codes, τ -ARGUS will inform you. But it can be your purpose and there is no objection to it. In the example above τ -ARGUS informs you that 4 codes have not been changed.

4.4.2.2 Recoding a hierarchical variable

In the hierarchical case the code scheme is typically a tree. To global recode a hierarchical variable means that you manipulate a tree. The standard Windows tree view is used to present a hierarchical code.



You can fold and unfold certain parts of a tree with the standard Windows actions.

The combo box at the top of the screen offers the opportunity to fold and unfold the tree to a certain level.

Additional you can change the coding for the missing values by entering these codes in the relevant textboxes.

Pressing the apply button will actually restructure your table. And if required you can always undo a recoding.

Change View

When you click on Change View this dialog box pops up. You can specify which variable you want in the row and the column. In the two dimensional case you can only transpose the table. In the higher dimensional case the remaining variables will be in the layer. For these layer variables a combo-box will appear at the top of the table, where you can select a code. This will show the corresponding slice of the table.

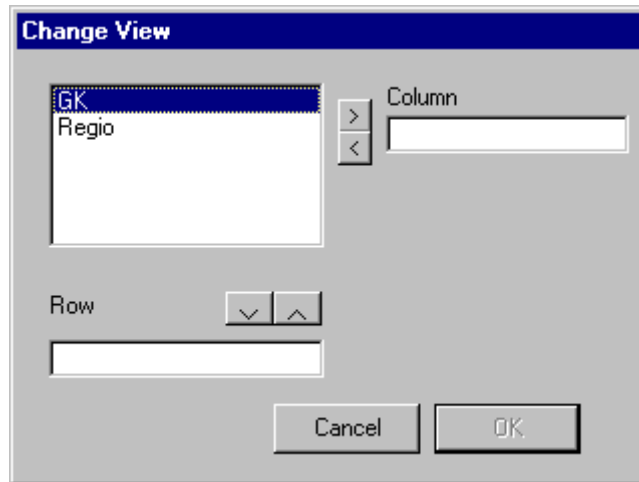
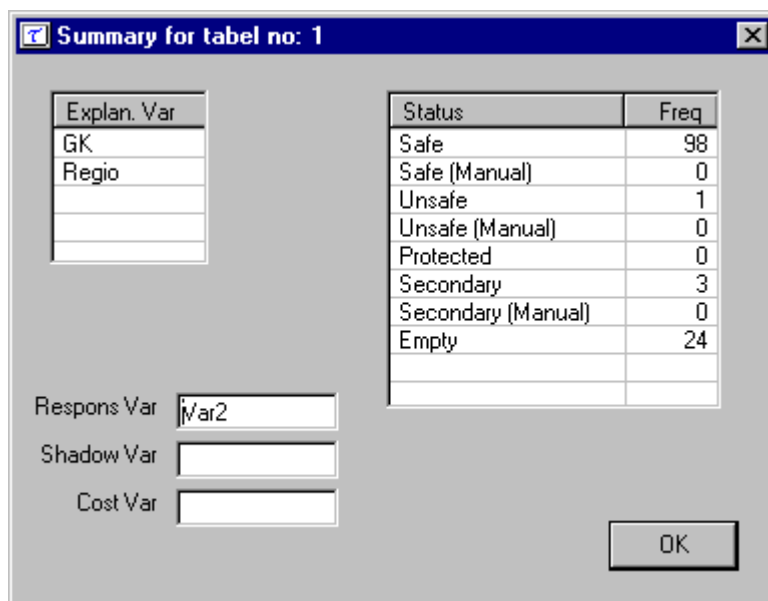


Table summary



The table summary will give an overview of the number of cells according to their status.

Secondary suppressions

With suppress you can protect your table by adding additional cells to be suppressed. This is necessary to make a safe table. In this version of τ -ARGUS you can choose between the hypercube method or the optimal solutions. In this version we did not yet implement the full optimal search-algorithms fro hierarchical tables, but have implemented the HITAS-approach (See section 2.9). This method will break the hierarchical table down to several non-hierarchical tables, protect them and compose a protected table from the smaller pieces. As this method uses the optimisation routines, an LP-solver is required. Either Xpress or CPLEX is required. It is the responsibility of the users of τ -ARGUS to apply for a licence of one of these commercial packages themselves. Information on obtaining one of these licences will be found in a read.me file that will be supplied with the software.

Just select one of the options and press the 'suppress'-button. τ -ARGUS will start working for you and finally it will show you a protected table. The secondary suppressed cells will be shown in blue. If you had selected the hypercube method τ -ARGUS will ask you for an adjustment for the ratio-parameter. Although τ -ARGUS will impute the theoretical value, the protection will sometime fail and in the case a lower value of the ratio parameter should be selected. More information on the hypercube method can be found in section 2.7

When you are satisfied with the table you can store the table. Press the write-table button. This is virtually the same button as via the menu Output|Save table (section 4.5.1)

4.5 The Output menu

4.5.1 Output|Save Table

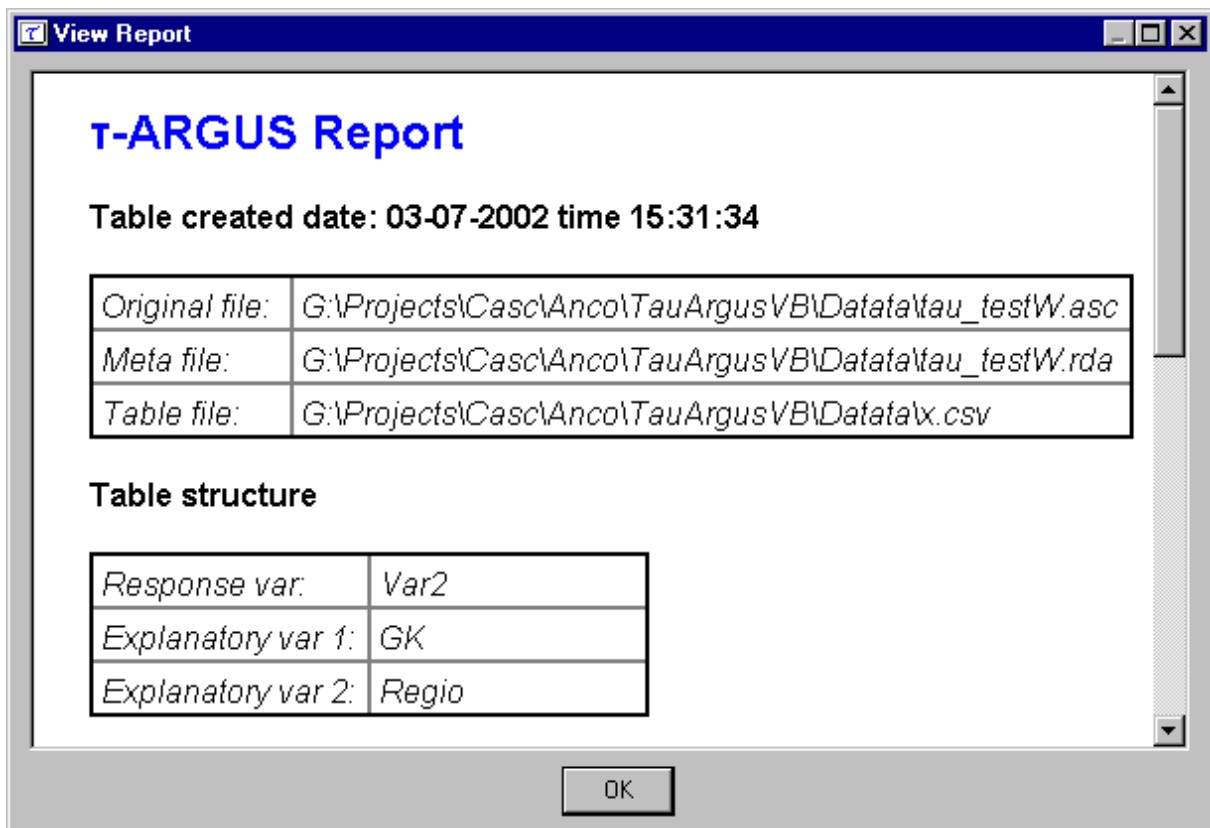
Basically you have three options of storing the tables

1. As a CSV-file. This Comma separated file can easily be read into Excel. Often it is better to click on this file in the windows explorer, than reading it into Excel by using the File|open option of Excel.
2. A CSV-file for a pivot table. This offers you the opportunity to make use of the facilities of pivot table in Excel.
3. A file in the format code value separated by comma's

Finally a report will be generated. This report will be shown, when the table has been written. As this is an HTML-file it can be viewed easily later.

4.5.2 Output|Report

Views the report file which has been generated with Output|Save Table



4.6 The Help menu

4.6.1 Help|Contents

Shows the contents page of the help file.
This program has context-sensitive help.

4.6.2 Help|Options

There is one option that can be set here. If in unforeseen situations the HITAS/Salazar optimal solutions might break down, a large debug-file (JJUIT.DAT) will be written in the TEMP-directory. In normal situations only the information of the last sub-table will be stored, but when the debug-option is activated, the full JJUIT.DAT will be written. This information might useful if you contact us.

4.6.3 Help|About

Shows the about box.