CASC PROJECT

Computational Aspects of Statistical Confidentiality

1 October 2001

Software concept and design specification document for τ-ARGUS

Sarah Giessing
Statistisches Bundesamt

deliverable 3-D2

PART 1: New facilities and data structures for τ-ARGUS

Draft proposal by StBA

Working document

Abstract In the course of the EU funded project CASC, the software τ-ARGUS shall be extended to become a generally applicable standard tool for tabular data protection. The required modifications will affect the facilities provided for (residual) disclosure risk statement, the data structure and the user interface. Methods will have to be implemented for protection of complex hierarchical tables and for table-to-table protection, especially in the context of public use data-base query systems. Extensions will be made concerning secondary cell suppression methodology provided by the package. The package will interface in particular with the GHQUAR hypercube algorithm. Finally, table perturbation tools will be added.
Keywords CASC project, τ-ARGUS, tabular data protection, hierarchical tables, table-to-table protection, secondary cell suppression

1. Introduction

It will be the objective of this document to explain the need for new facilities and new data-structures for τ-ARGUS. The author is member of the CASC steering committee. It is her responsibility in the project team to monitor research and development of supplementary methodology for tabular data protection in τ-ARGUS.

Concerning tabular data protection, work within the CASC project will be carried out by the Statistical Institutes of the Netherlands, Germany and Nordrhein-Westfalen/Germany (CBS, StBA, LDS NRW), and teams at the Universitat La Laguna (ULL), Technische Universität Ilmenau (TUI), and Universitat Politècnica de Catalunya (UPC) . CBS will perform the implementation in τ-ARGUS, ULL will provide optimal search algorithms, StBa will suggest methodological strategies, LDS NRW will supply the GHQUAR hypercube algorithm, UPC will provide an alternative optimisation algorithm and it will be the main task of TUI to develop algorithms for particular table perturbation techniques. StBa has a co-ordinating role in the research and CBS for the software development and research regarding linear programming.

2. New facilities addressing disclosure risk specification

Unless one does not disseminate any data at all, there may always remain some residual disclosure risk, even in protected data. In advance of running a disclosure control procedure, a user will have to state to the system how much of certain disclosure risks would be acceptable to him.

2.1 Primary Disclosure Risk Assessment

The first step in a disclosure control procedure for table is always to assess the disclosure risk that would be connected to the release of each cell within the table. This test is usually done by applying certain sensitivity rules to the data. A cell, that is sensitive according to the sensitivity rule employed, would not be published, e.g. ‘suppressed’. Other cells (so called ‘secondary’ or ‘complementary’ suppressions) must be suppressed along with these so called ‘primary suppressions’, in order to prevent the possibility, that users of the published table would be able to recalculate the primary suppressions exactly, or to derive too narrow an estimate by making use of the linear relations between published and suppressed cells of the table.

Currently, τ-ARGUS offers to employ a concentration rule (e.g. (n,k)-dominance rule) in combination with a minimum number of respondents rule, with the parameters n and k of the dominance-rule and the minimum number of respondents left to be chosen by the user.
These rules are special cases only (although the best-known and most commonly applied ones) of the more general class of 'upper linear sensitivity measures' (c.f. e.g. [1], or [11]). The new version of τ-ARGUS should allow to combine several upper linear sensitivity measures. It should be offered particularly to apply a p%-rule or a (p,q)-rule.

2.2 The Protection Interval

Users of a published table by making use of the linear relations between published and suppressed cells of the table are able to derive upper and lower bounds for the true value of any suppressed entry. The interval given by these bounds is usually called '*suppression interval*'. For proper selection of complementary suppressions, the disseminator should determine safety bounds for any primary suppression. We call the interval between the upper and lower safety bounds '*protection interval*'. The suppression procedure will ensure, that no suppression pattern will be considered to be feasible, unless the corresponding suppression interval encloses the protection interval for any sensitive cell.
In the current version of τ-ARGUS, the user is requested to specify bounds for the protection interval. However, it is quite essential for a suppression procedure that these bounds are determined properly. Otherwise, it may cause either a risk of disclosure, or over-suppression.
When for primary confidentiality a concentration rule, such as an (n,k)-dominance-rule or p%-rule has been employed, there would be an unacceptable risk of disclosure according to this rule, when the upper bound of the suppression interval does not exceed the true value of the sensitive cell sufficiently: If the distance between upper bound and true value is below a certain minimum size, then this upper bound could be used to derive an estimate for single contributions to the sensitive cell, which is too close according to the sensitivity criterion employed. Formulas for (upper) bounds for the protection interval meeting this criterion, can easily be obtained making use of the results of [1] and are given in the appendix for the most common sensitivity rules. The new version of τ-ARGUS should offer a default option computing the protection interval according to these formulas.

Unfortunately, in certain cases, even when the protection interval has been determined properly as described above, certain risks of residual disclosure would still remain unattended to. When complementary suppressions are sensitive as well, or when cells, which are components of the same (sub)total cell, share common respondents, as will be illustrated in 1.3 below, then the suppression pattern should also ensure, that the upper bound for the true value of the (suppressed) combined cell would not disclose any respondents (combined) contribution. This cannot be achieved through proper definition of a protection interval for the sensitive cell alone, by making e.g. sure, that a complementary suppression satisfies a certain minimum size condition.

The problem should be addressed and solutions be implemented, such as e.g. suggested in [7] or [10].

2.3 The problem of common respondents

Statisticians often construct the same table for different response variables. Sometimes there is an additive relation present in a set of response variables, e.g. one of the response variables will actually be the sum of the others in the set. An example for this kind of interrelationship is a relation such as "total investment = investment in building + investment in ground + investment in technical equipment + other investment".
A common table protection technique used often in cases such as this, is to do the table protection only for one of the response variables (the "total investment" for instance), and then suppress in this example those entries in the "building", "ground", "technical equipment" and "other investment" tables corresponding to suppressed entries in the "total investment" table. Though this approach is certainly valuable in reducing the effort for data protection, it should better not be used, when the information given by any of the other response variables has to be considered both, sensitive and identifying. Identifying in the sense, that it can be assumed, that attackers are not only able to identify those respondents with extraordinarily large responses to the 'overall' variable (e.g. the total investment), but might as well be able to guess large respondents to the other variables. For our

investment example, this might happen to be the case. If for example the table presents low level aggregate data, and one of the respondents has had an expensive new building, while the other respondents did not have considerable costs for building, then many of those respondents may know, who has a very large share in their common published value for building investment, and be able to disclose this single contribution. So, this cell should better be suppressed, even if the 'overall' investment may turn out to be safe. In this case, all the "investment" tables should be protected together as a single table, with the relation between the different investment categories as one dimension of this table.
Tables constructed in this fashion have a typical property: cells, which are components of the same (sub)total cell, share common respondents. If the combined contribution of a respondent to a union of these cells has to be considered as confidential information, such as, say, the "investment in building and ground", then, as stated in 1.2 above, the suppression pattern should also ensure, that the upper bound for the true value of the (suppressed) combined cell would not disclose any respondents combined contribution.

3. New data structures

In the current version, τ-ARGUS cannot handle tables with hierarchical substructure, nor tables with a decomposition structure of the response variable, as described in sec. 3.3 above. There is no option for table-to-table protection of linked tables, and if, due to a decentralised organisation structure like within in the European or the German statistical system, a potential user of the software is unable of providing the micro-data set on which the table he wants to protect is based, he cannot use the system. All these facilities will be offered by new versions, which will of course require modification and new concepts in the data structures.
New data structures need to be designed for tabular data input, modifications will be made in the structure for the microdata input, and the design of files containing meta-information, codelists, and other structural information on the tables has to be modified or newly invented.

3.1 The users perspective: what to expect regarding new interface options of τ-ARGUS

This section will briefly illustrate alternatives for the kind of user input to be supplied for specification of hierarchical tables, and control of table-to-table protection procedures.

3.1.1 Definition of hierarchical tables

For hierarchical tables, the user has to supply information on the structure of the table to be protected. Naturally, a variety of options can be imagined of how to provide this structural information. For users of the tabular data input option, e.g. users supplying tabular data, one possibility would be to extract this information from the data-file: In that case, the tabular-data input file for a hierarchical table should specify cells using hierarchical codes for any hierarchically structured variable. The software would then for each dimension of the table create a list of codes as appearing in the data file. This would of course be not an option in case the user supplies micro-data. A simple option here would be to ask the user to supply a list of hierarchical codes for each dimension of the table. Of course this list must properly suit the codes for this dimension as supplied by the micro-data file. The software would check whether certain conditions in this respect are fulfilled.
Many users may be quite happy with this kind of input information. Hierarchical codelists will be readily available for the most detailed variables, such as for instance NACE or NUTS for the variables 'industry' or 'region'. When no such codelist is already available for a variable (typically variables such as 'size class', 'year of foundation', etc., the effort for creating the list might be considered rather low, as these variables normally tend be given simple hierarchical structures (if any), with rarely more than 20 categories. So, designed along these lines, the software would be usable, although not very flexible, still leaving a rather high organisational burden regarding data preparation, especially for

non-standard applications, or when the input-data in some way may not instantly meet the requirements.
Considering, that we claim to design a standard tool, to be used in so many different environments as there are within the European Statistical System at least, we might want to build a more flexible, user-friendly software. Such a software would offer tools to assist and to guide a user in the design of a hierarchical structure for a variable, and the modification of a given structure (for instance, if the table should not display data for all NACE codes, but only part of them), while preventing him from misspecifications, that might cause disclosure risk, or breakdown of the secondary cell suppression procedure. Tools of this kind will be essentially useful for users attempting to optimise the design of tables by 'playing' with the data, redesigning the table over and over again, dropping sub-totals or introducing new ones, and so forth, until finally ending up with a favourable table and an acceptable suppression pattern. This may exceed the needs of 'new' users with little experience in automated tabular data protection, however with growing expertise, especially when the task is, not simply to protect one single table, but multiple tables, maybe to the degree even of providing to some extent safe input for public use statistical data base query systems, it is expected that users will learn to appreciate this flexibility.
Challenges for us will be, to meet existing table specification standards, and to find a good balance between user-friendliness and not wasting software development capacity by 'reinventing the wheel', as of course there is already software available to handle the management of variable codes.

3.1.2 Specification of table-to-table and data base protection procedures

In comparison to the effort required for creation of comfortable tools for specification of hierarchical tables as illustrated above, the effort connected to software development needed to create user input facilities to specify a table-to-table protection procedure is rather low. (This remark does however ignore the fact, that comfortable tools for specification of hierarchical tables, such as discussed above, with a high software development burden connected to them, become truly useful chiefly in the context of table-to-table protection.)
For performing a table-to-table protection run, the user will have to create his tables, and to define them to the system just as when protecting single tables only. Then he will list the tables for the run, and is done.
Similar comments regarding the effort required for their creation can be made as well in the context of data base protection facilities. Options will be created in order to specify a set of (protected or unprotected) tables to be pooled in a single file. This pool file will contain one record only for each cell within the set of tables, particularly for cells appearing in more than one table. There will be pool meta files, giving historical information, e.g. showing which tables are already contained in the pool, which of them have already been protected, and will for protected tables provide information on the run-parameters used and log-files created when protecting them. Finally some simple options shall be provided to extract data from the pool file into ready-made tables.
Though this sounds quite simple, it should again be noted, that, especially in the context of data base protection, other facilities, which otherwise might not really be needed, will be required for support, e.g. as to improve the performance of such a procedure. We will denote them as:

3.1.3 Preference facilities

'Preference facilities' are supposed to make the system prefer or even force certain cells to remain unsuppressed, or on the contrary, to make certain cells be used as complements first. There will be options provided to define 'preferences' in automated fashion, like a switch to be put on or off for certain classes of cells, such as e.g. sub-total cells, particular cells of overlapping parts of tables, cells used as secondary suppressions in a previous period for tables presenting results of periodical surveys, etc. . On the other hand, there may options be offered to define 'preferences' for user-defined sets of cells, where in the extreme a set may even contain a single cell only. Those cells may be specified as (sub)tables, to be stated as cross-combinations of subsets of the sets of codes used to define the original tables. There should of course then facilities be provided to select subsets from a code list.
We certainly will have to supply a lot of user guidance and default options for 'beginners' and less experienced users. If it should turn out in the course of the CASC project, that the project capacity

(chiefly: the capacity available for software development) does not suffice to implement all the user-friendly facilities that may have proven to be useful, then we will at least try to make the design of the system open enough as to allow for the experienced user by supplying his own procedures (outside the system) or by manual intervention, to run the suppression procedure according to his individual needs.

3.2 The developers perspective: methodological strategies for the extension of τ-ARGUS

This section will outline strategies how to apply a suppression algorithm for unstructured tables to hierarchical tables, how to implement table-to-table protection, and how to extend table-to-table protection efficiently in the context of data base query systems.

3.2.1 Strategies for the protection of hierarchical tables

Single table approach

From the pure methodological point of view, it is actually the best strategy for protection of hierarchical tables, to not at all treat it as hierarchical table, but to turn it into one single(!) non-hierarchical table in advance. This is always possible and would be relatively simple to implement. In fact, this would be the only method to protect such a table properly, avoiding a particular kind of disclosure risk, which will be illustrated farther below. The challenge of this strategy would be to speed up the suppression algorithm for single, unstructured tables sufficiently, as to be powerful enough for application to real-life sized tables. It should be noted in this context, that in a non-hierarchical representation, a hierarchical table is much, much larger than in the hierarchical representation. Considering that the input for the suppression algorithm based on linear programming methodology is mainly a set of equations, that might (for simplicity) be regarded as describing linear relations between (potentially) suppressed and unsuppressed cells of the table, one will of course at a certain stage of the cell-suppression process drop from the set of equations resulting from the non-hierarchical representation all the non-valid ones, removing e.g. identity equations and equations appearing more than once (to the extent of not even creating them at all).

The secondary cell suppression problem is known to be computationally hard. The number of computations required for solving the secondary cell suppression problem stated as Integer Linear Programming (ILP) problem grows exponentially with the size of the table. It is therefore clear in advance, that it will not be an option to protect extremely large tables of several hundred thousand cells or more using linear programming methodology within such a single table approach.

'Backtracking' strategies

A common approach is, to split the table into sub-tables and protect the sub-tables separately. Doing so, one must of course take into account that these sub-tables of the same table do have cells in common. Otherwise it might happen, that the same cell is suppressed in one sub-table, because it is used as secondary suppression, while within another table it remains unsuppressed. A user comparing the two sub-tables would then be able to disclose confidential cells in the first table. A common method of complying with this is to note any complementary suppression belonging also to one of the other sub-tables, suppress it in this sub-table as well, and repeat the cell suppression procedure for this table. This approach is sometimes called a '*backtracking procedure*', a denotation which we will follow here. Though within a backtracking procedure the cell-suppression procedure will usually be repeated several times for each sub-table, the number of computations required for the protection procedure will be much smaller than when the entire table is protected all at once. It must however be stressed, that a backtracking procedure is not fail-save. E.g. even though each suppressed cell in the table may be protected properly in each subtable, it may still happen, that cells can be disclosed exactly, when all the linear relations between the cells of the entire table are considered. The problem shall be illustrated considering a simple 2 dimensional table without substructure. We may consider this table as set of interrelated 1 dimensional tables, e.g. the set of all the rows and columns of the table. We might then protect each row and each column separately, and only make sure that finally each row and each column has been protected properly, which means each row and each column will contain at least two, or no suppressions at all. This is however not a sufficient criterion for a safe

suppression pattern in a 2 dimensional table (c.f. [3] for counter example). The problem extends analogously to the n-dimensional case. (c.f. [8]).
The speed of the backtracking procedure can be increased when cells appearing in more than one sub-table are given a low probability to be selected as secondary suppression, which can be performed for instance using dynamical weighting schemes (c.f. 4.2.2 “Implementing preferences” below).
A natural way of splitting a hierarchical table into sub-tables would be to split the table into the set of sub-tables without substructure, e.g. in a set of tables constructed in the following way: For any explanatory variable we pick one particular non-bottom-level category. Then we construct a ‘sub-variable’. This sub-variable consists only of the category picked in the first step and those categories of the hierarchical level below, belonging to this category. The table specified through this set of explanatory (sub)variables is free from substructure then, and is a sub-table of the original one, e.g. any cell within the sub-table does also belong to the original table. When we repeat this procedure for any combination of non-bottom-level categories in each dimension, we will have divided the original table into a set of sub-tables without substructure.
Instead of constructing sub-tables without any substructure, we can also construct tables with a less complex substructure. In that case we will not construct the above described ‘sub-variables’ for each explanatory variable, but only for a part of them. The table specified through original (hierarchical) variables in some dimensions and sub-variables in the other dimensions will then have a less complex structure as compared to the original table.

3.2.2 Strategies for table-to-table and data base protection

Table to Table protection
Usually some of the tables in the set of multiple tables published from the same source (e.g. response data from a survey) will be overlapping. Let for instance a table T1 present “turnover by enterprise employee size class”, some table T1.1 present “turnover by NACE and enterprise employee size class”, and some table T1.2 present “turnover by enterprise employee size class and enterprise legal form”. Then T1 is a sub-table of T1.1, as well as of T1.2 , if all the categories of “employee size class” are identical for both tables T1.1 and T1.2 . A cell of the overlap-table T1 will be a sensitive cell of T1.1 if, and only if, this cell is as well a sensitive cell of T1.2 .
When secondary cell suppression is carried out for T1.1 and T1.2 individually, then it is not unlikely, that there will be T1 cells unsuppressed in T1.1, while in T1.2 they are complementary suppressions, and vice versa. Any user given access to both tables T1.1 and T1.2 will be able to disclose these values, and may hence be able to recalculate sensitive cells.
Of course there are possibilities of preventing this situation. One could e.g. protect the ‘full’ table T1.3: “ turnover by enterprise employee size class, NACE, and enterprise legal form” and suppress in T1.1 and T1.2 any cells which also were suppressed in T1.3 . Assume now, maybe due to an exceedingly fine employee size class scheme, it is not possible to protect this table within a single run, because of huge computer resource requirement. In this situation, new versions of τ–ARGUS will offer to apply a table-to-table protection procedure. Like within a backtracking procedure as described above, the software will firstly apply secondary cell suppression to e.g. table T1.1, and then to table T1.2 , keeping track of any secondary suppressions in the overlap table T1. Secondary suppressions in T1, as resulting from protecting T1.1 will be treated like primary suppressions when protecting T1.2, and vice versa. The procedure will be repeated over and over again, until a step of the iteration is reached where no new secondary suppressions have been selected in T1. After the table-to-table protection procedure is finished, any cell of the overlap table T1 is either suppressed in both T1.1 and T1.2, or unsuppressed in both T1.1 and T1.2 . Moreover, none of the suppressions can be disclosed making use of the additive relationship between suppressed and unsuppressed cells in either of T1.1 or T1.2 .

Table-to-table protection in the context of data base query systems
Ideally a table-to-table protection procedure should be applied to the full set of tables ever to be published from this data source. This however seems less and less a realistic option. Nowadays, the process of releasing data turns to be more and more user demand driven and less pre-planned – to the extent even of providing public use data base query systems. This does yet cause serious trouble with

cell suppression. The situation can be improved to some extent, when data are 'pooled' to keep track of suppressions in those tables which have already been published, while still others get newly created. Upcoming versions of τ-ARGUS will be able of setting up such a 'data pool'. One might attempt to use a data pool created as will be illustrated below as data basis for public- or scientific use data base query systems. It shall be stressed here however, that it is not at all within the scope of the project, to implement such a data base query system. Nor do we claim that users or suppliers of data base query systems will be substantially happy with the level of detail or the proportion of unsuppressed low level cells in the data pool.
The data pool as corresponding to a particular micro data basis will contain one and only one record for each cell of any table already protected. This record will contain an entry regarding the suppression status of the cell. When a new table has to be protected, the software will for any cell of this table investigate the data pool. Assume now, the data pool does already contain an entry for this cell. If the cell has already been used as secondary suppression in one of the tables processed earlier, then, like within a backtracking procedure, in the new table it will be treated like a primary suppression. If on the contrary the suppression status for the cell is 'unsuppressed' according to the data pool entry, then the software will attempt to avoid to select this cell as complementary suppression in the new table to some extent. As this will not always be possible, sometimes the user may be forced to either abandon the new table, at least part of it, or to allow for an inconsistency between the suppressions patterns of a table already published and the new table, and hence put up with a risk of disclosure.

Implementing preferences
Strategies like that will be implemented by 'freezing' the previously published cells, e.g. making them uneligible for suppression. The weaker variant - instead of 'freezing' these cells completely - would be to give them a low probability to be selected as secondary suppression. Facilities of that kind can be implemented by manipulating (e.g. increasing) the 'costs' or 'weights' regularly assigned to the suppression of such cells.
In contrast, it may sometimes seem desirable to prefer certain cells as suppressions. When for instance cells had to be included into the table because secondary cell suppression requires a table to be complete, containing e.g. the entire set of linearly interrelated cells, although one actually does not intend to publish these cells. Or, when a cell, which is part of an overlap-table of a set of linked tables, is likely or well suited to be selected as secondary suppression in one of the other tables to be processed later. This would be implemented by decreasing the regular 'costs' assigned to the cell.

As another application for these 'preferences', imagine the situation of a table published periodically, monthly, quarterly, or annually for instance. A part of the sensitive cells may be sensitive in every period. As simple illustration, assume a table without substructure, containing some sensitive cells, which are 'for ever' sensitive. Assume further, that there is more than one feasible suppression pattern, and that the costs for each pattern differ only slightly. If nothing is done, it is then very likely, that the suppression pattern changes from period to period, which might be undesirable, and also cause a risk of disclosure, when the variation in the cell values of the secondary suppressions for different periods is only small. For this example, the problem would be solvable by preferential suppression of suppressions of the previous period.

Co-ordination of suppression patterns as a special application
Specific problems arise when data are published on different levels of a regional classification (e.g. on the national and on the super national (EU) level, or on the regional and national level) but secondary suppressions are to be assigned by different agencies actually (e.g. NSI's and Eurostat, or regional and national statistical institutes). This problem, due to decentralised organisation of official statistics within Europe, will be tackled using the 'preference facilities' of the software as implemented so far. Feasibility of several approaches to improve the situation will be researched with a particular view on the practicability of any methods suggested. The methods will be applied to several real-life data sets, available on national as well as regional level. Methods turning out to be promising should be supported by the software package, e.g. special options may be included in the software.

PART 2: Suggestions for the design of a Graphical User Interface for GHMITER

Draft proposal by StBA

Working document

This second part of our Software concept and design specification document will propose and discuss an user interface for GHMITER, which could, with some modification be used as part of the confidentiality software ARGUS.

The paper is based on a draft specification for a GUI for an earlier version of GHMITER, GHQUAR 2.2, supplied by the company Anite Systems (working document, version of dec. 1st, 1999), who, on behalf of Eurostat, has developed a simple GUI for GHQUAR. The Anite proposal suggests a user interface composed of a tab section on top consisting of three Tabs (data / protection / results) and a static feedback and action section.
This paper will discuss and propose aspects and functionality of the tab sections only. It will introduce a tab-section consisting of six Tabs, in order to provide more functionality.

The proposed tab-section is meant to work as part of an interface for the confidentiality software GHMITER International PC Version and/or for a higher version as currently developed by LDS and DESTATIS. This higher version will include a utility-routine (POOLAC) to be delivered by StBA in 2001 (calculation of weights for flexible control of the selection of suppressions, see sec. 5 below),

This is a first draft version of the proposal, which of course needs to be developed in more detail and moreover will require further discussion, refinement, improvement, etc..

1. First tab: Keys

- The user lists (i.e. types in) names of any key used in the table(s) to be processed.
- **Attributes**: for further processing it is neccessary that the user distinguishes between two different types of key-variables: typical ‘explanatory variables’ (as industrie, region, etc.) and variables, that define decompositions of the response variable (typically: turnover, employees).
- **Codelist**: the user may tick ‘automatic’. In that case the program will extract the list of codes from the data file storing the tabular data. Otherwise codes will be read from a codelist file, which must be supplied by the user.
- **Hierarchy**: if the codes in the codelist have an internal hierarchical structure and the codes are stored in fixed length format, the user may tick ‘automatic’ and give the number of digits used in the code for each level of the hierarchy. The program will then derive the structure of the variable from the codelist file and create corresponding keys suitable for input into GHQUAR (replacement keys).

2. Second tab: Hierarchy

The Hierarchy tab screen visualises the hierarchical structure of a key variable.
Main facilities of the hierarchy tab screen are the following:

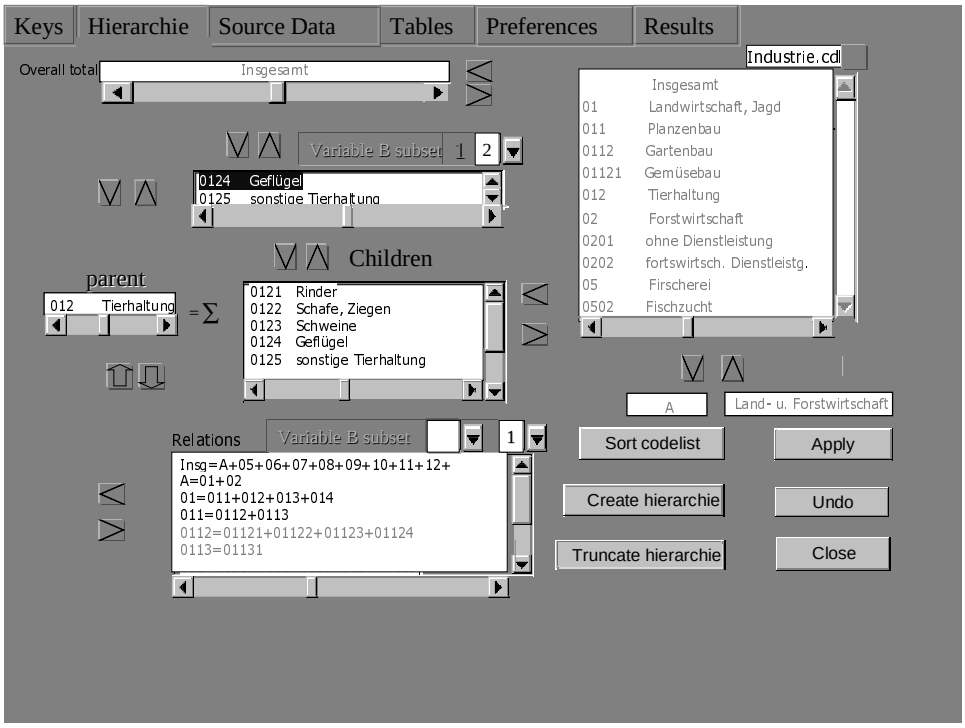
- **Create hierarchy**: Guides the user to express the hierarchical structure of the key-variable correctly.
- **Truncate hierarchy**: The user may derive a sub-hierarchy reduced locally or globally in hierarchical depth.
- **Subset of lables**: The user may specify parts from the basic hierarchy, or from one of the truncated hierarchies to belong to a ‚subset of lables‘.

The non-expert user, if able to supply the codes in such a way that the automatic hierarchy functionality of the KEYS tab screen is applicable, may skip the Hierarchy tab. To do however a more ambitioned table protection job, the facilities of the Hierarchy tab screen provides a user friendly tool to

- define / modify[1] / recode / improve the structure of a variable for the purpose of table protection, yielding ‘better’ suppression patterns (less suppressions), and to
- specify parts/areas of a table to be a ‘local preference area’ (see also tab screen Preferences) either to be avoided for secondary suppression (prepublished cells), or to be preferred indeed (parts of a table, that are not intended to be published anyway or considered less important).

Screen guidance will prevent the user from misspecifications, which might cause errors, breakdown of GHQUAR and, especially in the case of a table-to-table protection procedure, inflict disclosure risks. In particular, screen guidance will allow for introduction of new lables into the hierarchy only in a strictly top-bottom way, while it will allow to drop lables only strictly bottom-top.
For a more detailed description of functionality and user guidance of the Hierarchy tab screen see appendix.

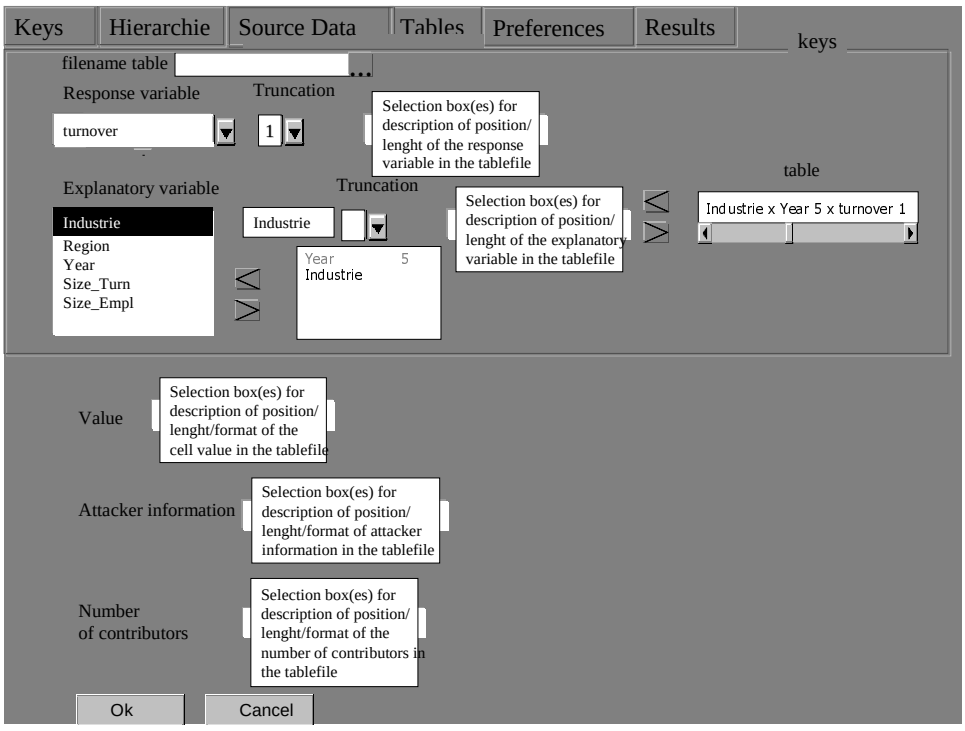
[1] When the automatic hierarchy functionality of the KEYS tab screen has been used, the program will initially supply this automatically created structure.



3. Third tab: Source Data

This tab section provides facilities for the user to describe the structure of the input data file.

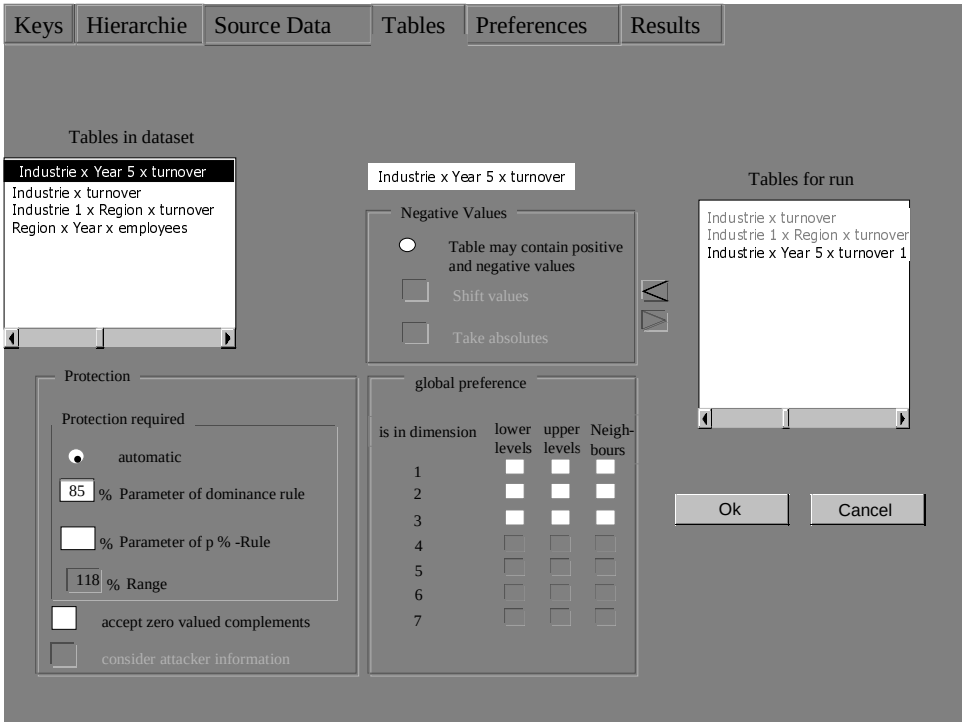
- The user choses one (or no) item from the list of response variables (see first tab: Keys), and
- up to seven items from the list of explanatory variables, as introduced by Keys tab functionality.
- The user may chose to use the original hierarchy for a particular key, or one of the truncations as specified in the 'Hierarchy' step.
- When the user presses the OK button, the input table will be stored in standard (GHQUAR-input?) file format. The original keys will be replaced by GHQUAR replacement keys.



4. Fourth tab: Tables

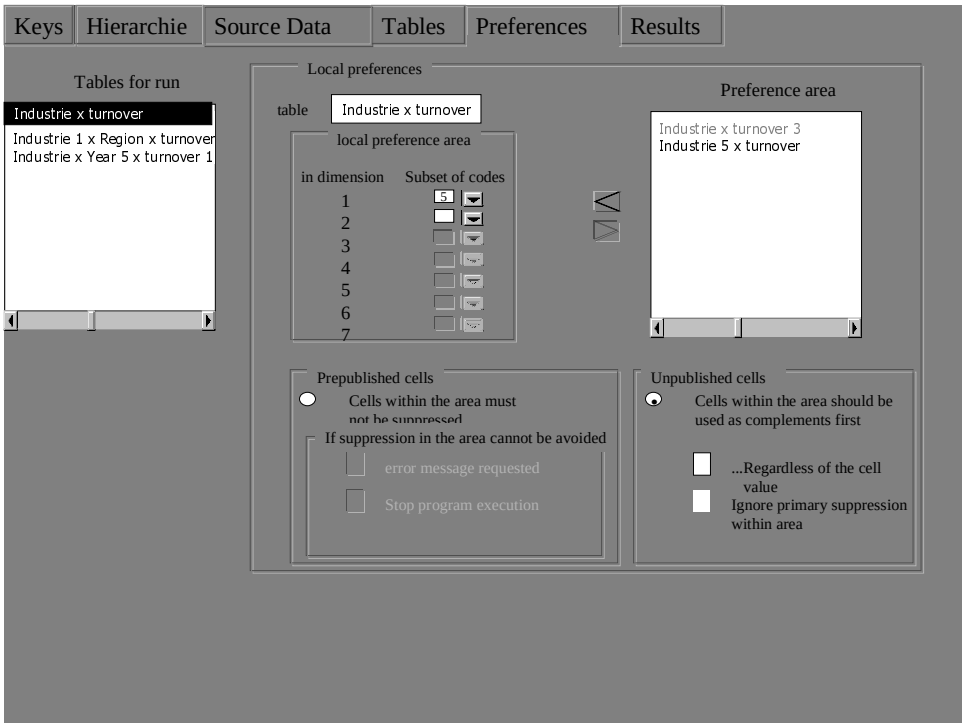
With assistance provided by tab Tables functionality, the user will supply GHQUAR control information ('end block': setting of range criteria, dynamic weighting) for any table selected from those provided by tab Source Data step.

- Protection required: If the user choice is 'automatic' the GHQUAR range parameter will be calculated in such a way that the amount of protection provided by the secondary suppression pattern will correspond to the amount of protection to be given to individual data as requested by setting of the primary suppression rule. The user is allowed to replace the 'automatic' range paramater, but in that case will be warned, that this may cause a risk of disclosure.
- Table-to-table protection: if the user enters more than one table into the 'Tables for run' window, a table-to-table protection procedure will be executed on this set of tables



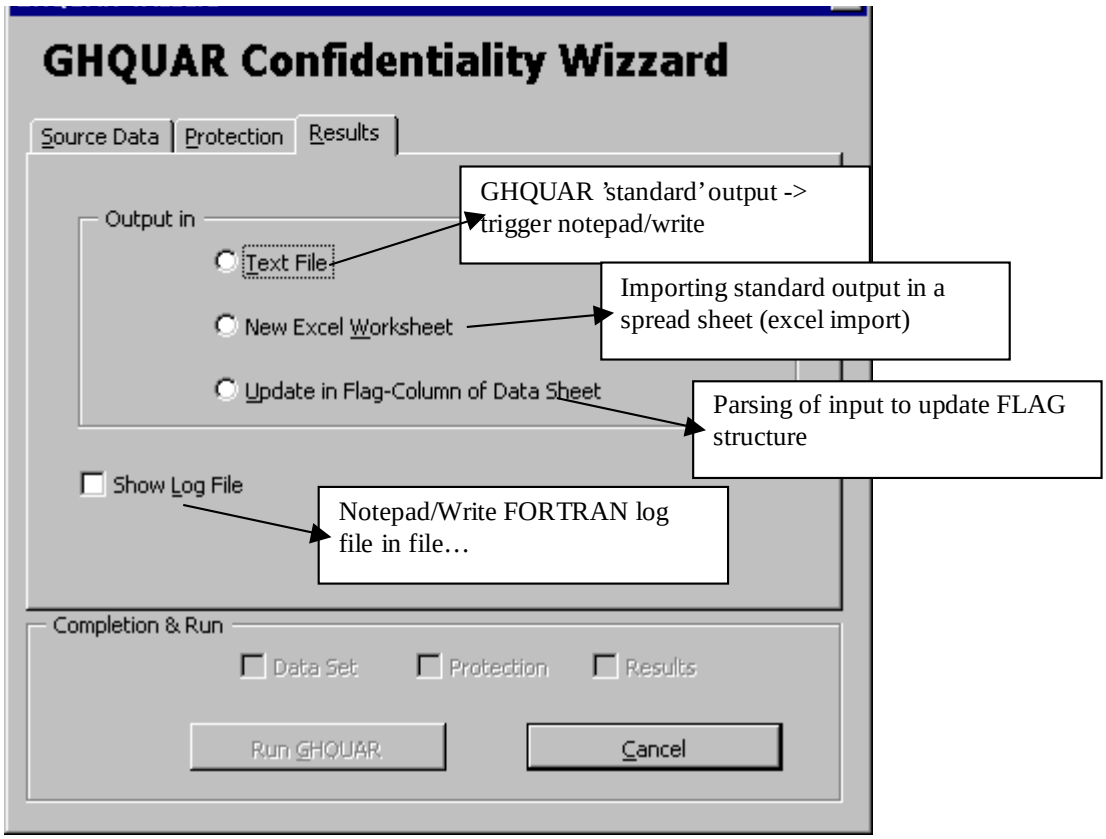
5. Tab: preferences

- Local preference areas: For any table within the current process (as listed in the 'Tables for run' window of the tab Tables screen) the user may define a certain part of a table to be a 'local preference area'. Local preference areas are given as cross combinations from selected 'subset of lables' (c.f. sec. 2.) parts of the codelists (or truncated codelists).
- Two main types of 'local preference areas may be assigned:
  (1) Type prepublished: cells which must not be used as secondary suppressions.
  (2) Type unpublished: cells, which logically belong to the table and must therefore be included in the table protection prodedure, but are considered less 'important' and therefore should be used as secondary suppressions 'first'.
  If the user actually does not intend to publish any cells from a particular local preference area, he may (certain conditions fulfilled) tick the 'ignore primary suppression' box. The program will check whether or not the required conditions are fulfilled indeed. If so, any cells from this area will be flagged 'suppressed', however not requiring to be protected by additional suppressions.
- In a future version GHQUAR will be able to assign a suitable weight to any cell within a 'local preference area'. These weights will control the selection of secondary suppressions according to the user preferences.



6. Tab: Results

Cf. proposal Anite systems:



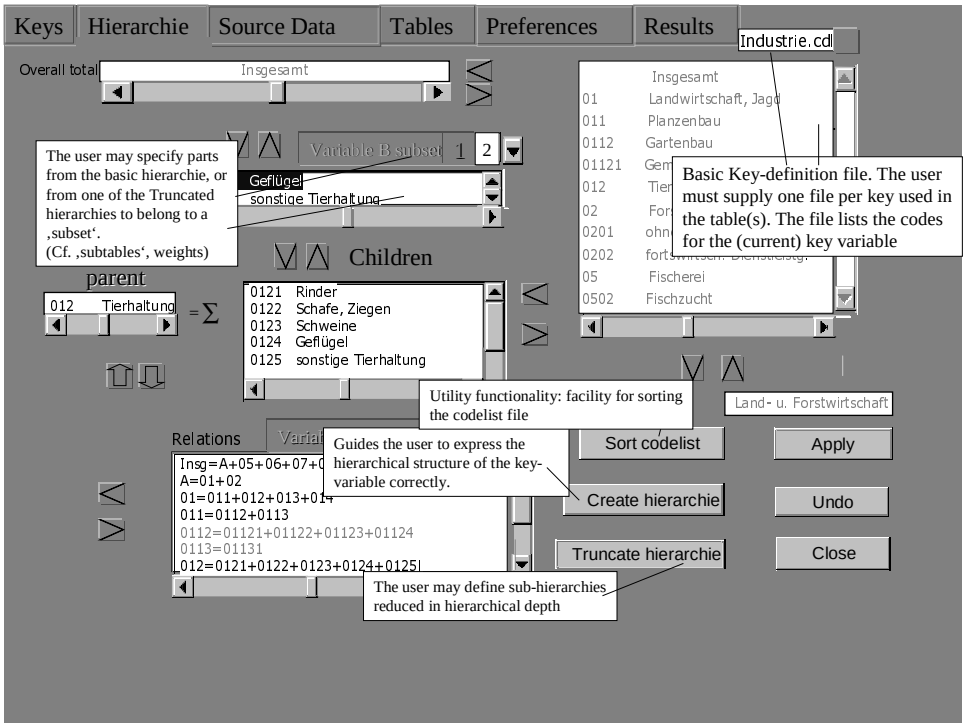
Summary and comments

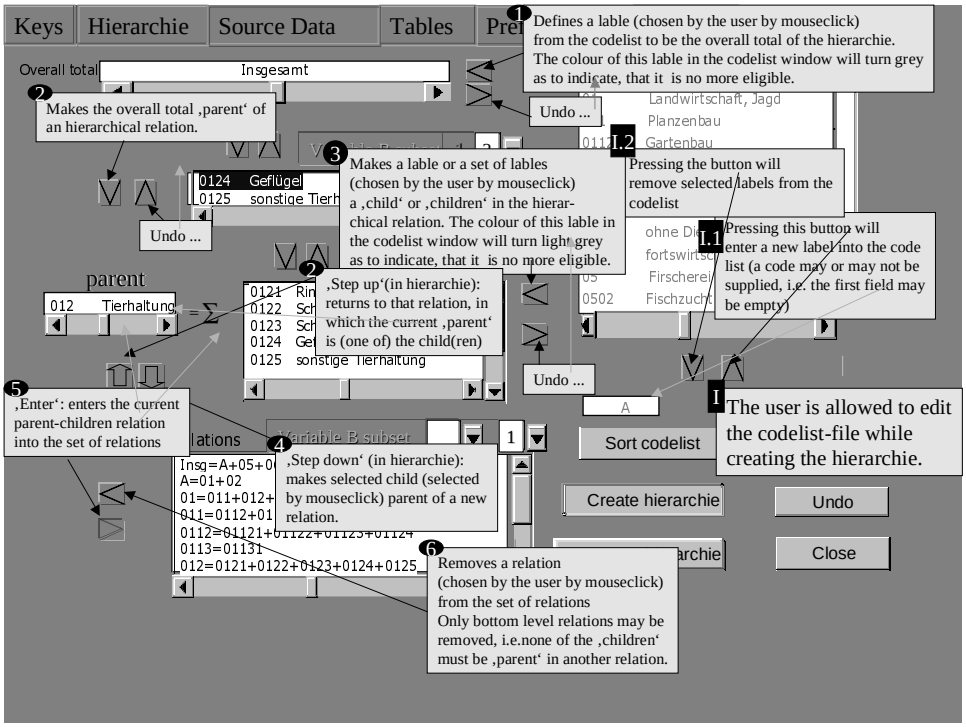
The proposal describes in detail design and functionality of six tab sections of a user interface for GHQUAR.

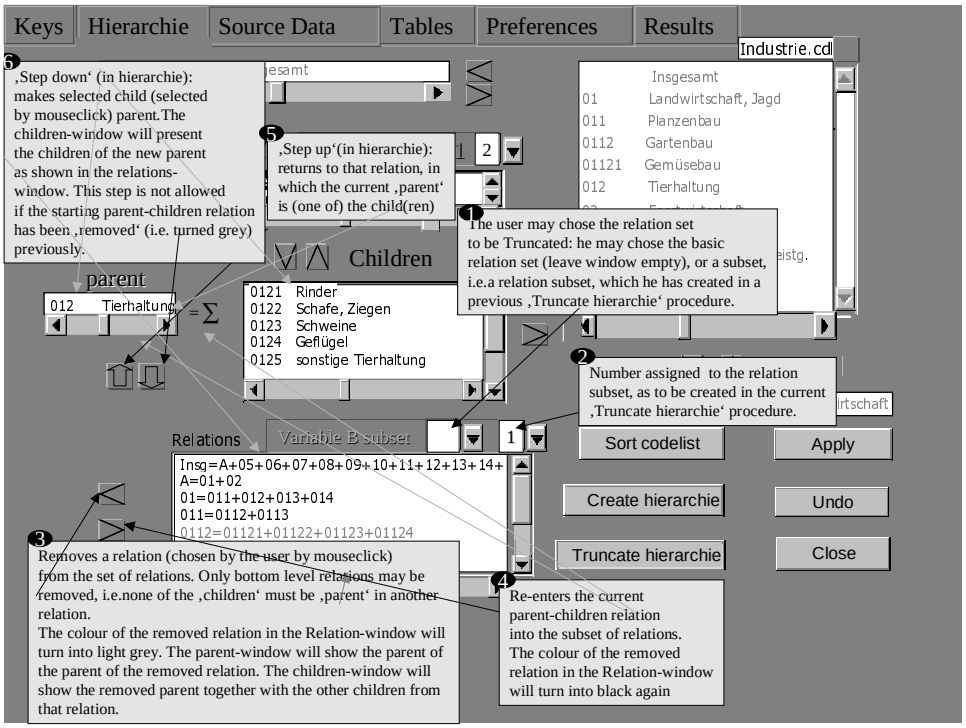
Final remarks

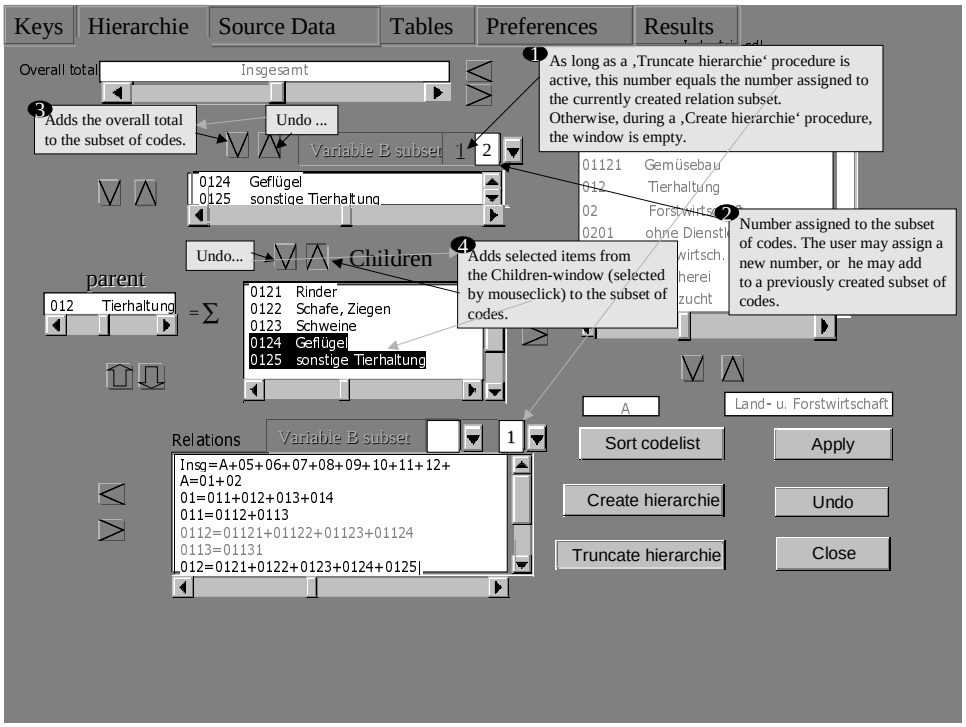
- For a first version tabs 2 (Hierarchy) and 5 (Preferences) might be dropped (and hence the option of using the improved control facilities of the coming GHQUAR version).
- The effort connected to software development of the part of the interface which is needed for table-to-table protection only is rather low. Therefore these facilities should be kept at any rate, also for a first version.

Appendix


Keys
Hierarchie
Source Data
Tables
Preferences
Results
Industrie.cdl
Overall total
Insgesamt
The user may specify parts from the basic hierarchie, or from one of the Truncated hierarchies to belong to a ‚subset'.
(Cf. ‚subtables', weights)
Variable B subset
1
2
Geflügel
sonstige Tierhaltung
Basic Key-definition file. The user must supply one file per key used in the table(s). The file lists the codes for the (current) key variable
Children
parent
012 Tierhaltung
=Σ
0121 Rinder
0122 Schafe, Ziegen
0123 Schweine
0124 Geflügel
0125 sonstige Tierhaltung
Utility functionality: facility for sorting the codelist file
Land- u. Forstwirtschaft
Relations
Guides the user to express the hierarchical structure of the key-variable correctly.
Sort codelist
Apply
Create hierarchie
Undo
Truncate hierarchie
Close
The user may define sub-hierarchies reduced in hierarchical depth


Keys
Hierarchie
Source Data
Tables
1
Defines a lable (chosen by the user by mouseclick) from the codelist to be the overall total of the hierarchie. The colour of this lable in the codelist window will turn grey as to indicate, that it is no more eligible.
Overall total
Insgesamt
2
Makes the overall total ‚parent' of an hierarchical relation.
Undo ...
I.2
Pressing the button will remove selected labels from the codelist
3
Makes a lable or a set of lables (chosen by the user by mouseclick) a ‚child' or ‚children' in the hierar-chical relation. The colour of this lable in the codelist window will turn light grey as to indicate, that it is no more eligible.
I.1
Pressing this button will enter a new label into the code list (a code may or may not be supplied, i.e. the first field may be empty)
parent
012 Tierhaltung
=Σ
2
‚Step up'(in hierarchie): returns to that relation, in which the current ‚parent' is (one of) the child(ren)
Undo ...
5
‚Enter': enters the current parent-children relation into the set of relations
I
The user is allowed to edit the codelist-file while creating the hierarchie.
Sort codelist
4
‚Step down' (in hierarchie): makes selected child (selected by mouseclick) parent of a new relation.
Create hierarchie
Undo
Close
6
Removes a relation (chosen by the user by mouseclick) from the set of relations
Only bottom level relations may be removed, i.e.none of the ‚children' must be ‚parent' in another relation.


Keys
Hierarchie
Source Data
Tables
Preferences
Results
Industrie.cdl
Insgesamt
01 Landwirtschaft, Jagd
011 Planzenbau
0112 Gartenbau
01121 Gemüsebau
012 Tierhaltung
6 ‚Step down' (in hierarchie): makes selected child (selected by mouseclick) parent.The children-window will present the children of the new parent as shown in the relations-window. This step is not allowed if the starting parent-children relation has been ‚removed' (i.e. turned grey) previously.
5 ‚Step up'(in hierarchie): returns to that relation, in which the current ‚parent' is (one of) the child(ren)
1 The user may chose the relation set to be Truncated: he may chose the basic relation set (leave window empty), or a subset, i.e.a relation subset, which he has created in a previous ‚Truncate hierarchie' procedure.
Children
parent
012 Tierhaltung
0121 Rinder
0122 Schafe, Ziegen
0123 Schweine
0124 Geflügel
0125 sonstige Tierhaltung
2 Number assigned to the relation subset, as to be created in the current ‚Truncate hierarchie' procedure.
Relations
Variable B subset
Insg=A+05+06+07+08+09+10+11+12+13+14+
A=01+02
01=011+012+013+014
011=0112+0113
0112=01121+01122+01123+01124
Sort codelist
Apply
Create hierarchie
Undo
Truncate hierarchie
Close
3 Removes a relation (chosen by the user by mouseclick) from the set of relations. Only bottom level relations may be removed, i.e.none of the ‚children' must be ‚parent' in another relation.
The colour of the removed relation in the Relation-window will turn into light grey. The parent-window will show the parent of the parent of the removed relation. The children-window will show the removed parent together with the other children from that relation.
4 Re-enters the current parent-children relation into the subset of relations. The colour of the removed relation in the Relation-window will turn into black again


Keys
Hierarchie
Source Data
Tables
Preferences
Results
Overall total
Insgesamt
3 Adds the overall total to the subset of codes.
Undo ...
1 As long as a ‚Truncate hierarchie' procedure is active, this number equals the number assigned to the currently created relation subset. Otherwise, during a ‚Create hierarchie' procedure, the window is empty.
Variable B subset
01121 Gemüsebau
012 Tierhaltung
02 Forstwirtsch.
0201 ohne Dienstl.
0124 Geflügel
0125 sonstige Tierhaltung
2 Number assigned to the subset of codes. The user may assign a new number, or he may add to a previously created subset of codes.
Undo...
Children
4 Adds selected items from the Children-window (selected by mouseclick) to the subset of codes.
parent
012 Tierhaltung
0121 Rinder
0122 Schafe, Ziegen
0123 Schweine
0124 Geflügel
0125 sonstige Tierhaltung
A
Land- u. Forstwirtschaft
Relations
Variable B subset
Insg=A+05+06+07+08+09+10+11+12+
A=01+02
01=011+012+013+014
011=0112+0113
0112=01121+01122+01123+01124
0113=01131
012=0121+0122+0123+0124+0125
Sort codelist
Apply
Create hierarchie
Undo
Truncate hierarchie
Close