

# **ESSNet on common tools and harmonised methodology for SDC in the ESS**

## **The structure of the ARGUS software**

**Anco Hundepool**

**28 July 2011**

**Draft**

### **1. Introduction**

The ARGUS software has been developed to meet the needs of statistical institutes for the production of safe microdata and safe tables. The origin of the software goes back to initiatives at Statistics Netherlands to develop tools for their own needs. However the needs for tools was not restricted to the Netherlands alone. The problems of protecting tables and microdata exist of course in all NSIs. Statistics Netherlands has a longer tradition of developing general applicable software for solving the needs of statistical institutes. A good example of this is the Blaise system. The development of Blaise however was also very demanding for the development capacity available at Statistics Netherlands. Apart from Blaise Statistics Netherlands was working on the Statline system for publishing statistical information and some other smaller systems, like Abacus a tabulation system. This all together was an obstruction for undertaking another larger scale development project, like software for confidentiality. However the need existed and there was also the desire to undertake something.

In an early stage the European option was chosen for finding resources to develop ARGUS. This has proved to be very successful. Also at the European level the need for tools for confidentiality protection was recognised. This has led to the 4<sup>th</sup> framework SDC project. During this project a prototype for  $\mu$ -ARGUS was developed into a mature program and also the start was made with  $\tau$ -ARGUS. In  $\mu$ -ARGUS an implementation was made of the approach developed at Statistics Netherlands to produce Microdata Files for Researchers (RDF) as well as Public Use Files (PUF).  $\tau$ -ARGUS was developed from scratch. The work on  $\mu$ -ARGUS was mainly done at Statistics Netherlands, but for  $\tau$ -ARGUS serious contributions have been made by M. Fischetti and JJ Salazar. They have developed the first models for the solution of the optimisation problems behind cell suppression. JJ Salazar also made the first real implementation of these models.

The work has progressed since then and a significant step forwards was the CASC-project.  $\tau$ -ARGUS was extended by including the modular approach, allowing to protect larger hierarchical tables.

### **2. $\mu$ -ARGUS**

#### **2.1. General Overview**

$\mu$ -ARGUS was build originally to automate the SDC-rules for microdata developed at Statistics Netherlands. But it has grown and now also includes other methods:

- Alternative risk-models, based on Luisa Franconi's work at IStat
- Microaggregation, a procedure build by Josep Domingo at URV
- Rank swapping, a procedure build by Josep Domingo at URV
- Qualitative microaggregation, a procedure build buy Vicenc Torra, CSIC
- PRAM, developed at Statistics Netherlands

- Sullivan's masking, developed by Destatis

The general structure of  $\mu$ -ARGUS is a Visual Basic user interface which controls the flow of the program. The real hard work is done by C++ DLLs.

- NewMuArg.dll

This dll contains routines for

- general data manipulation procedures, input/output
- tabulation procedures for frequency tables needed by the Dutch and Italian risk models
- routines for global recoding
- routines for local suppression
- PRAM routines

- Catalaan.dll

The Microaggregation and the rankswapping have been developed by Josep Domingo as C-functions. They have been included in a C++ com-dll for the use in  $\mu$ -ARGUS.

- QualMicAgg.dll

The qualitative microaggregation has been developed by Vicenc Torra in C. These functions have been included in a C++ com-dll.

- Sullivan's masking

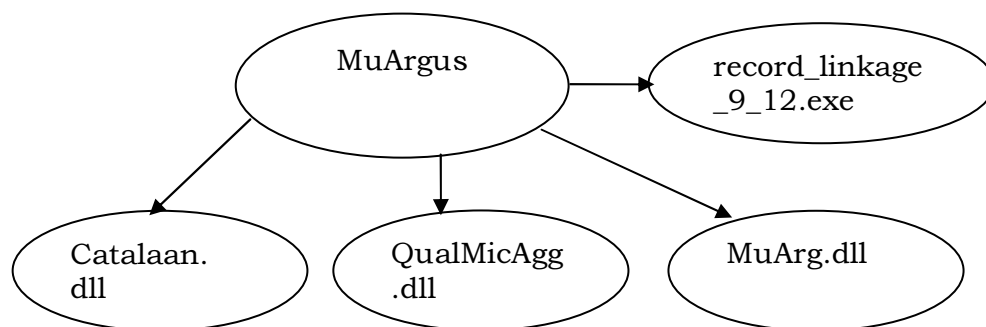
This procedure has been developed by Ruth Brandt Destatis. It has been written in GAUSS. However the routine was a good experiment, but proved to be applicable only to very small datasets (1000 records). As there has not been much interest, this procedure will not be supported any more in future versions of  $\mu$ -ARGUS. Therefore we will not pay any more attention to this.

- record\_linkage\_9\_12.exe. A record linkage program

The interaction between NewMuArg handling the data and Catalan.dll and QualMicAgg.dll is done via files written in the temp-directory. Although this is not the most efficient way from an IT-point of view, this is however very practical during the development of the software. External developers only had to concentrate on treating the input files and creating the required output files. This enabled the development of these procedures while the rest of  $\mu$ -ARGUS was still under development.

## **2.2. $\mu$ -ARGUS.VB user interface**

$\mu$ -ARGUS is a Visual Basic program that controls the flow through the whole process.



### 2.2.1. frmMain.frm

Variable	dim 1	dim 2	dim 3
REGION	2	67	11
SEX	0	117	11
AGE	0	19	26
MARSTAT	0	104	23
KINDPERS	0	186	45
NUMYOUNG	0	8	33
NUMOLD	0	6	10
AGEYOUNG	0	19	82
EDUC1	0	250	64
EDUC2	0	389	65
ETNI	0	106	17
PRIOCU	0	236	39
POSLABM	0	55	16
REGJOBC	0	58	16
RECBEN	0	82	22
RECUNBEN	0	21	4
RECOBEN	0	50	9
RECBILL	0	34	5
RECSOSEC	0	19	3
RECPENS	0	53	8
POSLABLY	0	86	19
POSFACT	0	87	21

Code	Label	Freq	dim 1	dim 2	dim 3
1	Aalburg	44	0	49	90
2	Aalsmeer	18	0	45	78
3	Aalten	9	0	19	34
4	Ter Aar	13	0	29	64
5	Aardenburg	10	0	23	50
6	Aarle-Rixtel	12	0	27	51
7	Abcoude	28	0	58	81
8	Achtkarspelen	12	0	29	43
9	Akersloot	7	0	25	56
10	Alblasserdam	20	0	38	69
11	Albrandswaard	11	0	22	33
12	Alkemade	43	0	55	98
13	Alkmaar	16	0	39	54
14	Almelo	16	0	39	65
15	Almere	10	0	31	56
16	Alphen aan d...	19	0	43	80
17	Alphen en Riel	4	0	15	39
18	Ambt Delden	2	0	28	54
19	Ambt Montfort	18	0	43	81
20	Ameland	13	0	37	60
21	Amerongen	8	0	25	46
22	Amersfoort	7	0	23	43
23	Ammerzoden	16	0	36	63

### Overview of the menu structure

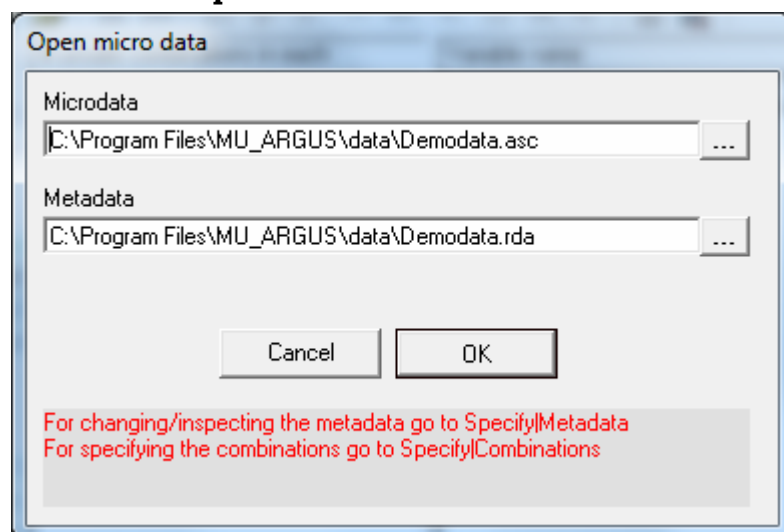
File	Specify	Modify	Output	Tools	Help
Open microdata	Metadata	Show Table Collection	Make protected file	Record linkage	Contents
Exit	Combinations	Global recode	View report		News

		PRAM specification			About
		Individual Risk specification			
		Household Risk specification			
		Synthetic data			
		Modify numeric var			
		Numerical Microaggregation			
		Numerical Rank Swapping			

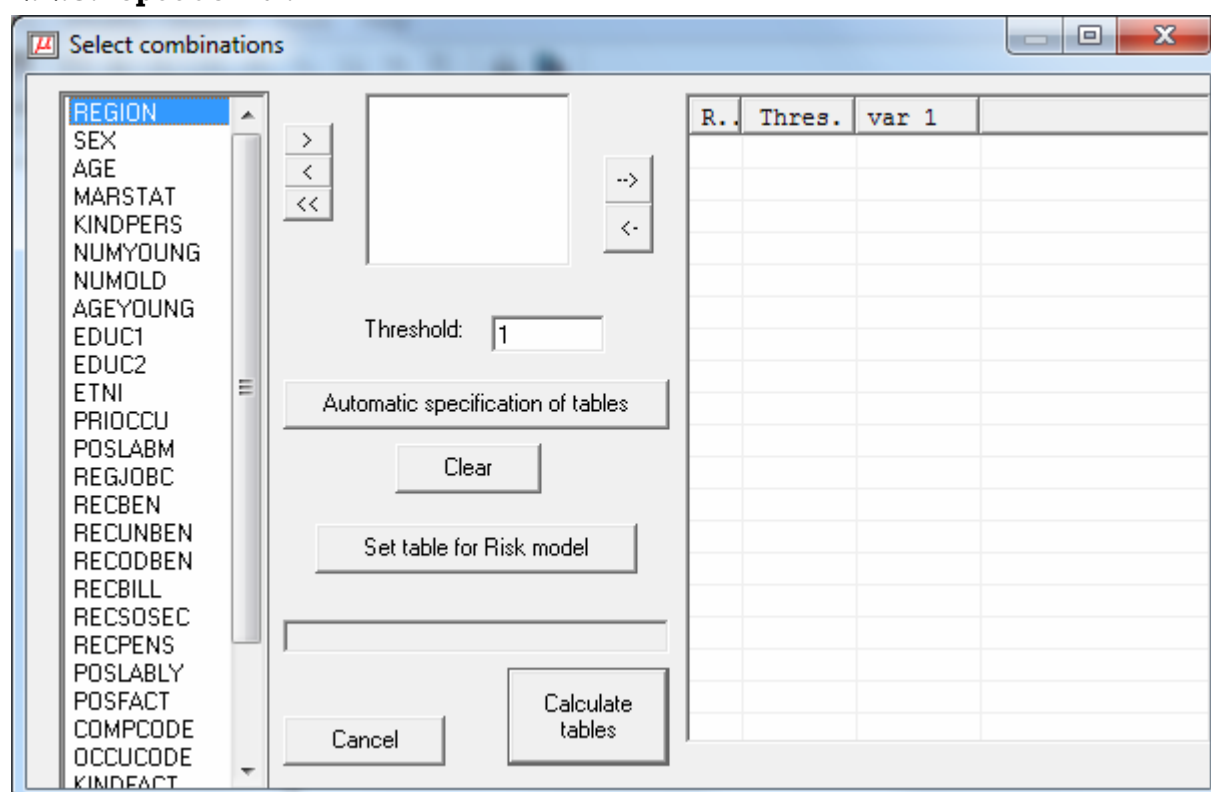
#### Overview of the sub-forms in the menu structure

File	Specify	Modify	Output	Tools	Help
-	frmSpecmet adata.frm	frmShowTables.frm	frmOutput.f rm	frmRec ordLink age.frm	-
-	SpecCombi. frm	frmGlobalRecode.fr m	frmViewRep ort.frm		frmVie wRepor t.frm
		frmSpecPRAM.frm			frmAbo ut.frm
		frmRiskChart.frm			
		frmRiskChart.frm			
		frmSyntheticData.fr m			
		frmNumericVars.fr m			
		frmMicroAggregatio n.frm			
		frmMicroAggregatio n.frm			

### 2.2.2. Frm Open file



### 2.2.3. SpecCombi.frm



Specifies the combinations for which the frequency tables have to be computed. The tables can be generated also using a generator based on Statistics Netherlands rules. If needed additional information for the Italian risk models are computed.

**Numerical Micro Aggregation**

Variable		Selected	
WEIGHT	> <	INCOME	^ v
INCOME		ASSETS	
ASSETS			
DEBTS			

Minimum Number of Records per Group:

☐ Use optimal method

**Partitioning**  
 Partitioning is not recommended for this file

☒ No partitioning  
☐ By block size  
☐ By Variable

-6-

### 2.2.5. frmNumericVars.frm

Modify numerical variables

M..	Variable
	WEIGHT
	INCOME
	ASSETS
	DEBTS

BottomCoding  
Minimum: 112.7  
BottomValue:   
Replacement:

TopCoding  
Maximum: 420  
TopValue:   
Replacement:

Round  
Rounding base:

WeightNoise  
Percentage:

OK

Various numerical transformation can be applied.

### 2.2.6. frmOutput.frm

**Make the protected file**

Suppression

☐ No suppression

☐ Use weights

☒ Use entropy

Suppression weight per variable

Variable	Prior.
REGION	50
SEX	50
AGE	50
MARSTAT	50
KINDPERS	50
NUMYOUNG	50
NUMOLD	50
AGEYOUNG	50
EDUC1	50
EDUC2	50
ETNI	50
PRIOCCU	50
PCSI ARM	50

HHIdent

☒ Keep in safe file

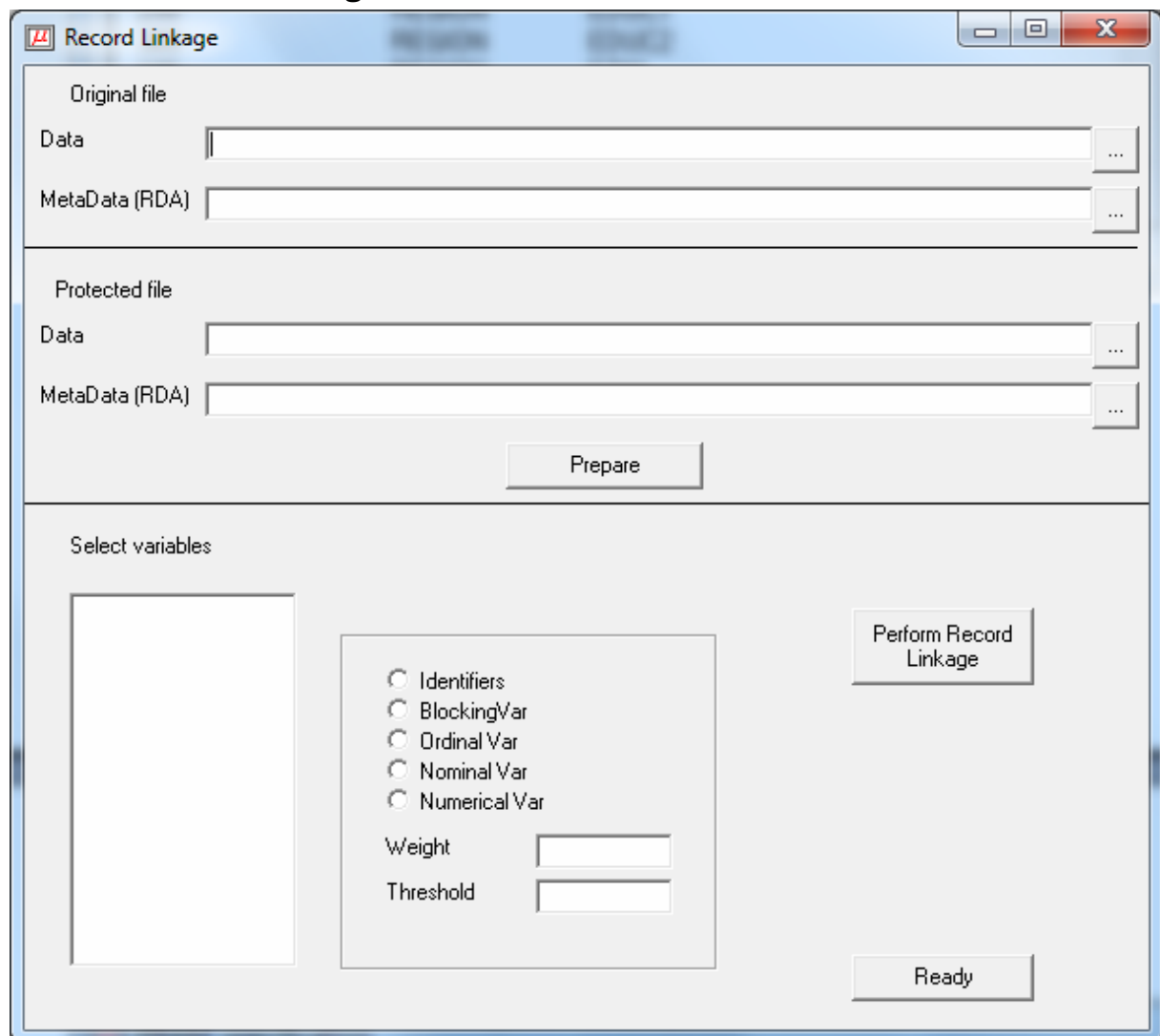
☐ Change into sequence number

☐ Remove from safe file

☐ Write records in random order

Cancel Make file

### 2.2.7. frmRecordLinkage.frm

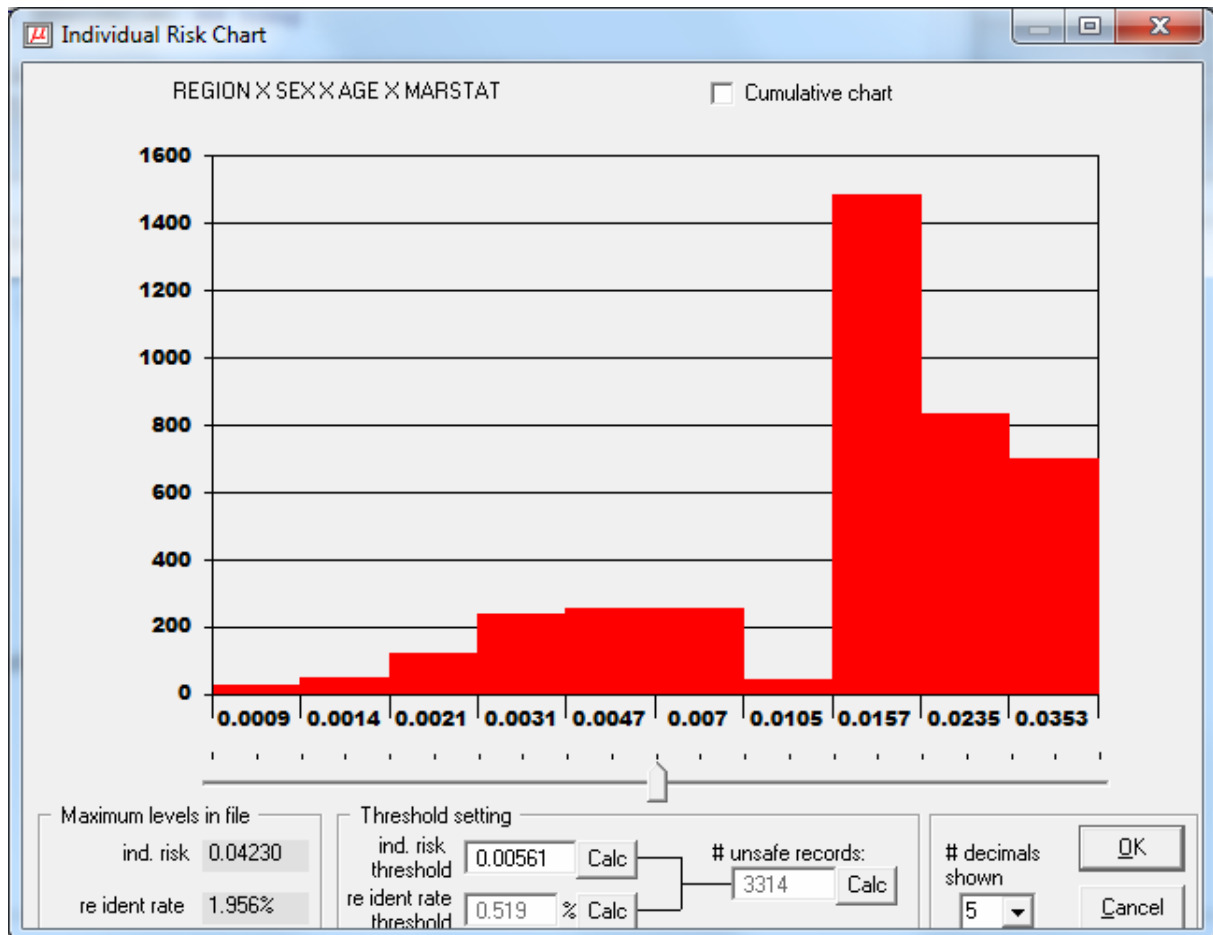


The screenshot shows a Windows-style dialog box titled "Record Linkage". It is divided into three main sections. The top section, "Original file", contains two text input fields: "Data" and "MetaData (RDA)", each followed by a browse button ("..."). The middle section, "Protected file", also contains two text input fields: "Data" and "MetaData (RDA)", each followed by a browse button ("..."). Below these fields is a "Prepare" button. The bottom section, "Select variables", features a large empty rectangular box on the left. To its right is a list of variable types with radio buttons: "Identifiers", "BlockingVar", "Ordinal Var", "Nominal Var", and "Numerical Var". Below this list are two more input fields labeled "Weight" and "Threshold". On the right side of this section are two buttons: "Perform Record Linkage" and "Ready".

Calls the external program record\_linkage\_9\_12.exe



### 2.2.8. frmRiskChart.frm



Displays a risk chart and gives the possibility to specify the parameters for the risk model

### 2.2.9. frmShowTables.frm

The screenshot shows the 'Overview of table collection' window. The title bar reads 'Overview of table collection'. There is a checkbox for 'Show all tables' which is currently unchecked. To the right of this checkbox is a 'Select variable:' dropdown menu set to 'all'. Below this is a table with 5 columns: '# unsafe cells', 'Var 1', 'Var 2', 'Var 3', and 'Var 4'. The table contains 15 rows of data. At the bottom of the window is a 'Close' button.

# unsafe cells	Var 1	Var 2	Var 3	Var 4
2	REGION			
3	COMPCODE			
49	OCCUCODE			
15	REGION	SEX		
1948	REGION	AGE		
104	REGION	MARSTAT		
186	REGION	KINDPERS		
8	REGION	NUMYOUNG		
6	REGION	NUMOLD		
19	REGION	AGEYOUNG		
250	REGION	EDUC1		
389	REGION	EDUC2		
106	REGION	ETNI		
236	REGION	PRIOCCU		
55	REGION	POSTLABM		
58	REGION	REGJOBC		

Displays an overview of all the frequency tables.

### 2.2.10. frmSpecPRAM.frm

**PRAM specification**

Variables:

F	B.	Variable
		REGION
		SEX
		AGE
		MARSTAT
		KINDPERS
		NUMYOUNG
		NUMOLD
		AGEYOUNG
		EDUC1
		EDUC2
		ETNI
		PRIOCCU
		POSLABM
		REGJOBC
		RECBEN
		RECUNBEN
		RECODBEN
		RECBILL
		RECSOSEC
		RECPENS
		POSTLABV

Codes

Code	Label	Prob.
1	Aalborg	0
2	Aalsmeer	0
3	Aalten	0
4	Ter Aar	0
5	Aardenburg	0
6	Aarle-Rixtel	0
7	Abcoude	0
8	Achtkarspelen	0
9	Akersloot	0
10	Alblasserdam	0
11	Albrandswaard	0
12	Alkemade	0
13	Alkmaar	0
14	Almelo	0
15	Almere	0
16	Alphen aan de Rijn	0

PRAM Options

☒ Individual chances

☐ Based on Global Recode

Default Probability

80

Set all codes to default

Bandwidth

☐ Use bandwidth

5

Apply Undo Close

Specification of the PRAM-parameters

### 2.2.11. frmSpecmetadata.frm

**Specify metadata**

Fixed format

REGION

Attributes

Name: REGION

Length: 1

Length: 4

Decimals: 0

Options for Argus

Identification level: 1

Weight for local suppression: 50

Categories

☒ Truncation allowed

☒ Codelistfile

regio.cdl

Variable Type

☐ HH Identifier

☐ HH Variable

☐ Weight

☒ other

☒ Categorical

☐ Numerical

Related to: --none--

Missings

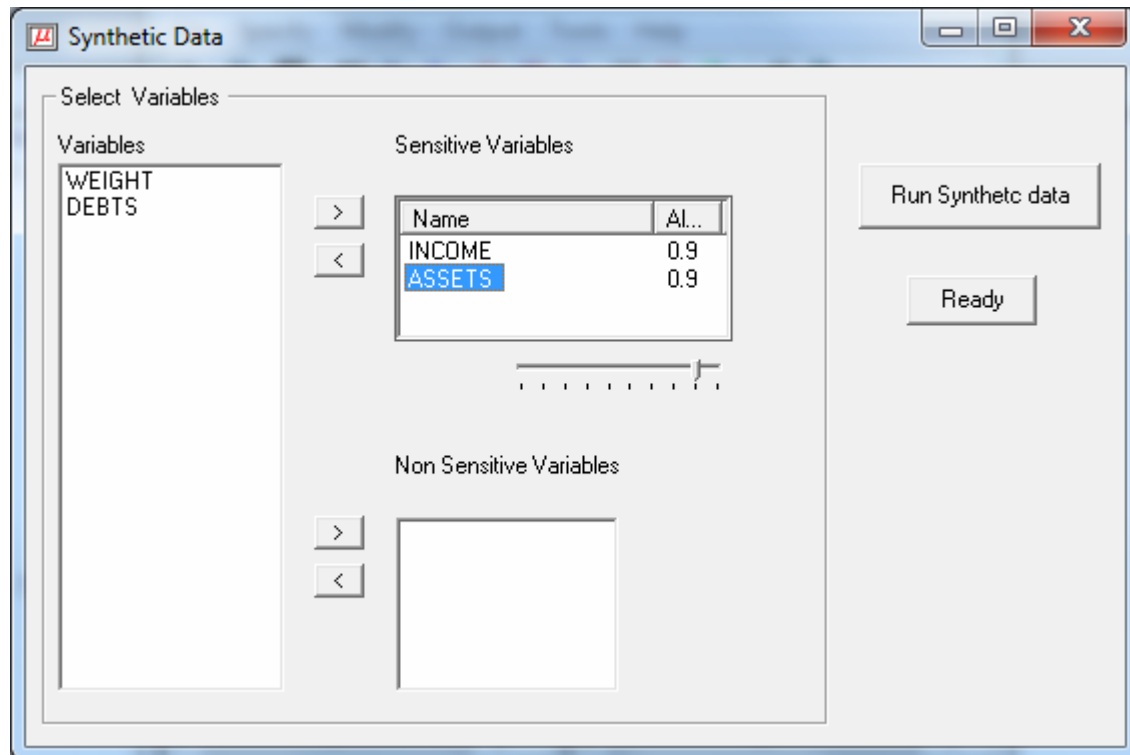
1: 9999

2: 9998

Generate New Delete OK Cancel

Modification of the meta data stored in the MetaDatastruct

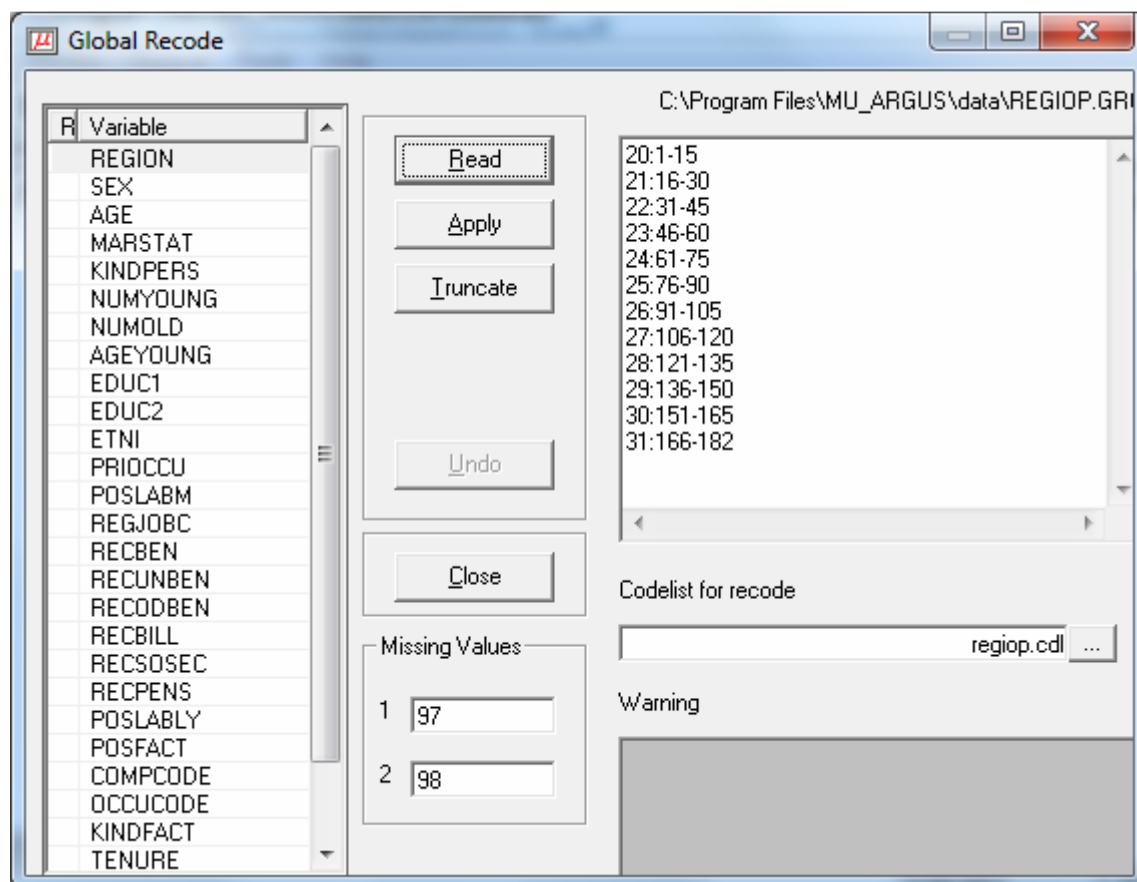
### 2.2.12. frmSyntheticData.frm



Generates synthetic data by calling an R-routine. It requires that R has been installed on the PC.

## Joint $\mu$ -ARGUS and $-\tau$ -ARGUS form

### 2.2.13. frmGlobalRecode.frm



Performs the global recodings

### 2.2.14. $\mu$ -ARGUS modules

muArgusData.bas

Contains the basic data structures for  $\mu$ -ARGUS.

MuFunctions.bas

SPSSFunctions.bas

### 2.2.15. MuTau modules

ArgusFunctions.bas

MuTauArgusData.bas

Exec.bas

## 2.3. Technical documentation MuArg.OCX

MuArg.OCX is a non-graphical ActiveX component that provides functions and events to make safe micro-files. As a development environment is used Visual C++ 6.0.

*For protection, the following tools available:*

- For numeric variables:

*top / bottom coding*

All values above or below a certain value are replaced by a fixed value

*Controlled rounding*  
values are rounded

- For weights:

*perturbate sampling weights*

A weight with a certain percentage will be increased or reduced

- For categorical variables:

*Interactive global recode*

The number of categories is reduced, for example, municipality (600) to province (12) or from age to age-class

*Post Randomization Method (PRAM)*

The alphanumeric code with a certain probability changed to another existing, for example, a mayor can be a plumber. It is possible to limit the distance from the original code (rather nice with age to prevent an infant to be changed into an old man.)

- For a combination of categorical variables:

*Local suppression*

The frequency is calculated for each combination of categories of variables in a table. A combination of which a frequency is less than or equal to a specified threshold is unsafe. One or more of the missing variables will be imputed when making a safe file. This procedure also applies to all subtables or marginalis. For example, a combination of three variables A, B and C, we also look at the combinations AB, AC and BC and A, B and C. Should identical tables be created then only the table with the highest threshold is used. A combination (table or subtable) of which at least one of the variables participates in the *Post Randomization Method* is not protected, as it is considered safe by *changing* a category.

*Base Individual Risk (BIR)*

Intended for sample files. Both the frequency and the sum of the weights are calculated for each combination of categories of variables in a table. A certain algorithm (input frequency and total weights, outputs a number between 0 and 1) determines the degree of safety.

There are the following restrictions:

- The variables of a BIR-table are not included in other tables (including other tables BIR).
- It is assumed that the threshold for BIR is properly put.
- None of the variables, the *Post Randomization Method* used.

*Missing:*

- Each categorical variable has at least one *missing value*, A second one is possible.
- A numeric variable (not being a weight) may have one or two *missing values*.
- It is required that the weight is numeric, but it has no *missing values*.
- When creating a safe file one or more categorical variables are set to *missing*. That is always the first *missing value*.

*For protection, the following events and functions are available:*

Each function and event includes:

1. A brief description
2. The parameters of type:

- LPCTSTR string = (input)
- Bstr = string (return)
- double = 8-byte real
- Long = 4-byte integer
- BOOL = Boolean, true or false only
- short = 2-byte integer
- $\Rightarrow$  a type: return parameter

### 3. Return value

nUC means **n**umber of **U**nsafe **C**ombinations.

#### 2.3.1. Events

##### Update progress

Description:

Indicates the percentage of a certain action is resolved.

Is fired in the functions file Explore , ComputeTables and MakeFileSafe .

Parameters:

$\Rightarrow$  short perc      Percentage processed

Return value:

no

---

#### 2.3.2. Functions

In alphabetical order:

- ApplyRecode
- BaseIndividualRisk
- CleanAll
- ClosePramVar
- ComputeTables
- DoRecode
- DoTruncate
- FileExplore
- GetBIRHistogramData
- GetMaxnUC
- GetMinMaxValue
- GetTableUC
- GetVarCode
- GetVarProperties
- MakefileSafe
- MakefileSafeClearOptions
- SetBIRThreshold
- SetCodingBottom
- SetCodingTop
- SetNumberTab
- SetNumberVar
- SetPramValue
- SetPramVar
- SetRound

- SetSuppressPrior
- SetTable
- SetVariable
- SetWeightNoise
- UndoRecode
- UnsafeVariable
- UnsafeVariableClose
- UnsafeVariableCodes
- UnsafeVariablePrepare

In functional order:

1. Specify variables
  - SetNumberVar
  - SetVariable
2. File explore
  - FileExplore
  - BaseIndividualRisk
3. Specify tables
  - SetNumberTab
  - SetTable
4. Calculating tables
  - ComputeTables
5. NUC request for variables and variable codes
  - UnsafeVariable
  - UnsafeVariablePrepare
  - UnsafeVariableCodes
  - UnsafeVariableClose
6. Records Request
  - GetTableUC
  - GetMinMaxValue
  - GetVarCode
7. Recoding
  - DoTruncate
  - DoRecode
  - ApplyRecode
  - UndoRecode
8. MaxUC
  - GetMaxnUC
9. Preparations for making a safe file
  - MakeFileSafeClearOptions
  - SetRound
  - SetCodingTop
  - SetCodingBottom
  - SetSuppressPrior
  - SetWeightNoise
  - SetPramVar
  - SetPramValue
  - ClosePramVar
  - GetBIRHistogramData
  - SetBIRThreshold
10. Creating safe file
  - MakeFileSafe
11. Results of making a safe file

- GetVarProperties
- 12. Cleaning
  - CleanAll

### 2.3.2.1. SetNumberVar

Description:

Specifies the number of variables, reserves memory for the administration of the variables.

First calls CleanAll to, in case a new action takes place.

Parameters:

long n\_var number of variables

Return value:

false if n\_var is not correct (<1) or insufficient memory, otherwise true

### 2.3.2.2. SetVariable

Description:

Specifies the properties of a variable.

Parameters:

long	Index	1, 2, ... n_var, index of variable
long	BPOs	1, 2, ... starting position in micro data file
long	nPos	1, 2, ... number of positions in micro data file, up to 100
long	nDec	0.1 ,.... number of decimal places (especially important when writing safe file)
LPCTSTR	Missing1	code for the first missing
LPCTSTR	Missing2	code for the second missing
BOOL	IsHHIdent	is identification variable for household
BOOL	IsHHVar	is a household variable
BOOL	IsCategorical	is a categorical variable
BOOL	IsNumeric	is a numerical variable
BOOL	IsWeight	is a weight

If Missing1 is empty Missing2 promoted. If both are empty is false returned. If both are equal is Missing2 deemed to be empty.

If *IsWeight* is true is *IsNumeric* is also required to be true, regardless of its value. Both *Missings* are ignored in this case without warning.

Return value:

false if one or more parameters are wrong, otherwise true

### 2.3.2.3. ExploreFile

Description:

Examines each *categorical* variable which codes occur. In the fixed format input file, all records are of equal length. As the only exception are empty records, which are ignored without warning.

Parameters:

LPCTSTR	FileName	Name of the file to be investigated
⇒ long	ErrorCode	See below
⇒ long	LineNumber	Line number where error occurs



⇒ long      VarIndex      Index of numeric variable considered, which is not numeric (0 = N)

Return value:

false if an error occurs, its nature is in error code:

- FILENOTFOUND the file can not be opened
- EMPTYFILE the file is empty
- WRONGLENGTH not all record lengths are equal, LineNumber gives the record number
- RECORDTOOSHORT a variable does not fit within the specified record length
- NOVARIABLES there are no variables specified

else true

#### 2.3.2.4. SetNumberTab

Description:

Specifies the number to tables to compute. Clears the previously specified tables, if any.

Parameters:

long NTAB Number of tables

Return value:

false if NTAB is incorrect or insufficient memory, otherwise true

#### 2.3.2.5. Settable

Description:

Specifies the attributes of a table. nDim indicates the number of dimensions, VarList the indices of the categorical variables. The *threshold* is not important in *IsBIR = true*; For BIR, the threshold is specified after inspection of the histogram .

Parameters:

long	Index	Index of the table (1, 2, ... NTAB)
long	Threshold	Threshold
long	nDim	Number of dimensions (max 10)
long	VarList	The index for each dimension of the variable (1, 2, ... n_var)
BOOL	IsBIR	Yes / no individual risk-base table, see Risk individual base
long	BIRWeightVarIndex	Variable index of BIR-associated weight

Return value:

False if specification errors, otherwise true

#### 2.3.2.6. ComputeTables

Description:

Calculates all with SetTable specified tables from the data file, and ll subtables thereof, for example, for table ABC also AB, AC, BC, A, B and C. This also applies to the tables with the BaseIndividualRisk property. For each (sub) table is calculated how many table cells are greater than 0 and less than or equal to the threshold. All tables are stored in memory.

Parameters:

- ⇒ long error code    See below, n = -1
- ⇒ long table index    Index of the table where the error occurred (1, 2, ... nTAB) -1 = Not applicable

Return value:

false if an error occurs, its nature is in error code:

- NOVARIABLES there are no variables specified
- NOTABLES there are no tables specified
- NOTENOUGHMEMORY all tables together have too much memory, at the moment the limit is 50M.  
This is to prevent the computer from *swapping* which unfortunately has unpleasant effects.
- NOTABLEMEMORY for a table is not a memory
- NODATAFILE there is no file specified to examine
- PROGRAM ERROR-self-explanatory

else true

#### **2.3.2.7. UnsafeVariable**

Description:

Calculates for each relevant dimension (1, 2, ..) the number of UCs of a variable.

Parameters:

- long VarIndex 1, 2, ... n\_var, index of variable
- ⇒ long Count    Number of elements in UCArrary
- ⇒ long UCArrary    Array of UCS of dimensions 1, 2, ... Count

Return value:

false if one or more parameters are wrong, otherwise true

#### **2.3.2.8. UnsafeVariablePrepare**

Description:

Prepares for a variable the information by code.

In m\_unsafe [nCode] [MAXDIM + 1] (see program) is stored for each code:

- frequency (index 0)
- for each relevant dimension (1, 2, ...) the number of UCS (index 1, 2, ...).

Any action not completed any other variable is reversed so

UnsafeVariableClose not necessarily required.

Parameters:

- long VarIndex 1, 2, ... n\_var, index of variable
- ⇒ long nCode    Number of codes of the variable, including *missing*

Return value:

false if one or more parameters are wrong (eg no categorical variable),  
otherwise true

#### **2.3.2.9. UnsafeVariableCodes**

Description:

Indicates a variable frequency code index, code and number UCS per dimension.

It should UnsafeVariablePrepare be invoked with the same index variable.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

long Code Index 1, 2, ... index code of variable

⇒ BOOL IsMissing Yes / no missing

⇒ long Freq Frequency code file

⇒ Bstr Code Alphanumeric value code

⇒ long Count Number of elements in UCArray

⇒ long UCArray NUC array with each dimension

Return value:

false if one or more parameters are wrong, otherwise true

#### **2.3.2.10. UnsafeVariableClose**

Description:

Frees the memory reserved in UnsafeVariablePrepare

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

Return value:

false if parameter error, otherwise true

#### **2.3.2.11. GetTableUC**

Description:

Gives for a certain number of dimensions of the properties of a table.

Parameters:

long nDim Number of dimensions

long Index Sequence number 1, 2, ...

⇒ BOOL base table Is a base table (no subtable)

⇒ long NUC Number of UCs

⇒ long VarList nDim variable indices of the (sub) table

Return value:

true if the table with the requested index is found, false otherwise. If for Index = 1 false is returned then there are no tables with that number of dimensions.

#### **2.3.2.12. DoTruncate**

Description:

Reduces the number of codes of a variable by cutting of one or more positions at the right.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

long nPos Number of positions (1, 2, ...) that are truncated

Return value:

false if one or more parameters are wrong (eg no categorical variable, there is nothing left of a code), otherwise true

### 2.3.2.13. DoRecode

Description:

Reduces the number of codes of a variable by grouping several codes together. The RecodeString are separated by *newline characters* ( $\backslash r \backslash n$ ). Lines contain successively object code, ':' and source codes. Source codes that are too short are expanded through spaces in front of it. Codes may be surrounded by quotation marks ("3 -"), it may be necessary if the code ends with a space or a comma as a code or contains a dash.

A code is always treated as an alphanumeric. Therefore, the code "1a" is greater than "19" and the code "1" less than "11".

For example:

0 - 90	all codes less than or equal to 90
1: 91-500	codes from 91 to 500
2: 501 -	codes greater than or equal to 501
3: 11, "3", 512, 530-570, 930-970	multiple series are allowed, separated by ','

Parameters:

long	VarIndex	1, 2, ... n_var, index of variable
LPCTSTR	RecodeString	The specification string of the recoding
LPCTSTR	Missing1	The value of Missing1
LPCTSTR	Missing2	The value of Missing2
⇒ long	Error Type	Wrong type
⇒ long	ErrorLine	Line where the error occurred
⇒ long	ErrorPos	Position where the error occurred
⇒ Bstr	WarningString	Warning of overlapping sources, not mentioned codes, etc.

Return value:

false if one or more parameters are wrong (eg no categorical variable, there is nothing left of a code), otherwise true

### 2.3.2.14. ApplyRecode

Description:

Re-calculates, because e.g. a recoding, the (sub) tables. The results can be retrieved using:

- UnsafeVariable
- UnsafeVariablePrepare
- UnsafeVariableCodes
- UnsafeVariableClose

Parameters:

no

Return value:

no

### 2.3.2.15. GetMaxnUC

Description:

Calculates the maximum number nUC of the permanent (sub) tables.

Parameters:

no

Return value:

Maximum number nUC of the permanent (sub) tables.

### UndoRecode

Description:

Undoes a recoding. There cannot be a re-encoding applied on a recoding. A recoding is always applied on the basic code of a list variable. A new recoding can be done without first calling this function first.

The results can be retrieved using:

- UnsafeVariable
- UnsafeVariablePrepare
- UnsafeVariableCodes
- UnsafeVariableClose

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

Return value:

false if the parameter is incorrect, otherwise true

### 2.3.2.16. MakeFileSafeClearOptions

Description:

Invalidates the special processing options:

- SetRound
- SetCodingTop
- SetCodingBottom
- SetWeightNoise
- SetPramVar

Parameters:

no

Return value:

no

### 2.3.2.17. GetMinMaxValue

Description:

Gives for a numerical variable in the input file the detected minimum and maximum value.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

⇒ double Minvalue Minimum value

⇒ double MaxValue Maximum value

Return value:

VarIndex false if error, otherwise true

### 2.3.2.18. GetVarCode

Description:

Gives for a categorical variable for the code and PRAM-percentage.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

long Code Index 1, 2, ... index code of variable

⇒ Bstr Code Alphanumeric value code

⇒ long Prammingspercentage Percentage used for PRAM, -1 = Not applicable

Return value:

VarIndex or false if error code index, otherwise true

### 2.3.2.19. SetRound

Description:

Gives for a numerical variable the rounding base. If necessary the rounding base can contain decimals, eg 2.5. In that case give as round base 2.5 and nDec = 1.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

double RoundBase Rouding Base

long nDec Number of decimals of the rounding of basic

BOOL Undo False when put to refute false

Return value:

false if VarIndex is incorrect, RoundBase0 ≤, nDec <0, otherwise true

### 2.3.2.20. SetCodingTop

Description:

Gives for a numerical variable from which the value the value of that variable should be replaced by a fixed value, eg all values above 600,000 receive the text "> 600,000".

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

double TopLevel From this value, the value is replaced by TopString

LPCTSTR TopString Replacement

BOOL TopUndo False when setting the topcode, true when undoing it

Return value:

false if VarIndex is incorrect, otherwise true

### 2.3.2.21. SetCodingBottom

Description:

Gives for a numerical variable below which the value the value of that variable should be replaced by a fixed value, eg all values below 200,000 receive the text "≤ 200,000".

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

double BottomLevel Until this value is replaced with bottom string value

LPCTSTR BottomString Replacement

BOOL BottomUndo False when setting the bottomcode, true when undoing it

Return value:

false if VarIndex is incorrect, otherwise true

#### 2.3.2.22. SetSuppressPrior

Description:

Gives for a categorical variable the priority when applying local suppression, i.e. imputing "Missing1". It is possible that for a certain unsafe combination of variables, more than one candidate is eligible to be suppressed. If when calling MakeFileSafe with the option *with prior*, the variable with the lowest priority suppressed. If the values are equal, the first variable is taken, i.e. the smallest VarIndex.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable  
long Priority Value of the priority

Return value:

false if VarIndex is incorrect, otherwise true

#### 2.3.2.23. SetWeightNoise

Description:

Decreases or increases the value of a numeric variable with a percentage, randomly selected from the interval 1- *weight noise* till 1+ *weight noise*.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable  
double WeightNoise Percentage by which value is reduced / increased  
BOOL Undo False when applying true when undoing it

Return value:

false if VarIndex is incorrect, otherwise true

#### 2.3.2.24. SetPramVar

Description:

Specifies on which variable the following SetPramValue's apply. After this calls the function ClosePramVar to be called. This function checks if all codes of the pramvariabele have received a value.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable  
long BandWidth Bandwidth, N = -1  
BOOL Undo False when applying true when undoing it

Return value:

false if VarIndex is incorrect, otherwise true

#### 2.3.2.25. SetPramValue

Description:

Give the pram-percentage for a code. This percentage is the chance the code is not changed. If the code is not changed, it is changed into one of the other codes, each with a probability of  $(100 - \text{pram-percentage}) / n$ , where  $n$ :

- of codes - 1 if *bandwidth* is not applicable.
- *bandwidth* x 2

*bandwidth*

If BandWidth is applicable codes are only changed within a certain width. A code of 13 for index to be changed and has 3 bandwidth is changed to 10,

11, 12, 14, 15 or 16, each with an equal chance.

A *missing* will not be changed.

The value of the prampercentage ranges from 0 (always code changes) to 100 (code never changes).

Parameters:

long CodeIndex 1, 2, ... Index Code

long Value        Pram-percentage (0 - 100)

Return value:

false if CodeIndex is wrong or Value <0 or > 100, otherwise true

#### **2.3.2.26. ClosePramVar**

Description:

Completes the specification of SetPramValue

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

Return value:

false if VarIndex is wrong, or if not all codes are set, otherwise true

#### **2.3.2.27. GetBIRHistogramData**

Description:

Histogram gives information about the BaseIndividualRisk base . This is the natural logarithm of the result of that function.

Parameters:

long        TabIndex        1, 2, ... n\_tab, index of table

long        nClasses        Number of histogram classes

⇒ double   ClassValueLeft Array of *nClasses* + 1 with lower class, the latter contains the upper limit of the latter class

⇒ long       Frequency       Array of *nClasses* with BIR number per class

Return value:

false if TabIndex is wrong or some other silly error, otherwise true

#### **2.3.2.28. SetBIRThreshold**

Description:

Indicates the value above which an unsafe combination.

Parameters:

long    TabIndex 1, 2, ... n\_tab, index of table

double Threshold Limit above which a variable combination of unsafe

⇒ long nUnsafe    Number of unsafe combinations, with the Threshold

Return value:

false if TabIndex is incorrect, otherwise true

#### **2.3.2.29. MakeFileSafe**

Description:

Creates a safe file.

- If requested, calculates the entropy of categorical variables
- Makes a record description of the output file
- If there are PRAM variables, the tables of the PRAM variables are ignored for local suppression



- Takes into account that if a household variable changes, it changes in every household record in the same way
- Calculates the best combination of variables that must be set to *missing*, taking account of entropy, or weights, when creating a safe record

If both WithPrior and WithEntropy are false nothing will be suppressed.

Parameters:

LPCTSTR	FileName	Name of the safe file
BOOL	WithPrior	Use Priorities
BOOL	WithEntropy	Use entropy
long	HHIdentOption	Option to use to HHIdent
BOOL	RandomizeOutput	Writes records in random order to the safe file; Only for fixed format data

Meaning HHIdentOption:

0. There is no household
1. Keep Household Variables consistent and do not change HHIdent
2. Keep Household Variables consistent and change HHIdent into a serial number
3. Keep Household Variables consistent and remove HHIdent Household Variables

Maintain consistency means that if in at least one record in a household a variable is changed into *missing* that variable is set to missing in all other records of that household.

Return value:

false if something went wrong, otherwise true

### 2.3.2.30. GetVarProperties

Description:

Provides for a variable position in the safe file, the number of times the variable is suppressed and the entropy.

Parameters:

long	VarIndex	1, 2, ... n_var, index of variable
⇒ long	StartPos	Start position in safe file
⇒ long	nPos	Number of Positions
⇒ long	nSuppress	Number of suppressions
⇒ double	Entropy	Entropy variable, -1 if N
⇒ long	BandWidth	Bandwidth entropy, -1 if N
⇒ Bstr	Missing1	Alphanumeric value missing1, blank if not applicable
⇒ Bstr	Missing2	Alphanumeric value missing2, blank if not applicable

Return value:

VarIndex false if error, otherwise true

### 2.3.2.31. BaseIndividualRisk

Description:

Calculates the individual risk of records based on the frequency of key variable combination (fk) and the sum of the weights of (Fk). Only important if a BIR-table is specified (see SetTable ).

If fk is 0 the result is 0, and  $fk = Fk = 1$  then 1.

This function is called by MakeFileSafe . The reason for this function as an export function to include is the fact that it is nice to have a complicated calculation process disposal.

*See Strategy for the Implementation of individual risk methodology to write-ARGUS: independent (and hierarchical) units:*

Maurizio Bianchi and Alessandra Capo Lucarelli, ISTAT, MPS / D, Via C. Balbo 16, 00184, Roma, Italy

Deliverable No: D1-1.2 May 30, 2001 (Third draft)

Parameters:

Long      fk    0, 1, 2 ... frequency of key variable combination

double    Fk    total weights of the records with this key variable combination

⇒ double Risk The risk of this combination,  $0 \leq \text{Risk} \leq 1$

Return value:

false if parameters are wrong, otherwise true

possible errors:

- fk < 0
- Fk < 0

### 2.3.2.32. CleanAll

Description:

Deletes all specified data and displays all the reserved memory.

Also called by SetNumberVar

Parameters:

no

Return value:

no

## 3. $\tau$ -ARGUS

### 3.1. General overview

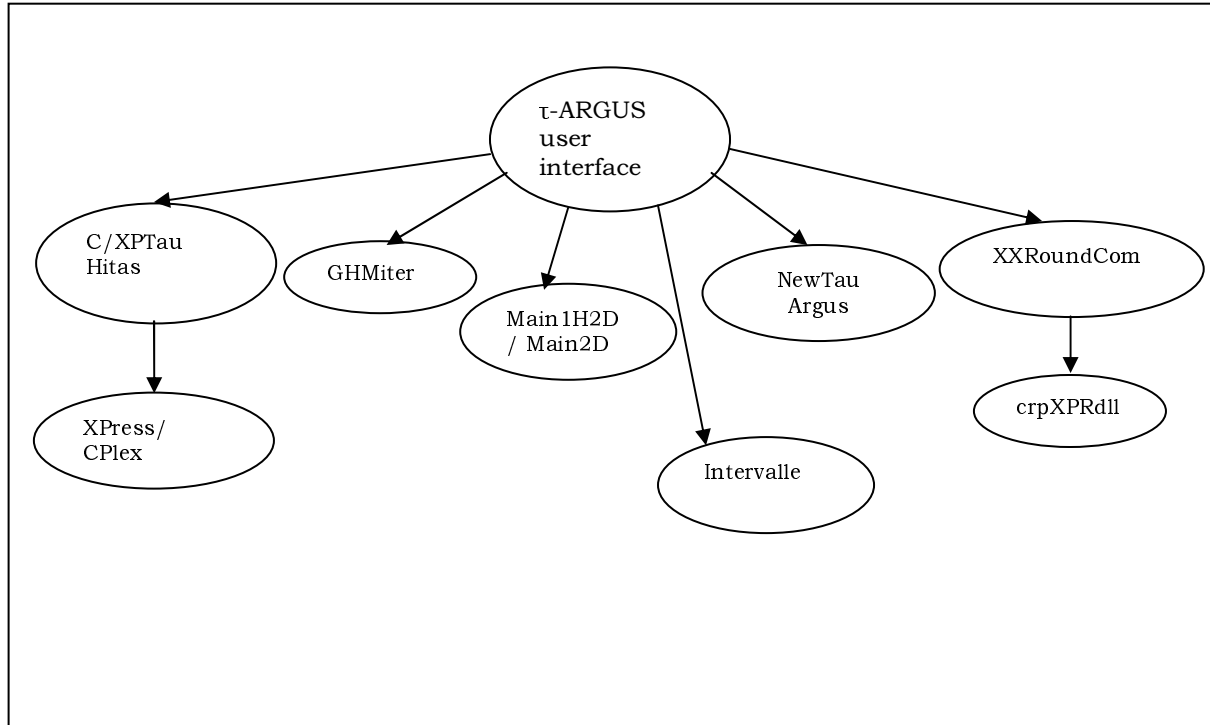
$\tau$ -ARGUS has been developed to protect tabular data.

It started in the SDC-project with a first version, capable of protecting non-hierarchical tables. Also the first implementation of the optimisation routines by Salazar was included.

The general structure of  $\tau$ -ARGUS is similar to  $\mu$ -ARGUS. A Visual Basic user-interface controls the flow of the program and other routines around this. However during the development of  $\tau$ -ARGUS in many situations the quickest solution was to add procedures to the VB-part of  $\tau$ -ARGUS. This because often a qualified C++-programmer was not always available. Therefore the VB-part is much bigger than originally planned. Some routines have been written as C++ dlls others are separate stand alone programs, called from the VB-part of  $\tau$ -ARGUS.

- **NewTauArgus.dll**
  - Basic data manipulations
  - Computing tables from microdata
  - Global recoding
  - Applying primary sensitivity rules
  - Writing input files for the other dlls and exes.
  - Processing the output of the other parts-
- **CPTauHitas.ocx**  
modular protection of the table produced by NewTauArgus.  
Breaks down the table into smaller non-hierarchical sub-tables, protects each sub-table and does all the needed backtracking. In this dll also JJ's routines are included.
- **XPTauHitas.ocx**  
Same as CPTauHitas but for using XPress as a solver. Due to peculiarities in JJ's part of the code this was the only solution.
- **CheckTime.exe**  
A small program called by C/XPTauHitas, when the time-limit has exceeded. A call back to the user-interface was too complex.
- **main1H2D.exe**  
The network solution of Cell suppression as developed by Jordi Castro. This is the version for a two-dimensional table with only one hierarchy.
- **main2D.exe**  
The network solution of Cell suppression as developed by Jordi Castro. This is the version for a two-dimensional table with no hierarchy.
- **Ghmitter4.exe**  
The German Hypercube method. This solution has been developed by the Statistical Office of Nordrhein-Westfalen. It is a separate Fortran program
- **Intervalle.exe**  
The audit routine. This is a separate Delphi program.
- **crpXPRdll.dll**  
The rounding procedure included in  $\tau$ -ARGUS. It was developed by JJ Salazar. This C-dll is embedded in the XXRoundCom.
- **XXRoundCom.dll**  
The wrapper around the crpXPRdll to link it to tau.
- **CheckTimeRound.exe**  
A similar program as TimeCheck to interface when the time limit has exceeded.

### τ-ARGUS Structure



## 3.2. Description of the individual modules

### 3.2.1. τ-ARGUS VB user-interface

The τ-ARGUS user-interface is build using a set of VB-forms and a set of VB-modules. The forms contain the code for the different windows and its corresponding routines, while the modules contain only more general functions and data structures.

A small number of forms and modules are used both by μ-ARGUS and τ-ARGUS.

The τ-ARGUS main menu structure

File	Specify	Modify	Output	Help
Open Microdata	Metafile	Select Table	Save Table	Contents
Open Table	Specify Tables	View Table	View Report	News
Open Tableset		Linked Tables	Generate Apriory	Options
Open Batch process			Write Batch file	About
Exit				

Each entry in the menu opens a new window/form. So the menu structure can also be presented in the corresponding forms

File	Specify	Modify	Output	Help
<i>dlgOpenFile.frm</i>	<i>frmTableMeta.frm</i>	<i>dlgSelectTable.frm</i>	<i>frmSaveTable.frm</i>	Opens the Help system
<i>frmOpenTable.frm</i>	<i>dlgSpecifyTables.frm</i>	<i>frmViewTable.frm</i>	<i>frmViewReport.frm</i>	<i>frmViewReport.frm</i>
<i>frmOpenTableSet.frm</i>		<i>frmLinkedModular.frm</i>	<i>frmMakeAP.frm</i>	<i>frmOption.frm</i>
Standard File open box			Standard File open box	<i>frmAbout.frm</i>

Exit				
------	--	--	--	--

In the remainder we will describe the functions and procedures in the various forms and modules. However trivial smaller self-explanatory functions and procedures have been omitted.

**τ-ARGUS forms** in alphabetical order

### 3.2.1.1. dlgGHMiterSpec.frm

Just asks a few options for the hypercube method. A parameter on the inferential disclosure protection and some info about the memory model. In general the normal model is OK. Only for very large, unstructured tables a large model is needed. However the performance will be less.

### 3.2.1.2. dlgMinRoundBase.frm

Min. required roundingbase = 7596

Rounding base:

Number of steps allowed:

Max computing time:  min.

☐ Partitions

Stopping Rule

☐ First RAPID only

☐ first feasible only

☒ Optimal solution

Partitions

8 subtables

18 cells per subtable

OK

Cancel

Asks the parameters for the rounding procedure. The rounding base should be at least larger than minimum specified. This is the largest protection interval.

If partitions is selected then the table is broken down over the first spanning variable/layer. Each block is then rounded and the total-layer is computed from the results. Partitioning is only needed for tables over 100K cells

The JJ-file split into parts here.

- Public Function TestBatchRound(TabNo As Long)  
Checks in batch if the table is not too large
- Function SplitJJ() As Long  
This function splits the JJ-file for partitioning
- Function JoinRounded(Part As Long)  
The function brings the rounded blocks together

### 3.2.1.3. dlgSelectTable.frm

Explanatory variables	Resp. var
Size, Region	Var2
Region, Year	Var2

Cancel

OK

If more than one table has been specified, the active table can be selected.

The global variable “Selectedtable” is set.

#### 3.2.1.4. dlgSpecifyTables.frm

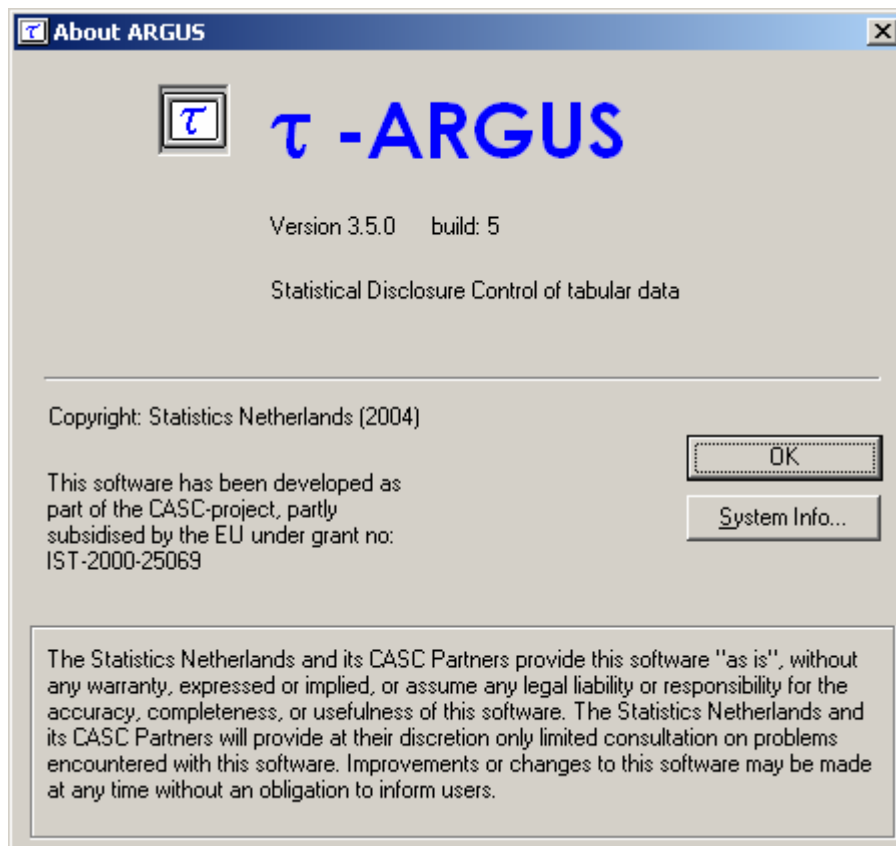
The whole process of specifying table is controlled here. This is only available when the tables are computed from microdata.

On opening the two listboxes (explanatory variables and cell items) are filled.

- Sub AddToTableStruct()  
called by 'btnSelectTable\_Click' to store the information in the 'TableSetStruct'
- Private Sub btnDeselectExplanatory\_Click()
- Function TestDubbel() As Boolean  
Obsolete. Specifying a table twice is now allowed and could be used for the linked tables procedure.
- Private Sub btnSelectTable\_Click()  
When a table has been specified a lot of checks are performed, before storing the table via AddToTableStruct.
- Private Function GetToken(S As String) As String  
Gets a token, should be replaced by a call to the general function in ARGUSfunctions
- Private Sub CancelButton\_Click()  
No tables are computed and the forms will be closed. The specified tables however will remain to be stored.
- Public Sub LaadRegister()  
Many parameters chosen will be stored in the registry. Just to present the familiar parameters. This function retrieves them
- Sub BewaarRegister()  
Partner of the previous function. This function stores them
- Private Sub Form\_Activate()  
Fills the listboxes

- Private Sub OKButton\_Click()  
Will start the process of storing all the information and the actual table computation process. Calls ComputeTables
- Function ComputeTables() As Boolean  
Prepares for handing over the information to the  $\tau$ -ARGUS kernel.
  - the name of the datafile
  - file type (fixed of free or SPSS)
  - the number of variables
  - for each variable the details
 The kernel will explore the file and build the codelists.  
 For each table the structure is passed and the table safety information. Two calls are needed given a restriction on the number of parameters.

### 3.2.1.5. frmAbout.frm



Just shows the about box



### 3.2.1.6. frmChangeView.frm

Change View

Size  
Region

>  
<

Column

Max. level shown 2

Row

Max. level shown 2

Number of decimals shown 2

Cancel OK

Called from the table presenter to change the presentation of a table.

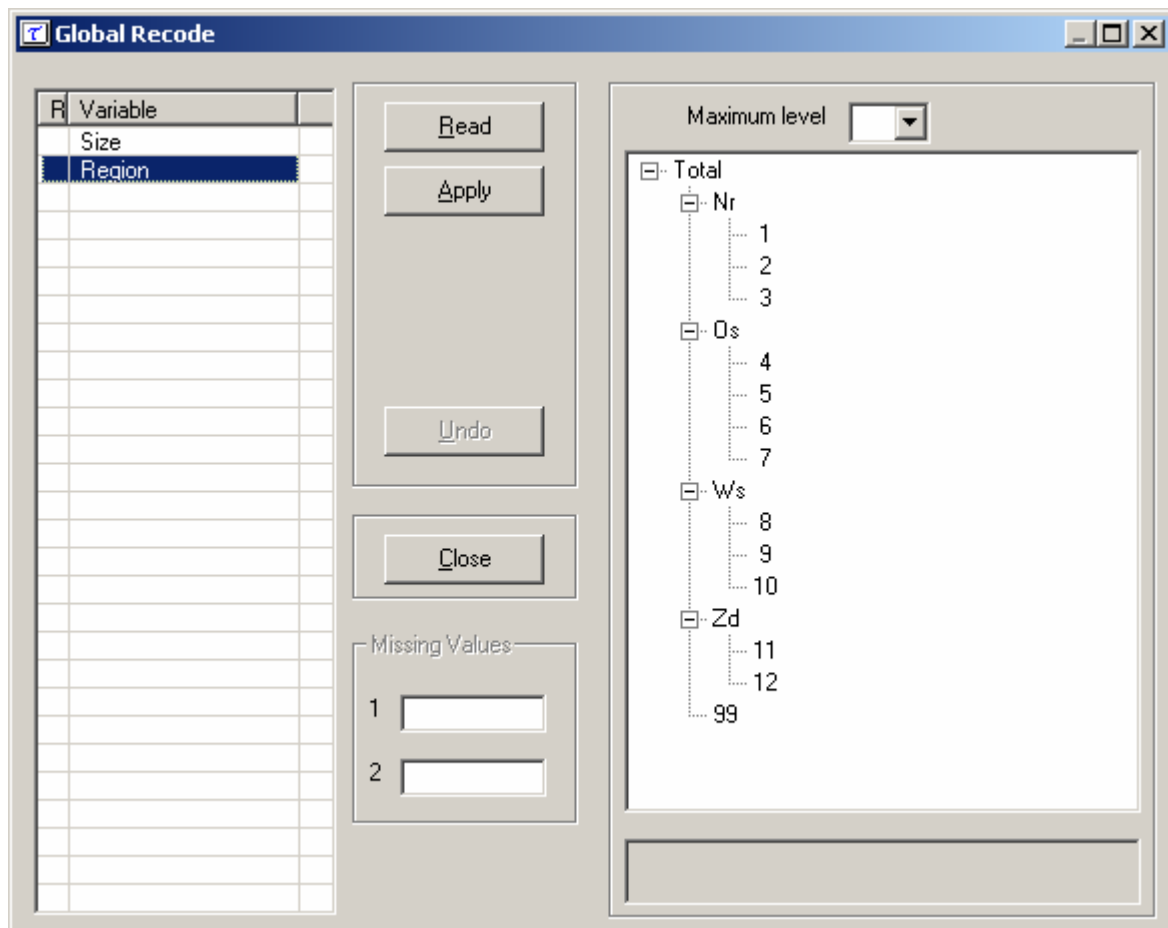
The row and column variable can be chosen and also the maximum level of the hierarchical tables. Also the number of decimals shown.

Via 'SelectedRow' and 'SelectedColumn' the information is passed back.

The number of decimals are set via the variable 'frmViewTable.Decimals'

### 3.2.1.7. frmGlobalRecode.frm

The screenshot shows the 'Global Recode' dialog box in SPSS. On the left, a list of variables includes 'R', 'Variable', 'Size', and 'Region', with 'Size' currently selected. The right side of the dialog contains several controls: 'Read', 'Apply', 'Undo', and 'Close' buttons; 'Missing Values' input fields for values 1 and 2; a 'Codelist for recode' input field with a dropdown arrow; and a 'Warning' section with a large empty text area. The window title bar at the top reads 'Global Recode'.



This form has two appearances, one for non-hierarchical variables and one for hierarchical variables. In this case a treeview is presented.

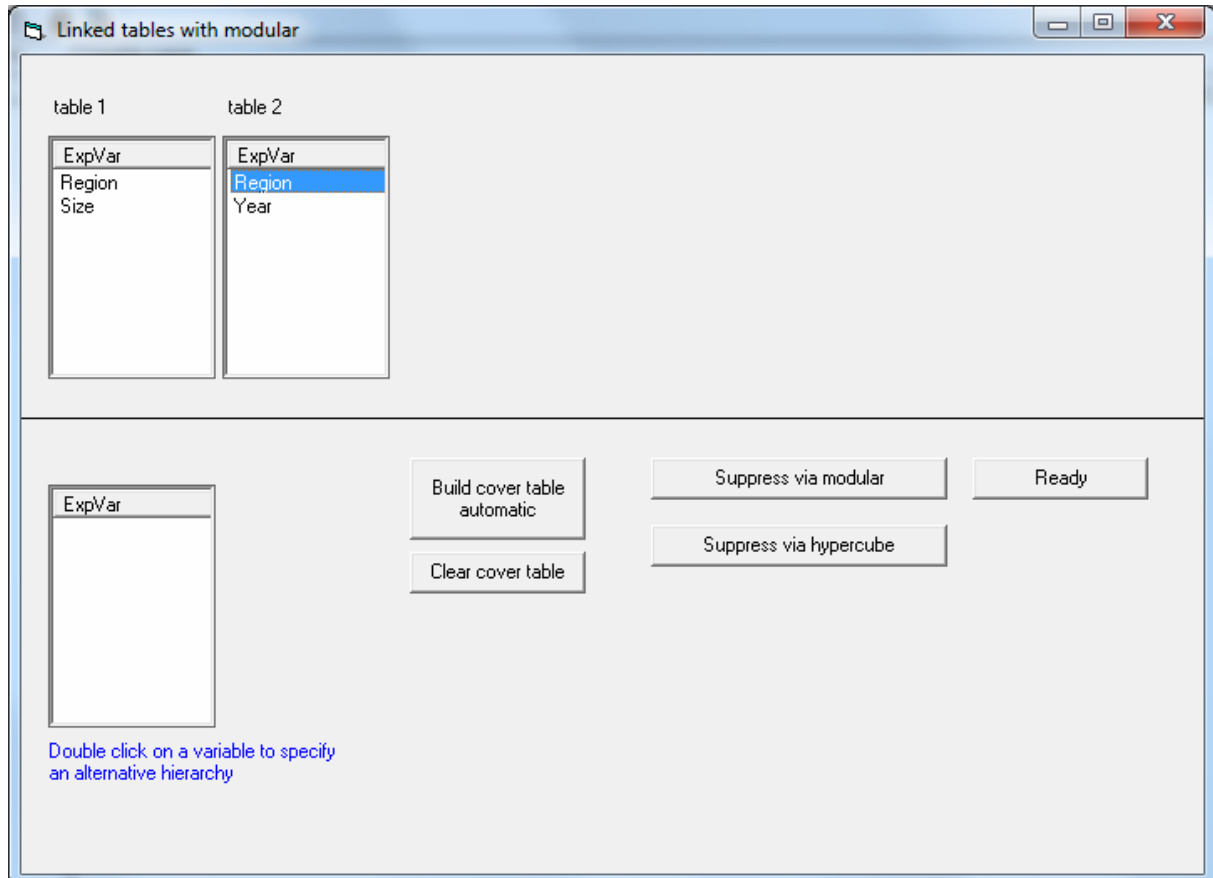
- Private Sub RemoveQuote(Hs As String)
- Public Function LeesFeitelijkRecodeFile(Fn As String, RCode As String, Mis1 As String, Mis2 As String, CodeList As String) As Boolean  
Reads a recode file, called from 'LeesRecodeFile'
- Sub LeesRecodeFile(Fn As String)  
Reads a recode file for non-hierarchical variables
- Sub BewaarRecodeFile(Fn As String, Ask As Boolean)  
Stores the above mentioned file
- Sub UpdateLstVariables()  
Updates the left pane, color etc
- Private Sub VerwerkKnoopLevel(ii As Long)  
Transfers info on the 'closed' nodes in a tree to the kernel
- Private Sub cmdApply\_Click()  
Splits directly for hier., and non-hier. variables  
Transfers the recode info to the kernel
- Sub BuildTree(Optional MaxLevel As Long = 0)  
Builds the treeview for a hier. variable with a maximum depth
- Private Sub cmdTruncate\_Click()  
For a hier. variable with a hierarchy form the codes itself (like NACE) truncate is a recode option. This function also transfers it to the kernel.
- Private Sub cmdUndo\_Click()  
Instructs the kernel to go back to the original code list
- Public Function ApplyTree(Fn As String, VarNo As Long) As Boolean  
Transfers the information of the current tree as a recode to the kernel
- Private Sub Form\_Load()
- Sub ZoekVar(Optional OpenForm = False)  
Organises the windows if another variable is selected in the left pane

- Private Sub Form\_Unload(Cancel As Integer)  
Instructs the kernel to apply the recodes and closes this window

### 3.2.1.8. frmInfo.frm

Shows an information file

### 3.2.1.9. frmLinkedModular.frm



Several command-buttons directly call another function. This to prepare for batch-version of linked tables. This way the code can be called easier.

- Function PrepareLinked() As Boolean  
Prepares the windows/panes
- Function ControleCodeLists() As Boolean  
Checks whether the codes of the n on-cover variables can be found in the cover table
- Function ExportTables() As Boolean  
Exports the individual tables in the cover-format. Rearranges the order of the variables and inserts a total-code for the cover variables not present in the current table Adjusts also the RDA-file.
- Sub LeesVar(K As Long, R As Long, VarNo As Long)  
Reads from the metafile K the metadata from var R
- Sub copyStaartRDA()  
Reads the remainder of a metadata file
- Function RunMotherTable() As Boolean  
Starts a batchrun for the cover table
- Function LeesResultsBack() As Boolean  
Converts the output of a the cover table into individual apriori-files and applies them to the individual tables.
- Public Function BuildCover() As Boolean  
Builds the cover table. For each variable he searches for the longest coelist and takes that var for the cover table.

- Private Sub cmdBuildCover\_Click()  
Calslabove
- Public Function ProtectLinked() As Boolean  
Calls the different steps in the protection process
- Function AdjustEingabe(n As Long) As Boolean  
Adjusts the EINGABE file (part of the hypercube)
- Function PlusNul(n) As String
- Function AdjustSteuer(n As Long) As Boolean  
Adjust the STEUER fiel for the hypercube
- Function ProtectLinkedHypercube() As Boolean  
Writes all the input files for the hypercube and calls the adjustment procedure
- Private Sub cmdGHMiter\_Click()  
Calls the linked tables version of eth hypercube

### 3.2.1.10. frmMain.frm

Variable	dim 1	dim 2
Size	0	9
Region	0	9

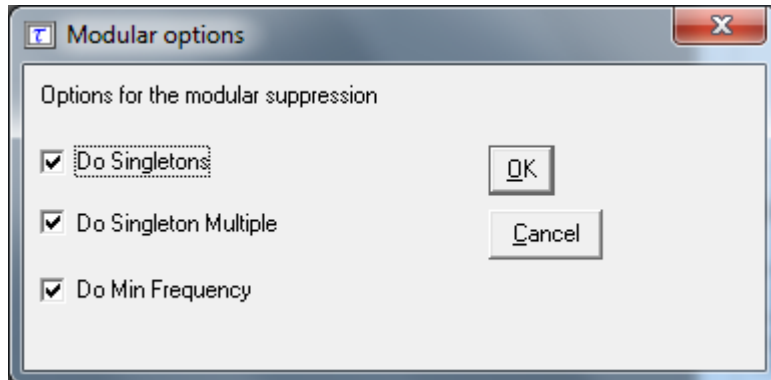
Code	Label	Freq	dim 1	dim 2
	Total	42723	0	0
·2		9	0	4
·4		5	0	4
·5		20002	0	0
·6		8831	0	0
·7		5498	0	0
·8		4594	0	0
·9		3779	0	1
99		5	0	0

The main program window. Contains many smaller functions calls to invoke the procedures of all the sub-windows. Also many smaller functions are needed to catch the fire-events from the DLLS linked to  $\tau$ -ARGUS.

We just mention a few.

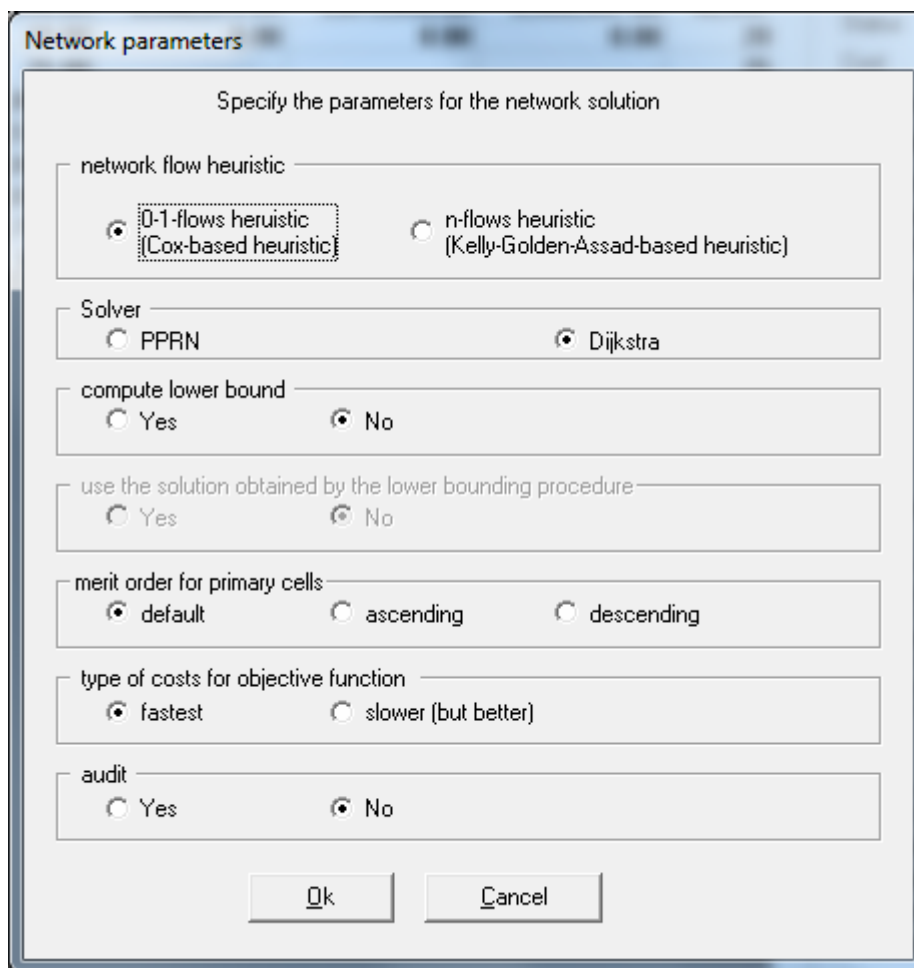
- Sub DiscrepancyAndTime()  
Displays a time difference. Should be in an other standard module
- Sub ActivateMenuOptions()  
Controls which part of the menu is active. The controlling variable is 'Voortgang'.
- Sub RefreshPanels()  
Refreshes the left and right pane

### 3.2.1.11. frmModularOptions.frm



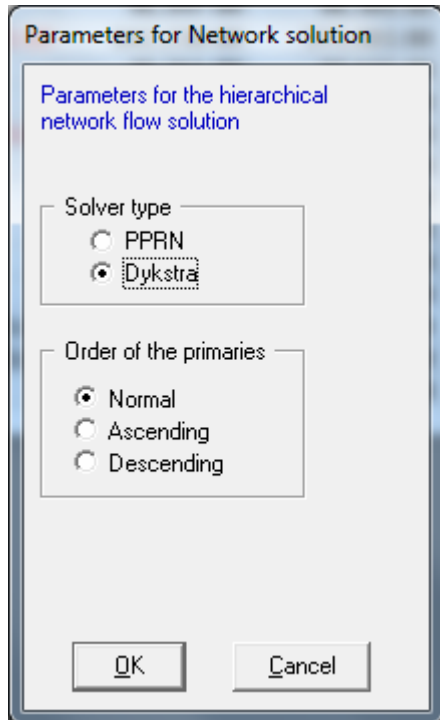
Ask for the 3 singleton options. Also stores them in the registry. Used for FullJJ now too.

### 3.2.1.12. frmNetwork.frm



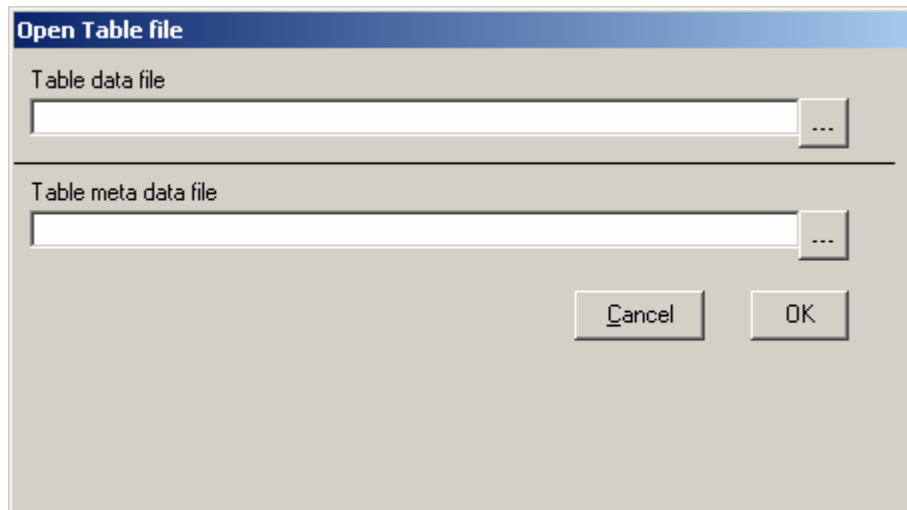
Allows to specify all the parameters for the non-hierarchical2-dim network solution. Details can be found in the documentation of this procedure

#### 3.2.1.13. frmNetwork2D1H.frm



The 2-dim-network solution with one hierarchy (the first) asks only a subset of the parameters.

#### 3.2.1.14. frmOpenTable.frm



Asks for the name of the table file and the meta data file.

### 3.2.1.15. frmOpenTableSet.frm

The screenshot shows a Windows-style dialog box titled "Open Table Set". At the top, there are two input fields: "Table file:" and "Meta data file:". Each field consists of a text box followed by a small button with three dots (...). Below these fields is a central "Add" button. The main body of the dialog is split into two vertical panes. The left pane is labeled "Tables" and the right pane is labeled "Metafiles". Both panes contain large, empty rectangular areas, likely for displaying a list of files. At the bottom of the dialog, there are two buttons: "Clear" on the left and "OK" on the right.

For the linked tables all the table files and their corresponding meta datafiles.



### 3.2.1.16. frmOption.frm

**Options**

**Cell foreground colours**

- Safe
- Safe (manual)
- Unsafe
- Unsafe (request)
- Unsafe (Freq)
- Unsafe (Zero cell)
- Unsafe (Singleton)
- Unsafe (Singleton) (manual)
- Unsafe (manual)
- Protected
- Secondary
- Secondary (from man.)
- Empty (non-struct.)
- Empty
- AuditError (Primary)
- AuditError (Secondary)

Reset default colours

**Cell background colours**

- hier. level 0
- hier. level 1
- hier. level 2
- hier. level 3
- hier. level 4
- hier. level 5
- hier. level 6
- hier. level 7
- hier. level 8
- hier. level 9
- hier. level 10

Reset default colours

Max time per table for Modular solution: 1 min

Logfile name: [Browse]

**Specify solver information**

☐ No solver available

☒ Xpress

☐ Cplex

Cplex licence file: [\\RDV1PF\Projecten\CASC\Anco\TauArgus\VB\access(1).il] [Browse]

OK

Allows to specify all the colours used in the table window. The type of the solver (Xpress or Cplex) can be selected.

All parameters specified here are stored in the registry and used for future sessions as well.

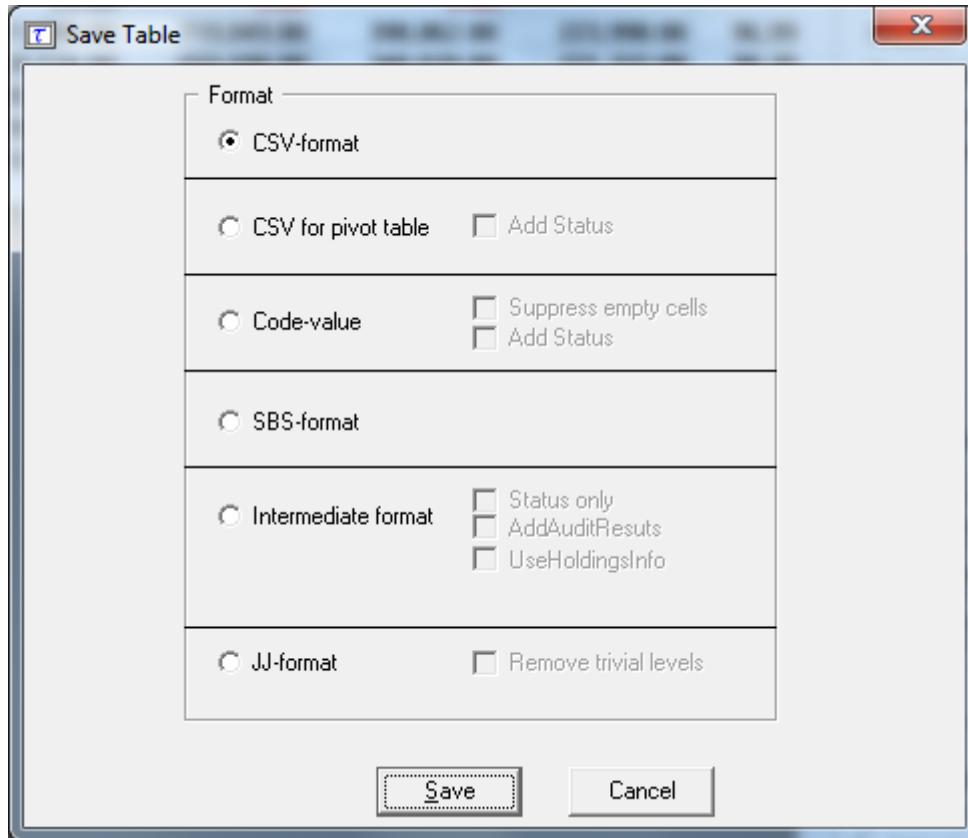
### 3.2.1.17. frmReadHistory.frm

Type	Correct	InCorrect	
UnKnown	0	0	
Status	3	0	
Cost	0	0	
Apriory Bound	0	0	
Prot.level	1	0	

The apriori file was originally called the history file. That explains also the extension.

The real work is done in the procedure “VerwerkHist”

### 3.2.1.18. frmSaveTable.frm



The current table can be saved in various formats.

- Sub PrintTree(VarNo As Long)  
Prints a code list as a simple tree in HTML
- Private Sub PrintTabel(VarNo As Long)  
A procedure to print a table in HTML
- Sub prDOM(j As Long)  
Print the dominance info in HTML
- Sub PrPQ(j As Long)  
Prints the P%-rule info in HTML
- Sub WriteReport(SuperCross As Boolean)  
This writes the whole HTML-report. It uses the previous 4 routines for smaller tasks.  
If called from Superctross only a small part of the report is generated
- Private Sub btnCancel\_Click()
- Sub ControleerJJFormat(Filename As String)  
Obsolete
- Private Sub cmdSaveFile\_Click()  
The variable 'Versie' controls the type of output. The parameters are stored together in 'Param', a so called add-up coding. The next procedure does the actual work. This allows for calling this procedure from batch
- Function WriteTabel(TabNo As Long, Versie As Long, Param As Long, FName As String) As Boolean  
The actual write procedure.  
For each version a procedure in the kernel is called, except for intermediate format
- Private Sub Form\_Load()
- Private Sub optFormat\_Click(Index As Integer)  
Organises the activity of certain parameter fields, depending on the option chosen

#### **3.2.1.19. frmSpecifyMeta.frm**

This form has different appearances for micro data and tabular data. The first window is for micro data the second one for tabular data.

**Specify metafile**

Fixed format ▼

Year  
IndustryCode  
Size  
Region  
Wgt  
Var1  
Var2  
Var3  
Var4  
Var5  
Var6  
Var7  
Var8  
Request

**Attributes**

name: Year ☒ explanatory variable

starting position: 1 ☐ response variable

length: 2 ☐ sample weight variable

decimals: 0 ☐ holding indicator

☐ request protection

☐ distance for suppression weight

**Codelist**

☒ automatic

☐ codelist filename

Missings: 1: 99 2:

☐ hierarchical

☐ Levels from microdata

☐ Levels from file

Leading string

New ↑

Delete ↓

Cancel OK

Version for microdata

#### Version for tabular data

In case of using a SPSS-data file a special listbox is shown, where the variables of interest from the current run can be selected. They will then be saved in a fixed format file (in TEMP) and then  $\tau$ -ARGUS will continue as if a fixed format micro data file is used. However the meta data cannot be changed any more, because the structure is implied by SPSS..

- Function CheckMeta() As Boolean  
Checks a.o. whether variables do not overlap and whether cat. variables have a missing value.
- Sub TextBoxEnable(O As Object, B As Boolean)  
Organises the activity of certain text-boxes
- Sub CheckStatusVelden()  
Organises the visibility and activity of parameter fields and options
- Sub CheckSubFrames()  
Resizes the frames. E.g. for free format data the fields separator should be visible. So the var.list begins lower.
- Sub FillLstVar()  
Lists the var-list box
- Function Bewaar(Ind As Long) As Boolean  
Saves the information of a variable in the data structure 'MetaDataStruct'.  
Also checks whether the variable has been updated. If so in the end he will ask to save the changes.
- Sub Update(Ind As Long)  
Fills all the fields when a new variable is selected.
- Private Sub btnCancel\_Click()  
Stops without saving.

- Private Sub btnDelete\_Click()  
Deletes a variable
- Private Sub btnNew\_Click()  
Creates a new variable
- Private Sub btnOK\_Click()  
Will create a new RDA-file if the meta has been changed and the users asks for it
- Sub SchrijfRDA(RDAFile As String)  
Called from the SPSP-routines to write a new RDA-file
- Sub AdjustForFileType()  
Rearranges the window allowing for different file types (Fixed, free, SPSS).
- Private Sub cmbFileType\_Click()  
Calls above
- Sub MoveUpDown(PlusMin As Long)  
Rearranges the order of the variables. The order is essential in case of free format
- Private Sub cmdGenerateSPSS\_Click()  
Sub AddSPSSVariables()
- Private Sub cmdSPSSCancel\_Click()
- Private Sub cmdSPSSOK\_Click()  
Generates a MetaDataStruct from tehSPS-meta data info
- Private Sub Form\_Activate()  
Initialises the window
- Private Sub lstVariables\_Click()  
Selects an other variable and updates the windows
- Private Sub optAutomatic\_Click()  
Selects that no codelist is used
- Private Sub optCodelist\_Click()  
Selects that that a codelist is available for this variable.

### 3.2.1.20. frmSummary.frm

Summary for table no: 1

Explan. Var	# Codes	Status	Freq	# rec	Sum Resp	SumCost
Size	9	Safe	110	256295	101073868.04	101073868.04
Region	18	Safe (manual)	0	0	0	0
		Unsafe	9	43	12013	12013
		Unsafe (request)	0	0	0	0
		Unsafe (Freq)	0	0	0	0
		Unsafe (Zero cell)	0	0	0	0
		Unsafe (manual)	0	0	0	0
		Protected	0	0	0	0
		Secondary	0	0	0	0
		Secondary (from man.)	0	0	0	0
		Empty (non-struct.)	0	0	0	0
		Empty	43	0	0	0
		Total	162	256338	101085881.04	101085881.04

Resp. Var: Var2  
Shadow Var: Var2  
Cost Var: Var2

**Not yet protected**

OK

Is called from the table presentation screen.

Gives an overview of the status oif the current table.

- Private Sub cmdOK\_Click()  
Closes the window

- Private Sub Form\_Activate()  
Fills all the info, when the window is invoked  
Retrieves the information from the kernel via GetCellStatusStatistics
- 

### 3.2.1.21. frmTableMeta.frm

This window is used when tabular data is used. Both in case of a single table or a set of linked tables.

- Private Sub Afsluiten()  
Closes the window and stores parameters in the registry
- Private Sub OrganizeActivity()  
Activates certain options when appropriate. Because the table data can have different information certain options are not always available.
- Private Function Opbergen() As Boolean  
Creates an entry in the "TableSetStruct" and stores all the information there
- Private Sub cmdOK\_Click()  
Will start the process of saving the information and dreading the table.  
Calls:
  - Opbergen
  - PrepareTableData(frmTableMeta) (only if single table)
  - LeesOneTabel
- Function Controles()  
Checks the consistency of the parameters selected.
- Sub Additivity\_Report()  
Obsolete
- Private Sub Form\_Activate()  
Initialises the window, Activates appropriate options etc
- Public Function ControleerTableMeta(NT As Long)  
Checks whether enough information is available for applying dom. rule, P% rule etc.



### 3.2.1.22. frmViewTable.frm

The screenshot shows the 'frmViewTable.frm' window. The title bar reads 'Table: Size x Region | Var2'. The main area contains a table with columns labeled 'tot', '2', '4', '5', '6', and '7'. The rows are labeled with codes like '.Nr', '.. 1', '.. 2', etc., and values are displayed in various colors (black, red, green). To the right of the table is a 'Cell Information' panel with fields for Value, Status, Cost, Shadow, # contributions, Top n of shadow, Holding level, and Request. Below this is a 'Change status' panel with buttons for 'Set to Safe', 'Set to Unsafe', 'Set to Protected', 'Set Cost', 'A priori info', 'All Non-StructEmpty', and 'Recode'. At the bottom right is a 'Suppress' panel with radio buttons for 'HyperCube', 'Modular', 'Network', 'Optimal', and 'Rounding', along with buttons for 'Round', 'Undo Rounding', and 'Audit'. At the bottom left are checkboxes for '3 dig. separator' and 'Output View', and buttons for 'Select Table', 'Change View', 'Write table', 'Table Summary', and 'Close'.

	tot	2	4	5	6	7
tot	16,847,646.84	20.00	25.00	2,711,808.00	2,320,534.00	2,505,000.00
- .Nr	4,373,664.00	5.00	5.00	719,049.00	659,680.00	688,000.00
.. 1	1,986,129.00	5.00	5.00	398,062.00	348,039.00	354,000.00
.. 2	1,809,246.00	0.00	-	223,990.00	221,332.00	241,000.00
.. 3	578,289.00	-	-	96,997.00	90,309.00	92,000.00
- .Os	3,703,896.00	15.00	5.00	642,238.00	515,003.00	534,000.00
.. 4	124,336.00	5.00	-	36,311.00	32,132.00	25,000.00
.. 5	526,279.00	-	-	93,589.00	94,957.00	110,000.00
.. 6	2,234,995.00	10.00	5.00	345,803.00	251,358.00	251,000.00
.. 7	818,286.00	-	-	166,535.00	136,556.00	140,000.00
- .Ws	4,576,115.84	-	-	648,972.00	543,570.00	663,000.00
.. 8	485,326.00	-	-	63,767.00	75,442.00	87,000.00
.. 9	3,664,559.84	-	-	537,911.00	430,851.00	519,000.00
.. 10	426,230.00	-	-	47,294.00	37,277.00	61,000.00
- .Zd	4,193,971.00	-	15.00	701,549.00	602,281.00	618,000.00
.. 11	2,752,743.00	-	15.00	488,613.00	392,395.00	363,000.00
.. 12	1,441,228.00	-	-	212,936.00	209,886.00	254,000.00
.99	-	-	-	-	-	-

This is the main window controlling all activities around a table.

- Sub Showframe(Number As Long)  
At the bottom of the window different frames can be shown, showing the progress of different processes.
- Sub TooLarge(B As Boolean)  
If the table is too large to be shown, this will be shown, but the remaining functionality remains in tact
- Public Function GrandTotal(Selectedtable As Long) As Double  
Computes the grand total by a call to 'GetTableCell'
- Sub CheckHitas()  
Checks whether certain suppression solutions are available given the constraints
- Sub FillComboBoxes()  
If more than 2 dimensions only one layer is shown. The remaining variables are shown in some combo-boxes at the top
- Sub SetDimArray(R As Long, K As Long)  
dimarray has the codes of the current cell. Used for retrieving info from the kernel
- Sub SetCellColor(ColorNr As Long)
- Sub KleurCel(St As Long, AuditOK As Long)
- Sub BuildRowColCodeList()  
Builds the code list from the current row and column given the detail shown.
- Sub CheckUnfoldRow()  
After a call to changeView the level of the hierarchy shown is set. This routine handles it further

- Sub CheckUnfoldCol()  
Similar as above
- Sub FillText(R As Long, K As Long, St As String)
- Sub BuildTable(ShowRow As Long, ShowCol As Long)  
Builds the whole table matrix, First the row and column with the codes, then the interior (cell values)
- Private Sub btnChangeView\_Click()  
Calls frmChangeView and processes the new view.  
'StrafBankVar' are the variables not shown in the current view but visible in the combo boxes at the top
- Private Sub btnClose\_Click()  
Closes the window and stores the last used method in the registry.
- Private Sub btnRecode\_Click()  
Calls the frmGlobalRecode and displays the new table
- Sub SetSuppressButtons()  
activates the now relevant suppression-buttons.
- Sub schrijfTABELLE(Fn As String, Singleton As Boolean, RespVar As Long, TabNo As Long, Linked As Boolean, CoverDim As Long)  
Write TABELLE for the hypercube method
- Function zoekILM() As String  
Searches for the Cplex licence file. If the default file 'access(1).ilm' in the τ-ARGUS-directory is found, this one is chosen. The result is stored in the registry
- Public Sub OpschonenEingabe(Temp As String, APriori As Long, Singleton As Boolean, RV As Long, TabNo As Long, UseStatusOnly As Boolean)  
Some final checks on the EINGABE as generated by the kernel. E.g. the frequency is set to a higher value f on the the given status is used. Avoids the singleton procedures.
- Private Sub btnSuppress\_Click()  
Initiates the suppression procedures, (incl. rounding)
- Function VraagMinTabVal(Soort As Long, TabNo As Long, XMin As Double, XMax As Double) As Boolean  
Asks the real minimum value (including the lower bound. By default 1.5 x Minimum cell value
- Function SchrijfSteuer(STab As Long, Singleton As Boolean, Nummer As String, Temp As String) As Boolean  
Writes STEUER for the hypercube. Partly via a procedure in the kernel
- Sub CleanFilesGHMiter(Temp As String, TabNo As Long)  
Cleans the temp-dir fro files generated by the hypercube. The hypercube does not like those files to be present.
- Function DraaiGHMiter(Singleton As Boolean, Temp As String) As Long  
Runs the hypercube (a stand-alone program). First moves the current dir to temp as the hypercube wants to write in the current dir.
- Function TestProto003(TabNo As Long) As Boolean  
The hypercube writes a file TestProto003 if there are some frozen cells. This procedure tests whether this file is empty
- Sub SchrijfFrozenCells(Singleton As Boolean, NDim As Long, TabNo As Long)  
Writes a report on the frozen cells and shows it
- Function RunGhMITER(Tabnummer As Long, Singleton As Boolean) As Boolean  
Runs the hypercube. First writes EINGABE, asks some parameters via "dlgGHMiterSpec", writes Steuer, cleans output files, writes TABELLE, updates EINGABE, Runs the hypercube, tests for TestProto003, transfers the secondaries to the kernel via 'SetSecondaryGHMITER', and shows Proto002 if there is a problem.
- Function StartXPress() As Boolean  
Checks whether there is a XPress licence file, now 'XPAUTH.XPR'
- Sub ToonXpressMelding()  
Shows the XPress logfile if the licence error was returned.
- Sub SetModularOptions(B1 As Boolean, B2 As Boolean, B3 As Boolean)  
Converts the 3 singleton options into one value.

- Function TestHitasFiles() As Boolean  
Checks whether the top-level of a hierarchy is a single code. This si a problem for the kernel.
- Function RunHITAS(Tabnummer As Long) As Boolean  
Writes the NPF and the NSF files with the parameters fro the Modular.,Adds a few extra parameters, calls 'SjoemelBestandenVoorHitas' for a 1-dim table, 'checks the real unequalities for the apriory bounds and the protection intervals, asks for the singleton options, runs modular (different version for Cplex and XPress) and if successful passes the secondaries to the kernel.
- Sub TestTrivialSolution(TabNo As Long, NSec As Long)  
Checks whether there are any unsafe cells.
- Function RunJJ(Tabnummer As Long, TotUnsafe As Long) As Boolean  
Runs FullOptimasation solution. First generates a JJ-file, calls 'FullJJ' from the modular OCX and is successful stores the secondaries in the kernel.
- Function RunRounding(Tabnummer As Long)  
Runs the rounding procedure. If needed partitions the JJ-file, runs the rounder, if needed on the partitioned table. Only available for XPress.
- Function CountSecondaries(Tabnummer As Long) As Long
- Function RunNetwork(Tabnummer As Long, ForBatch As Boolean) As Boolean  
Rusn the network solution. Different versions for a non-hierarchical and a hierarchical table.
- Sub Suppress(Tabnummer As Long, Singleton As Boolean, Soort As Long)  
Calls the suppression method selected.
- Sub InitialiseerTable(Helemaal As Boolean)  
Build a new table presentation.
- Private Sub CMDAudit\_Click()  
Calls the audit routine. Writes the JJ-file, the parameter file and calls the audit-routine ('Intervalle')
- Private Sub cmdTableSummary\_Click()  
Show the table summary
- Private Sub cmdWriteTable\_Click()  
Writes the table via 'frmSaveTable'
- Sub UpdateFlexAndCaption()  
Updates thecell details info in the right hand top frame
- Private Sub cmdHistory\_Click()  
Calls the apriori/history file
- Private Sub UpdateInfo()

### 3.2.1.23. MakeAP.frm

**Make Apriory file**

Name of the safe file: K:\TauArgus\VB\Data\tx.txt

Name of the apriory file: K:\TauArgus\VB\Data\xx.hst

Separator: ,

Dimension Output: 2

1: Other

2: Other

Go Ready

Status	Omit	Safe	Unsafe	Protect	Weight	Weight Value
Safe	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>	6
Safe (manual)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="checkbox"/>	7
Unsafe	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>	5
Unsafe (request)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>	
Unsafe (Freq)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>	
Unsafe (Zero cell)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Unsafe (Singleton)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Unsafe (Singleton) (manual)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Unsafe (manual)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Protected	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/>	
Secondary	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Secondary (from man.)	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Empty (non-struct.)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	
Empty	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>	

Writes a aprori file, bases on a table saves via the code/value option in writetables. For each of the different statuses the action can be specified and a weight for the suppression process.

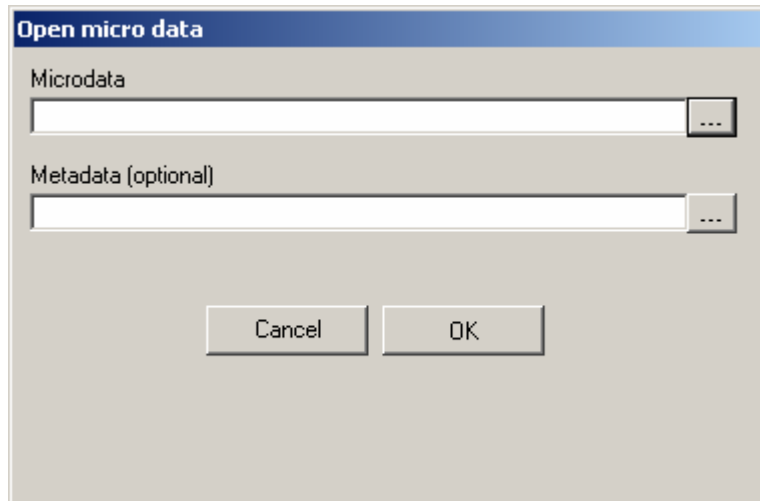
### 3.2.2. $\mu$ -ARGUS and $\tau$ -ARGUS forms

Sometimes a compiler directive (MU or TAU) is used when appropriate

#### 3.2.2.1. dlgArgusInput.frm

Shows a pop-up window to ask an input value. MinScore is a minimum value for the entry and can be set before calling this window

#### 3.2.2.2. **dlgOpenFile.frm**



This window uses the VB file open box to ask for the name of the data file and the meta data file. If the meta data file has the same name as the data file, except for the extension (.RDA) this name is filled in automatically. But it can be overwritten. If non-standard extensions are used, these will be stored in the registry. When using  $\tau$ -ARGUS again these extensions are added to the list of extensions on the file open box.

#### 3.2.2.3. **frmBatch.frm**

This window just shows the progress of a batch-run. A text-window is filled during the process. The progress bars at the bottom show the progress of the modular protection. Similar to the progress bars in the interactive version (frmViewtable). All the functions for the batch-process are in the module TauFunctions.

#### 3.2.2.4. **frmSplash.frm**



#### 3.2.2.5. **frmViewReport.frm**

When a table is saved, a report is written (in HTML). This window just shows the report.

### 3.2.3. $\tau$ -ARGUS modules

#### 3.2.3.1. Module1.bas

A standard module, generated by VB. The program starts here and calls some initialisation

#### 3.2.3.2. TauArgusData.bas

Contains the main data structures of  $\tau$ -ARGUS.

MetaDataStruct As Metadata

All the information for each variable is stored here.

TableSetStruct(1 To 10) As TableSetT

All information for each table is stored here.

Also the initialisation routines are in this module.

#### 3.2.3.3. TauFunctions.bas

- Public Function StatLabel(i As Long) As String  
Returns the name of a status
- Function LeesRegel(n As Long) As String  
Reads a line and replaces a tab by a space
- Public Function LeesMetaDataFile() As Boolean  
Reads a RDA-file for microdata
- Public Function LeesTableMetaData(DFNumber As Long) As Boolean  
Reads a RDA-file for tabular data
- Function TestTabularMetadata(DFNumber As Long) As Boolean  
Checks a tabular RDA for some consistencies
- Sub SjoemelBestandenVoorHitas1()  
Modular/Hitas cannot handle a 1-dim table. Therefore in that case a artificial second dimension (a column of zeros) is added to the table before calling modular.
- Sub AddTrail(S As String, L As Long)  
Adds trailing spaces
- Sub Progress(Part As Long, Tot As Long, NLines As Long, Frm As Object)  
Updates a progress bar
- Public Function PrepareTableData(Frm As Object) As Boolean  
Sets up the table structures when a table is read in batch
- Public Function LeesOneTabel(CalcTotal As Boolean, DFNumber As Long, Frm As Object) As Boolean  
Reads a table. Goes through the file 3 times, to find the sizes of each filed, to prepare for the code lsits and to actually raed the cell info./.
- Function GenerateAdditErrOverview(Fn As String, DFNumber As Long, ForIncomplete As Boolean) As Long
- Dim NR As Long, NLiveTot As Long, NLiveSub As Long, CompleteOke As Boolean
- NLiveSub = 0
- If Freqs(K) > 0 Then NLiveSub = NLiveSub + 1
- If NR = NLiveTot + NLiveSub Then
- If NLiveTot = 1 And NLiveSub = 0 Then CompleteOke = True 'normal not available part If NLiveTot = 0 And NLiveSub = 0 Then CompleteOke = True 'all not available
- Function Avail(f As Long, InComp As Boolean) As String
- Function TF(B As Boolean) As String
- Sub InverseWeightJJ(FName As String, Temp As String)
- Public Function ReScale(Hs As String, Scalefactor As Double, D As Long) As String
- Sub OpschonenJJFormatForKomma(FF As String, Temp As String, D As Long, scaling As Boolean, MinTabVal As Double)
- Sub AprioryWeightJJ(FName As String, Temp As String, Q As Long, D As Long)
- Function NaamPlusMis(i As Long) As String

- Function VarName(i As Long) As String
- Function SchrijfRecodeFile(VarNo As Long, Pad As String, Filename As String) As Boolean
- Function SchrijfIntermediateFile(TabNo As Long, Filename As String, Param As Long, SuppressEmpty As Boolean, WithTopNFreq As Boolean, Optional FreqOnly As Boolean = False) As Boolean
- Sub InitOCXen()
- Sub LeesZoekCodes(TabNo As Long)
- Function StopFout(St As String, Fn As String, R As Long, Regel As String) As Long
- Function VerwerkHist(Ignore As Boolean, txtSep As String, Fn As String, ExpandBogus As Boolean) As Boolean
- Sub AppendFile(Fn As String)
- Function AddBatchMicroTable(St As String) As Boolean
- Function GetGetal(Hs As String, Optional Sep As String = ",") As Long
- Function ReadSafetyRule(St As String) As Boolean
- Sub PrepareAndUndo(TabNo As Long)
- Function BatchSuppress(St As String) As Boolean
- Function ApplyBatchAPriory(Staart As String) As Boolean
- Function BatchJJ(St As String) As Boolean
- Function BatchWrite(St As String) As Boolean
- Function ReportSTR(St As String) As Boolean
- Function BatchRecode(St As String) As Boolean
- Function BatchLinkedModular() As Boolean
- Function VerwerkBatch(BatchFileName As String) As Long
- Sub WriteBatchJob()
- Sub FillDimArray(n As Long, TabNo As Long)
- Sub LeesCell(n As Long, TabNo As Long)
- Sub EmptyTotCell()
- Sub AddToTotCell()
- Function MarginalSuppression(TabNo As Long, NSecondary As Long) As Boolean
- Function AMPL1H2D(Tabnummer As Long) As Boolean
- Function SchrijfJJ(Tabnummer As Long, FName As String, Rounding As Boolean, Singletons As Long, MinFreq As Long) As Boolean
- Public Sub OptimizeParameters()
- Public Function AddSingletonToJJ(JJFN As String, Singleton As Long, MinFreq As Long) As Boolean
- Sub Controleer(Temp As String, Fn As String, D As Long, scaling As Boolean)

### 3.2.4. $\mu$ -ARGUS and $\tau$ -ARGUS modules

#### 3.2.4.1. ArgusFunctions.bas

This module contains a set of general purpose functions; only the more relevant functions are listed here.

- Public Sub ActivateObject(obj As Object, B As Boolean)  
Sets an object to active or vice versa. A textfield becomes grey etc
- Sub MoveToTemp(NaarTemp As Boolean, CurrentDir As String)  
Some external executables like GHMiter want to write in the current dir.  
In that case the current dir is temporarily changed to the temp dir
- Public Sub schrijfKopHtml(Titel As String)  
Writes the standard beginning of a HTML-report file
- Public Function SchrijfStaartHTML()  
writes the standard last lines of a HTML-report
- Function OpenBestand(cmd As Object, Filter As String, title As String, TT As Integer, ll As Integer, Fn As String, Optional CancelPressed = False, Optional Openen = True) As String  
Asks for a file name, can be used for opening and saving a file. TT and ll are not used any more

- Sub WriteLogBoek(St As String)  
Writes a string to the logbook. Including date and time. Default Tempdir+\logbook.txt
- Function SDCMsgBox(St As String, keyval As VbMsgBoxStyle, Optional Caption As String="" ) As VbMsgBoxStyle  
Replaces the VB messagebox. But in batch-mode writes to the logbook. In batch the  $\tau$ -ARGUS stops if the message is other than a warning
- Public Sub ReadCodeList(Fn As String, VarNo As Long, Mis1 As String, Mis2 As String)  
Reads the optional codelist file when displaying the right hand pane of the main window
- Public Sub RefreshLeftPane(Optional CurVar As Long = 0)  
Writes the left hand side of the main window: vars and unsafe combinations per variable, both for  $\mu$ -ARGUS and  $\tau$ -ARGUS.
- Public Sub RefreshRightPane(VarNo As Long, Optional Sorted As Boolean = True)  
Writes the right hand side of the main window: unsafe combinations per code, both for  $\mu$ -ARGUS and  $\tau$ -ARGUS.
- Public Sub TextBoxEnable(O As Object, B As Boolean, Optional EraseField = True)  
Activates/deactivates a textbox
- Public Function ZoekVariabele(VarNaam As String, SelectedTable As Long) As Long
- Public Sub Muis(Ob As Object, Soort As String)  
Different shapes for the mouse
- Public Sub GeneralInitialiseer()  
This is the first function called. Searches the TempDir and in case of a batch-call prepares for that.
- Public Function GetToken(R As String, Optional DoubleQuote As Boolean = False) As String  
Gets a token from a string and if needed removes the double quotes; a space is the separator
- Public Function GetTokenSEP(R As String, Sep As String) As String;  
Gets a token, but with a specified separator
- Public Function GetCommaToken(R As String, Sep As String) As String  
Same as above, but also removes double quotes
- Function OntQuote(S As String) As String  
removes double quotes
- Function FormatSeconds(S As Long) As String  
Converts a number of seconds in a string with hours, minutes and seconds
- Function VervangExtensie(FF As String, EE As String) As String  
Replaces the extension of a file name.
- Function ValKomma(St As String) As Double  
Replaces a “,” by a “.”. Needed for numerical values passed over to codes that require a decimal point in all cases.
- Function GetValSetting(Hs1 As String, HS2 As String, hs3 As String, DefValue As Double) As Double  
Converts a string from the settings in the registry into a numerical value.

#### **3.2.4.2. MuTauArgusData.bas**

Declaration of some general variables

#### **3.2.4.3. SPSSFunctions.bas**

- Function OpenSPSS(FName) As Boolean
- Function VerwerkSPSS\_RDA() As Boolean
- Sub UpdateSPSSBPos()
- Function ExporteerSPSS()
- Public Sub GetAndKillProcesses(strProcess As String)
- Public Function VervangSPSS(NewFile As String, OldFile As String, RawAsciiFile As String, SafeAsciiFile As String) As Boolean
- Public Sub KillSPSS(FName As String)



#### **3.2.4.4. Exec.bas**

- Public Function ExecCmd(cmdline\$)  
A rather standard VB-solution for running a child program
- Public Sub StrReplace(St As String, OldSt As String, NewSt As String)
- Public Function VerlengString(Hs As String, L As Long) As String  
Retruns a string to a given length (spaces added in front)
- Public Sub DeleteFile(Fn As String)
- Public Function Fullpath(Fn As String) As String
- Public Sub SplitFileName(Fn As String, Pad As String, FName As String, Ext As String)  
Splits a file name in path, name and extension
- Public Function GetPath(Fn As String) As String  
Returns the path only of a file
- Public Sub KopieerFile(FN1 As String, FN2 As String)  
Copies a file
- Public Function ErrorMessageString(EN As Long) As String  
Returns an error message from the resource file
- Function FileExist(Fn As String) As Boolean  
Checks the existence of a file
- Function XMin(m1 As Double, m2 As Double) As Double
- Function XMax(m1 As Double, m2 As Double) As Double
- Function Max(m1 As Long, m2 As Long) As Long
- Function StrToLong(Hs As String, ii As Long, ErrMess As String) As Boolean
- Function StrToDouble(Hs As String, XX As Double, ErrMess As String) As Boolean

## 4. Global description of structure of Modular Approach (HiTaS) source code

Two OCX-versions are made: one for use with XPress and one for use with CPLEX. Both OCX-code make use of files “FullJJCall.cpp” and “MainCall.cpp” which are independent of used LP-solver. Compiler directives are used to compile for CPLEX or XPress (this is needed for correctly using JJ’s source files).

Two methods are available through the OCX:

AHiTaS (to protect a table using modular (HiTaS) approach): uses MainCall.cpp

FullJJ (to protect a table using JJ-source on complete table-structure): uses

FullJJCall.cpp.

### 4.1. AHiTaS (MainCall.cpp)

Check if XP or CPLEX Licence is indeed available.

Set values of variables using OCX-interface (if not set in interface, use default []):

MAX\_TIME (maximum time to spend on single subtable) [600]

ZERO (JJ-variable) [1e-7]

ZERO\_1 (JJ-variable) [1e-7]

ZERO\_2 (JJ-variable) [1e-10]

INF = (JJ-variable) [21.4e12]

MAX\_COLS\_LP (JJ-variable) [50010]

MAX\_ROWS\_LP (JJ-variable) [15000]

MAX\_CUTS\_ITER (JJ-variable) [50]

MAX\_CUTS\_POOL (JJ-variable) [500000]

MIN\_VIOLA (JJ-variable) [0.0001]

MAX\_SLACK (JJ-variable) [0.01]

FEAS\_TOL (JJ-variable) [1e-7]

OPT\_TOL (JJ-variable) [1e-9]

Read values for constants from file (if not in file, use default values []):

MINCOUNT (parameter for frequency rule) [3]

LOWERMARG (factor lower protection level marginal cell in backtracking) [0.9]

UPPERMARG (factor upper protection level marginal cell in backtracking) [1.1]

DOLIFTUP (boolean: lift table in case of negative values) [if MINTABVAL = 0 true, else false]

MINTABVAL (smallest cell value in table) [0]

MAXTABVAL (largest value in table) [20000000]

DECIMALS (number of decimals used) [0]

MAXWEIGHT (maximum weight) [20000]

APRIORI (boolean: use aprioribounds with percentages (p-q rule)) [0 = false]

APRIORILB (percentage for lower aprioribound) [0]

APRIORIUB (percentage for upper aprioribound) [2000000000]

DISTANCE (use “distance” as cost-function) [0 = false]

D1 (5 values to be used in distance cost function) [1 1 1 1 1]

D2 (5 values to be used in distance cost function) [1 1 1 1 1]

D3 (5 values to be used in distance cost function) [1 1 1 1 1]

D4 (5 values to be used in distance cost function) [1 1 1 1 1]

LINKED (boolean if linked tables are provided) [0 = false]

Read hierarchies from files and construct hierarchy-structures.

If no hierarchy-files: non-structured table. If at least one hierarchy table: structured table.

### If no hierarchy AND not linked => protect table as a single table

BTab.ReadData(BTabFileName)

Read table info from file into BTab structure. BTab contains complete  $n$ -dimensional table ( $1 \leq n \leq 4$ ), each cell has info about its

value, status, protection levels, apriori levels, number of contributors, cost to suppress.

FillTableNoCost

Fill table BTab in format needed for JJ-routines

If needed and requested

LiftSubTableUp (lift values of Tab such that no negative cells remain)

If needed and requested

SetCountBounds (add virtual cells consisting of multiple cells per row with number of contributors < MINCOUNT)

DoSingletons (add virtual cells consisting of two singletons or one singleton and one other unsafe cell, only when exactly two unsafe cells in corresponding row)

Returns (among other things) reldim (dimension of subtable may be smaller than  $n$ )

SuppressNoCost

Protect table using JJ-routines:

First check if search for suppression pattern is necessary (*not* necessary in case only safe cells or only unsafe cells)

Adjust weights (costs) if necessary (JJ wants all weights  $\leq$  MAXWEIGHT)

LoadTableIntoPCSP: Load table into JJ-structure

PCSPoptimize: find optimal suppression pattern

PCSPsolution: read found solution

Check whether obviously infeasible or not

Update

Update statuses of cells of BTab

BTab.APrintTabel

Write solution to outputfile (only coordinates of secondary suppressions)

Free memory

### **If hierarchical AND/OR linked => protect table using subtables**

BTab.ReadData(BTabFileName)

Read table info from file into BTab structure. BTab contains complete  $n$ -dimensional table ( $1 \leq n \leq 4$ ), each cell has info about its value, status, protection levels, apriori levels, number of contributors, cost to suppress

*Start Track*

ReadBTInfo

Read info about BackTracking (if available), i.e., read statuses

DefineSubGroups

Construct list of all subgroups, defined by the hierarchies (see HiTaS-nota)

*For each Subgroup:*

DefineSubGTabs

Construct list of all subtables in the subgroup

*For each subtable in the subgroup:*

FillTable

Fill subtable Tab in format needed for JJ-routines

If needed and requested

LiftSubTableUp (lift values of Tab such that no negative cells remain)

If needed and requested

SetCountBounds (add virtual cells consisting of multiple cells per row with number of contributors < MINCOUNT)

DoSingletons (add virtual cells consisting of two singletons or one singleton and one other unsafe cell, only when exactly two unsafe cells in corresponding row)

Returns (among other things) realdim (dimension of subtable may be smaller than  $n$ )  
 If realdim = -11: subtable does not need to be protected (part of Linked-table setup, see papers)  
 TestNewTable  
     Check if Table has been protected in previous Track and if no status has changed due to Backtracking  
     If yes:       ReadOldStats (saved in previous Track)  
               Update (update statuses in BTab)  
     If no:       SaveBasisStatsBefore (save statuses before suppression routines start)  
               Suppress (Protect Table using JJ-routines)  
               Update (update statuses in BTab)  
               SaveBasisStatsAfter (save statuses of solution)  
               If marginal was newly suppressed:  
                   AddHistory (add info to file for BackTracking)  
                   Set flag for BackTracking  
               If subtable was obviously infeasible: stop process  
 Continue with next *subtable*  
  
 If BackTracking is needed go to *Start Track*  
 Else continue with next *Subgroup*  
  
 If no more *Subgroups*:  
     BTab.APrintTabel  
     Write solution to outputfile (only coordinates of secondary suppressions)  
     Free memory

## 4.2. FullJJ (FullJJCall.cpp)

Check if XP or Cplex Licence is indeed available.  
 Set values of variables using OCX-interface (if not set in interface, use default []):  
     MAX\_TIME (maximum time to spend on single subtable) [600]  
     ZERO (JJ-variable) [1e-7]  
     ZERO\_1 (JJ-variable) [1e-7]  
     ZERO\_2 (JJ-variable) [1e-10]  
     INF = (JJ-variable) [21.4e12]  
     MAX\_COLS\_LP (JJ-variable) [50010]  
     MAX\_ROWS\_LP (JJ-variable) [15000]  
     MAX\_CUTS\_ITER (JJ-variable) [50]  
     MAX\_CUTS\_POOL (JJ-variable) [500000]  
     MIN\_VIOLA (JJ-variable) [0.0001]  
     MAX\_SLACK (JJ-variable) [0.01]  
     FEAS\_TOL (JJ-variable) [1e-7]  
     OPT\_TOL (JJ-variable) [1e-9]  
 Read table from file (JJ-format) into structure to feed to JJ-routines (first read cells, then read constraints)  
 PCSPLoadprob  
     load problem into JJ-structure  
 PCSPOptimize  
     Find optimal suppression pattern  
 PCSPsolution  
     Read found solution  
 Write solution to outputfile (only coordinates of secondary suppressions)  
 Check whether obviously infeasible or not  
 Free memory  
 Check if optimality has been reached for sure and return info about this to caller of OCX

## 5. Technical documentation NewTauArg.Dll

NewTauArgus is a COM DLL that contains functions and events for securing table files. As a development environment is used Visual C++ 6.0.

For protection, the following events and functions.

Each function and event includes:

1. A brief description
2. The parameters are of type:
  - VARIANT = variant datatype (meestal gebruikt om een string van arrays door te geven als input) VARIANT = variant data type (usually used for passing a string or an arrays as input)
  - Bstr string = (input and return)
  - double = 8-byte real
  - Long = 4-byte integer
  - VARIANT\_BOOL = VARIANT\_TRUE or VARIANT\_FALSE (used only as return value)
  - short = 2-byte integer
3. Return value

### 5.1. Events

#### Fire update progress

Description:

Indicates the percentage of a certain action is resolved.

Is fired in the functions [file explore](#) and [compute tables](#).

Parameters:

short perc percentage processed

Return value:

no

### 5.2. Functions

#### 5.2.1. Introduction

The functions in alphabetical order:

- [ApplyRecode](#)
- [CheckRealizedLowerAndUpperValues](#)
- [CleanAll](#)
- [CompletedTable](#)
- [ComputeCodesToIndices](#)
- [ComputeTables](#)
- [DoActiveRecode](#)
- [DoRecode](#)
- [ExploreFile](#)
- [GetCellDistance](#)
- [GetCellStatusFreq](#)
- [GetMaxnUC](#)
- [GetStatusAndCostPerDim](#)
- [GetTableCell](#)
- [GetTableDimensions](#)
- [GetTableRow](#)
- [GetTotalTableSize](#)
- [GetVarCode](#)

- [GetVarCodeProperties](#)
- [GetVarNumberOfCodes](#)
- [PrepareHITAS](#)
- [SetHierarchicalCodelist](#)
- [SetHierarchicalDigits](#)
- [SetInCodeList](#)
- [SetInTable](#)
- [SetNumberTab](#)
- [SetNumberVar](#)
- [SetRealizedLowerAndUpper](#)
- [SetSecondaryGHMITER](#)
- [SetSecondaryHITAS](#)
- [SetSecondaryJJFORMAT](#)
- [SetTable](#)
- [SetTableCellStatus](#)
- [SetTableSafety](#)
- [SetTableSafetyInfo](#)
- [SetTotalsInCodeList](#)
- [SetVarCodeActive](#)
- [SetVariable](#)
- [UndoRecode](#)
- [UndoSecondarySuppress](#)
- [UnsafeVariable](#)
- [UnsafeVariableCodes](#)
- [WriteCellRecords](#)
- [WriteCSV](#)
- [WriteGHMITERDataCell](#)
- [WriteGHMITERSteuer](#)
- [WriteJJFormat](#)

The functions in functional order:

#### 1. Variables

- [ApplyRecode](#)
- [DoActiveRecode](#)
- [DoRecode](#)
- [GetVarCode](#)
- [GetVarCodeProperties](#)
- [GetVarNumberOfCodes](#)
- [SetHierarchicalDigits](#)
- [SetHierarchicalCodelist](#)
- [SetNumberVar](#)
- [SetVarCodeActive](#)
- [SetVariable](#)
- [UndoRecode](#)
- [UnsafeVariable](#)
- [UnsafeVariableCodes](#)

#### 2. Tables

- [ComputeTables](#)
- [GetCellStatusFreq](#)
- [GetMaxnUC](#)
- [GetStatusAndCostPerDim](#)
- [GetTableCell](#)
- [GetTableDimensions](#)
- [GetTableRow](#)
- [GetTotalTableSize](#)
- [SetNumberTab](#)

- [SetTable](#)
- [SetTableCellStatus](#)
- [SetTableSafety](#)
- [UndoSecondarySuppress](#)
- [WriteCellRecords](#)
- [WriteCSV](#)
- 3. File explore
  - [ExploreFile](#)
- 4. Security
  - [PrepareHITAS](#)
  - [SetSecondaryGHMITER](#)
  - [SetSecondaryHITAS](#)
  - [SetSecondaryJJFORMAT](#)
  - [WriteGHMITERDataCell](#)
  - [WriteGHMITERSteuer](#)
- 5. Cleaning
  - [CleanAll](#)

### 5.2.2. CleanAll

Description:

Deletes all specified data and displays all the reserved memory.

Also called by [SetNumberVar](#)

Parameters:

no

Return value:

no

### 5.2.3. SetNumberVar

Description:

Specifies the number of variables, reserves memory for the administration of the variables.

First calls [CleanAll](#) to, in case a new action takes place.

Parameters: Parameters:

long n\_var number of variables

Return value:

false if n\_var incorrect (<1) or memory, otherwise true

### 5.2.4. SetVariable

Description:

Specifies the properties of a variable.

Parameters:

long	Index	1, 2, ... n_var, index of variable
long	bPos	1, 2, ... starting position in each record of the input file
long	nPos	1, 2, ... number of positions in each record of the input file, up to 100
long	nDec	0,1,... number of decimal places (especially important when writing sage file)
LPCTSTR		Missing1 code for missing the first (only important when categorical variable)
LPCTSTR	Missing2	code for the second missing (only important when categorical variable)
BOOL	IsCategorical	is a categorical variable
BOOL	IsNumeric	is a numerical variable
BOOL	IsWeight	is a weight (must also have property IsNumeric)
BOOL	IsHierarchical	is a hierarchical variable
BOOL	IsHolding	a variable that identifies a holding company and all subsequent records with the same code are considered to belong to the same holding company

For categorical variables: If Missing1 is empty Missing2 promoted. If both are empty is false returned. If both are equal is Missing2 deemed to be empty.

For non-categorical variables missing is irrelevant.

Only one variable can have the property IsHolding or IsWeight.

A variable with property IsWeight may not be (IsCategorical or IsHierarchical).

A variable with property IsHolding may not be (IsCategorical, IsNumeric or IsWeight or IsHierarchical).

Return value:

false if one or more parameters are wrong, otherwise true

### 5.2.5. SetHierarchicalDigits

Description:

Specifies the properties of the codes of a hierarchical variable must precede the call to [explore file](#).

Can not coexist with [SetHierarchicalCodelist](#). The sum of the array must equal *nDigits nPos* (see [SetVariable](#)). The number of groups should not exceed 10.

The code list of a hierarchical variable is determined from the input file and the data hierarchy.

For example a code of 5 characters with partition {2,1,2}

- 12
- 123
- 12301
- 12302
- 12340
- 12345
- 12349
- 124
- 12401
- 12402
- 12403
- 12404
- 12405

All 5-digit codes were in the input file, the others are inferred according to the distribution {2,1,2}. Code 12 is the sum of 123 and Code 124, Code 123 code is the sum of 12,301 - 12 349, Code 124 code is the sum of 12,401 - 12 405.

Parameters:

long	Index	1, 2, ...n_var, index of variable
long	nDigitPairs	number of groups, each with a certain number of characters
Array long	nDigits	Array with <i>NDigitsPairs</i> array of elements with any number of characters in the hierarchy.

Return value:

false if one or more parameters are wrong, otherwise true

### 5.2.6. ExploreFile

Description

Examines each *categorical* variable which codes occur, so the code list are defined.

The only exception: there is a [hierarchical code list](#) specified, then the detected codes should necessarily occur in the code list.

Determines for each *numeric* variable, the maximum and minimum value.

In the input file, all records of equal length, in case of fixed format. As the only exception are empty records, which are neglected without warning.

Parameters:

LPCTSTR	FileName	Name investigate file
---------	----------	-----------------------



<input type="checkbox"/> long	ErrorCode	See below
<input type="checkbox"/> long	LineNumber	Line number where error occurs
<input type="checkbox"/> longVarIndex	Index of numeric variable considered, which is not numeric (0 = N)	

Return value:

false if an error occurs, its nature is in [error code](#) , otherwise true

### 5.2.7. SetNumberTab

Description:

Specifies the number to tables. Clears the previously specified tables, if any.

Parameters: Parameters:

long	nTab	Number of tables
------	------	------------------

Return value:

false if nTab incorrect or insufficient memory, otherwise true

### 5.2.8. SetTable

Description:

Specifies the attributes of a table. Should immediately be followed by [SetTableSafety](#), as a function can simply no more than 15 parameters

Parameters:

long	Index	Index van de tabel (1, 2, ... nTab) Index of the table (1, 2, ... NTAB)
long	nDim	Number of dimensions (max 10)
Array long	ExplanatoryVarList	The index for each dimension of the variable (1, 2, ... n_var)
long	ResponseVar	Index of Response variable (1, 2, ... n_var)
long	ShadowVar	Index of Shadow variable (1, 2, ... n_var)
long	CostVar	Index of Cost of variable (1, 2, ... n_var), -1 (number of records), -2 (equal weights)

Return value:

False if specification errors, otherwise true

### 5.2.9. SetTableSafety

Description:

Gives parameters for table security

Parameters:

long	Index	Index table (1, 2, ...)
long	SafetyRule	Safety Rule: <i>Dominance</i> (1) or <i>PriorPosteriorRule</i> (PQ rule) (2)
long	DominanceNumber	Number high scores of Shadow to track
long	DominancePerc	Dominance percentage
long	PriorPosteriorP	P value for the PQ rule
long	PriorPosteriorQ	Value of Q for the PQ rule ( $P \leq Q$ )
long	PriorPosteriorN	Value of N for the line PQ ( $N \geq 1$ )
long	SafeMinRec	Minimum number of records per table cell that is regarded to be safe
BOOL	ApplyWeight	Whether or not weight applied to table cells
BOOL	ApplyWeightOnSafetyRule	Whether or not weight applied to table cells for Safety rules
BOOL	ApplyHolding	Whether or not to apply Holding measures
long	ManualSafetyPerc	Protection level for calculating <i>manually</i> unsafe table cells (0 .. 100)
long	FreqSafetyPerc	For calculation Protection Levels in frequency unsafe cells (0 .. 100)

Return value:

false if specification errors, otherwise true

### 5.2.10. ComputeTables

Description:

Calculates all tables specified with SetTable and SetTableSafety from the input file.  
All tables are stored in memory.

For each table cell the following information is kept:

- Resp (Response variable)
- Cost (Cost variable)
- Freq (frequency)
- Shadow (Shadow variable)
- Status
- nMaxScore (number to keep track of high scores)
- MaxScore (Shadow, unweighted)
- MaxScoreWeight (unweighted MaxScores associated weight, 0 if not applicable or not to apply)

Parameters:

☐long      ErrorCode      See below, not applicable = -1  
☐long      TableIndex Index Index of the table where the error occurred (1, 2, ... NTAB) not applicable = -1

Return value:

false if an error occurs, its nature is in error code, see the [errorlist](#) , otherwise true

### 5.2.11. DoRecode

Description:

Reduces the number of codes of a variable by taking several codes together. In the RecodeString lines are separated by *newline characters* ( $\backslash r \backslash n$ )

The lines contain successively the object code, a ':' and source codes.

Source codes that are to be expanded through adding spaces in front of it.

Codes may be surrounded by quotation marks ("3 -"), it may be necessary if the code ends with a space or a comma as a code or contains a dash.

A code is always treated as an alphanumeric. Therefore, the code "1a" greater than "19" and the code "1" less than "11".

For example:

0 : - 90	all codes less than or equal to 90
1 : 91 - 500	codes from 91 to 500
2 : 501	codes greater than or equal to 501
3 : 11, "3 ", 512, 530-570, 930-970	multiple series allowed, separated by ','

Parameters:Parameters:

long	VarIndex	1, 2, ... n_var, index of variable
LPCTSTR	RecodeString	The specification of the recoding
LPCTSTR	Missing1	The value of Missing1
LPCTSTR	Missing2	The value of Missing2
<input type="checkbox"/> long	ErrorType	Type of the error
<input type="checkbox"/> long	ErrorLine	Line where the error occurred
<input type="checkbox"/> long	ErrorPos	Position where the error occurred
<input type="checkbox"/> BSTR	WarningString	Warning of overlapping sources, not mentioned codes, etc.

Return value:

false if one or more parameters are wrong (eg no categorical variable, there is nothing left of a code), otherwise true

### 5.2.12. DoActiveRecode

Description:

Reduces the number of codes of a hierarchical variable by ignoring inactive codes  
Codes can be set (in)active with [SetVarCodeActive](#)

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

Return value:

false if parameter error (eg no hierarchical variable), otherwise true

### 5.2.13. SetVarCodeActive

Description:

Sets all the underlying (descendants) codes for a parent code of a hierarchical variable (in) active.

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

long CodeIndex 0 (total), 1, 2, ... index of code of variable

BOOL Active true or false (active or inactive)

Return value:

false if parameters are wrong (eg no hierarchical variable, not parent code), otherwise true

### 5.2.14. ApplyRecode

Description:

Re-calculate, because a recoding example, the (sub) tables. The results can be retrieved using:

- [UnsafeVariable](#)
- [UnsafeVariableCodes](#)

Parameters:

no

Return value:

no

### 5.2.15. UndoRecode

Description:

Reverses a recoding. There can be no re-encoding on a recoding, a recoding is always performed to the basic code of a list variable. In a hierarchical variable, all codes will be put to active first. A new recoding can be performed without first calling this function.

The results can be retrieved using:

- [UnsafeVariable](#)
- [UnsafeVariableCodes](#)

Parameters:

long VarIndex 1, 2, ... n\_var, index of variable

Return value: Return value:

false if the parameter is false, otherwise true

### 5.2.16. GetTableRow

Description:

Gives the cells of a table row and calculates their status according to the [SetTable](#) data safety rule

Parameters:

long TableIndex Table Index (1, 2, ...)

Array long DimIndex -1 = Column variable (exactly once), 0, 1, ... = index in code list of variable (the dimension total is always 0)

<input type="checkbox"/> Array double	Cell	cell content according to <i>counttype</i>
<input type="checkbox"/> Array long	Status	See status list
long	CountType	1= Response, 2 = Shadow, 3 = Cost

Return value:

false if something went wrong, otherwise true

### 5.2.17. GetToalTableSize

Description:

Calculates memory space for all the tables together, it takes into account the number of scores to keep track of (score and its weight) for each table cell

Parameters:

no

Return value:

Total number of bytes (long)

### 5.2.18. GetTableCell

Description:

Gives the properties of a table cell

Parameters:

long	TableIndex	Table Index (1, 2, ...)
Array long	DimIndex	0.1 ... = Index code list of variable dimension (the dimension is always 0 total)
<input type="checkbox"/> double	CellResponse	Value of the cell response
<input type="checkbox"/> double	CellShadow	Value of the cell shadow
<input type="checkbox"/> double	CellCost	Value of the cell cost
<input type="checkbox"/> long	CellFreq	Value of the cell freq
<input type="checkbox"/> long	CellStatus	Status of the cell
<input type="checkbox"/> Array double	Array of CellMaxScore	The n largest contributors to the shadow table cell, unweighted
<input type="checkbox"/> Array double	Array of CellMaxScoreWeight	The n corresponding weights

Return value:

false if something went wrong, otherwise true

### 5.2.19. GetTableDimensions

Description:

Displays the size of the dimensions of a table.

Parameters:

Long	TableIndex	Index of table (1, 2, ...)
<input type="checkbox"/> Array long	TableDim	Array with dimensions of the table, this takes into account the total and any subtotals, any recoding are also be respected

Return value:

false if something went wrong, otherwise true

### 5.2.20. GetVarNumberOfCodes

Description:

Returns a variable number of codes.

Parameters:

long	VarIndex	Index variable (1, 2, ...)
<input type="checkbox"/> long	NumberOfCodes	Number of codes of a variable, including missing and (sub) totals, when recoding the number of the recoded variable data, including missing and also (sub) totals

<input type="checkbox"/> long	NumberOfActiveCodes	Number of active codes in a variable, including missing and (sub) totals, when recoding the number of the recoded variable data, also including missing and total (or subtotals idle codes are here (yet))
-------------------------------	---------------------	--

Return value:  
false if something went wrong, otherwise true

### 5.2.21. GetVarCode

Description:  
Displays type and code of a variable.

Parameters:

long	VarIndex	Index variable (1, 2, ...)
long	CodeIndex	0 (total), 1, 2, ... index of code of variable
<input type="checkbox"/> long	CodeType	1 = elementary, 2 = (sub) total
<input type="checkbox"/> BSTR	CodeString	String of code (can be empty in total)
<input type="checkbox"/> long	IsMissing	0 = Normal Code, 1 = Missing code
<input type="checkbox"/> long	Level	For hierarchical level (0 = total, 1, 2,...), for non-hierarchical: 0 = total , other = 1

Return value:  
true false if something went wrong, otherwise true

### 5.2.22. GetVarCodeProperties

Description:  
Displays properties of the code of a *hierarchical* variable.

Parameters:

long	VarIndex	Index variable (1, 2, ...)
long	CodeIndex	0 (total), 1, 2, ... index code of variable
<input type="checkbox"/> long	IsParent	0: no parent, other value is parent
<input type="checkbox"/> long	IsActive	0: inactive, other value is active, only for non-parents
<input type="checkbox"/> long	Level	hierarchical level of code from variable (0 = total, 1, 2, ...)
<input type="checkbox"/> long	nChildren	Number of children (0 (only if not older), 1, 2, ...)
<input type="checkbox"/> BSTR	Code	String of code (is blank for total)

Return value:  
false if something went wrong, otherwise true

### 5.2.23. UnsafeVariable

Description:  
Calculates for each relevant dimension (1, 2, ..) the number of unsafe cells for a table variable.

Parameters:

long	VarIndex	1, 2, ... n_var, index of variable
<input type="checkbox"/> long	Count	Number of elements in UCharArray
<input type="checkbox"/> Array long	UCharArray	Array of UCS of dimensions 1, 2, ... Count

Return value:  
false if one or more parameters are wrong, otherwise true

### 5.2.24. UnsafeVariableCodes

Description:  
Gives for each variable code index the frequency, the code and the number of unsafe cells per dimension.

Parameters:

long	VarIndex	1, 2, ... n_var, index of variable
long	CodeIndex	0 (total), 1, 2, ... index of code of variable
<input type="checkbox"/> long	IsMissing	Yes / no missing
<input type="checkbox"/> long	Freq	Frequency code in file
<input type="checkbox"/> BSTR	Code	Alphanumeric value code
<input type="checkbox"/> long	Count	Number of elements in UCharArray
<input type="checkbox"/> Array long	UCharArray	Array with number of unsafe cells per dimension

Return value:

true false if one or more parameters are wrong, otherwise true

### 5.2.25. GetMaxnUC

Description:

Calculates the total number of unsafe cells of all tables.

Parameters:

no

Return value:

Number of unsafe cells from the tables (long).

### 5.2.26. SetTableCellStatus

Description:

Sets the status of a table cell. The given status cannot be *EMPTY*

Parameters:

long	TableIndex	Index table (1, 2, ...)
<input type="checkbox"/> Array long	DimIndex	Array with dimensions of the table (0 = total, 1, 2, ...), for example the grand total of a three-dimensional array: [0, 0, 0]
<input type="checkbox"/> long	CellStatus	Status of the cell

Return value:

false if something went wrong, otherwise true

### 5.2.27. WriteGHMITERSteuer

Description:

Writes control file for table GHMITER.

Parameters:

LPCTSTR	FileName	Control file name table (Steuer)
LPCTSTR	EndString1	Content last but one line of file
LPCTSTR	EndString2	Content last line of file
long	TableIndex	Index table (1, 2, ...)

Return value:

1: OK, otherwise an ErrorCode (long)

### 5.2.28. WriteGHMITERDataCell

Description:

Writing table file Eingabe GHMITER

Parameters:

LPCTSTR	FileName	File name (eingabe)
long	TableIndex	Index table (1, 2, ...)

Return value:

1: OK, otherwise an ErrorCode (long)

### 5.2.29. SetSecondaryGHMITER

Description:

Processes the secondaries in (Ausgabe) for GHMITER

Parameters:

LPCTSTR	FileName	File name table (Ausgabe)
---------	----------	---------------------------

long	TableIndex	Index table (1, 2, ...)
<input type="checkbox"/> long	nSetSecondary	Number of secondary table cells

Return value:  
1: OK, otherwise an ErrorCode (long)

### 5.2.30. PrepareHITAS

Description:

Preparing secondary confidentiality method HITAS/modular

Parameters:

long	TableIndex	Index table (1, 2, ...)
LPCTSTR	NameParameterFile	Name of file containing parameters HITAS
LPCTSTR	NameFilesFile	Name of a file containing the number of dimensions and for each dimension a, code list, a file name for the table cells (a record for each dimension index combination) and a filename for the results

Return value:

false if something went wrong, otherwise true

### 5.2.31. WriteJJFormat

Description:

Writing table control file for JJ

Parameters:

long	TableIndex	Index table (1, 2, ...)
LPCTSTR	FileName	Control file name table
double	LowerBound	lower protection level (0.5, 0.6, ...)
double	UpperBound	upper level (1.5, 1.4, ...)
Bool	WithBogus	whether or not bogus levels (true yes, false no)

Return value:

false if something went wrong, otherwise true

### 5.2.32. SetSecondaryJJFORMAT

Description:

Set secondaries

Parameters:

long	TableIndex	Index table (1, 2, ...)
LPCTSTR	FileName	Control file name table
Bool	WithBogus	whether or not bogus levels (true yes, false no)
<input type="checkbox"/> long	nSetSecondary	Number of secondary table cells

Return value:

1: OK, otherwise an ErrorCode (long)

### 5.2.33. UndoSecondarySuppress

Description:

All secondaries reset to safe, taking into account either manually or normal safe cells

Parameters:

long	TableIndex	Index table (1, 2, ...)
------	------------	-------------------------

Return value:

false true if when correctly terminated, false otherwise

### 5.2.34. GetStatusAndCostPerDim

Description:

Calculates for each dimension the frequencies of all statuses, besides the sum of the cost variable values

Parameters:

long	TableIndex	Index table (1, 2, ...)
<input type="checkbox"/> Array long	StatusArray	An array of at least (nDim + 1) * 14 elements to be allocated. caller responsibility
<input type="checkbox"/> Array double	CostArray	An array of at least (nDim + 1) * 14 elements to be allocated caller responsibility

Return value:

true if when properly terminated, false otherwise

### 5.2.35. WriteCSV

Description:

Writes a table in CSV format, to be read directly into Excel. Each record represents a row in the spreadsheet, the column values are separated by a comma. The labels are enclosed for safety reasons by double quotes, they can contain a comma. If there are double quotes they are replaced by some.

Parameters:

long	TableIndex	Index table (1, 2, ...)
LPCTSTR	FileName	Name of the result file
Array long	DimSequence	Order in which dimensions are treated first in the layer, in the penultimate in the row, the last in the column. So for a three-dimensional table if DimSequence = [3,2,1] is Variable Dimension 3 in the layer-, 2 in the row, 1 in the column dimension

Return value:

false if something went wrong, otherwise true

### 5.2.36. WriteCellRecords

Description:

Writes for each cell of the table a record with successively the dimension codes and the corresponding cell, separated by a comma. The dimension codes are enclosed in double quotes, double quotes are there are being replaced by a single quote.

Parameters:

long	TableIndex	Index table (1, 2, ...)
LPCTSTR	FileName	Name of the result file
BOOL	SuppressEmpty	Empty cells are not (true) or does (false) written

Return value:

false if something went wrong, otherwise true

### 5.2.37. SetSecondaryHITAS

Description:

Set secondary table cells by using PrepareHITAS generated results.

Parameters:

long	TableIndex	Index table (1, 2, ...)
<input type="checkbox"/> long	nSetSecondary	Number of secondaries

Return value:

false if something went wrong, otherwise true

### 5.2.38. GetCellStatusFreq

Description:

Shows frequency of the table cell statuses of a table

Parameters:

long	TableIndex	Index table (1, 2, ...)
<input type="checkbox"/> Array long	StatusFrequency	Number per celstatus

Return value:

false if something went wrong, otherwise true



### 5.2.39. SetHierarchicalCodelist

#### Description:

Specifies the properties of the codes of a hierarchical variable; must precede the call to ExploreFile .

Can not be used together with SetHierarchicalDigits .

#### Parameters:

long	VarIndex	1, 2, ... n_var, index of variable
LPCTSTR	FileName	Name the file with the hierarchical code list, the levels are indicated by the <i>level string</i> at the beginning of a line. A general-total is deemed not to be present, which is added by the program itself as the first (blank) code from a code list. For example: (".." as levelstring)

```
01
..0101
....0101001
....0101002
....0101003
..0102
....0102345
02
..0201
....020134
....020135
etc.
```

LPCTSTR	LevelString	Indicates a hierarchical level, eg " " or "." or "..", in the example above the latest one is chosen
---------	-------------	--

#### Return value:

Correct: 1, otherwise an error code (HC\_FILENOTFOUND ...), long

### 5.3. Error Codes

#### General

```
FILENOTFOUND = 1000
CANTOPENFILE
EMPTYFILE
WRONGLength
RECORDTOOSHORT
WRONGRECORD
NOVARIABLES
NOTABLES
NOTENOUGHMEMORY
NOTABLEMEMORY
SUBTABLENOSUB
SUBTABLEWRONGVAR
NODATAFILE
PROGRAMERROR
WRONGHIERARCHY
TABLENOTSET
VARIABLENOTSET
CODENOTINCODELIST
ISNOTNUMERIC
```

#### Parsing Recode

```
E_HARD = 2000
E_SOFT
E_NOVARTABDATA
E_LENGTHWRONG
E_RANGEWRONG
```

```

E_VARINDEXWRONG
E_EMPTYSPEC
Recode Codes
R_FROMTOOBIG = 3000
R_CODENOTINLIST
R_NOSENSE
R_MISSING2VALID
GHMITER
GHM_TABLEINDEXWRONG = 4000
GHM_NOTPREPARED
GHM_CODEWIDTHTOOBIG
GHM_TABELLEINCORRECT
GHM_STEUERINCORRECT
GHM_EINGABEINCORRECT
GHM_EXECUTEINCORRECT
GHM_SOURCECELLINCORRECT
GHM_NOFILEAUSGABE
Hitas
HITAS_ = 5000
JJFormat
JFF_NOFILE = 7000
JFF_SOURCECELLINCORRECT
JFF_TABLEINDEXWRONG
Hierarchical code list
HC_FILENOTFOUND = 6000
HC_LEVELSTRINGEMPTY
HC_CODEISMISSING
HC_LEVELINCORRECT
HC_CODEEMPTY
HC_NOTHIERARCHICAL
HC_HASSPLITDIGITS

```

#### **5.4. CellStatussen**

```

CS_SAFE = 1,
CS_SAFE_MANUAL,
CS_UNSAFE_RULE,
CS_UNSAFE_PEEP,
CS_UNSAFE_FREQ,
CS_UNSAFE_ZERO,
CS_UNSAFE_SINGLETON,
CS_UNSAFE_SINGLETON_MANUAL,
CS_UNSAFE_MANUAL,
CS_PROTECT_MANUAL,
CS_SECONDARY_UNSAFE,
CS_SECONDARY_UNSAFE_MANUAL
CS_EMPTY_NONSTRUCTURAL,
CS_EMPTY

```

#### **5.5. Program constants**

There are some constants in the program. They are well taken, but can always be put more broadly. Compilation is then an obvious requirement.

MAXCODEWIDTH 100

Maximum length of a variable in the input file, see Explorefile

MAXRECORDLENGTH 32000

Maximum number of positions of a record in the input file

SEPARATOR "\r\n"

Delimiters new line for a specification of a Recode

FIREPROGRESS 1000

After dealing with .. records an event is fired

MAXDIM 8  
Largest number of dimensions of a table. A variable is considered as a hierarchical dimension.

MAXDIGITGROUP 10  
Maximum number of groups of a hierarchical variable, see SetHierarchicalDigits

MAXLEVEL 8  
Maximum number of groups of a hierarchical variable for GHMITER

m\_ValueSeparator ','  
Character used in WriteCSV and WriteCellRecords to separate data, a comma

## 6. The Audit-program INTERVALLE

The Audit program originally has been developed by a team at the University of Ilmenau. It has been written in Delphi and is called by  $\tau$ -ARGUS as a separate program.

The program computes the upper and lower bound of each suppressed cell in a table by solving 2 standard Linear Program (LP) problems.

By comparing these bounds with the upper and lower protection level, it is easy to see whether a cell has been properly protected.

The program reads a table in the standard JJ-format. For Intervalle this is a convenient format as it is almost the input structure for a LP.

First the program reads the JJ-file and then reduces the structure by setting all published cells to (given) constants and the suppressed cells to variables. Then all the relations that have no variables, but only constants are removed from the problem.

On this reduced problem the real computations are performed.

### Structure

The main program is Intervalle.dpr.

The main computations are in UnitIntervalle.pas

Some general data structures are in IntervalleData.pas

All information is passed to the program by a parameter file, which is the first parameter of the program-call. The second parameter is the logfile

The parameter file has 16 lines

1. Name of the JJ\_file
2. Not used
3. Name of the output file
4. Not used
5. PrimOnly or PrimSec. In the first case only the primaries are checked else all suppressed cells
6. Not used
7. 2 (parameters 7-14 are not used)
8. 1
9. 0
10. 0
11. 50
12. 50
13. 0
14. 5
15. Cplex or Xpress, the solver used
16. Name of the licence file for Cplex, Not used for Xpress

The function 'Intervalle' in 'UnitIntervalle' does the real job.

First reduces the problem and writes the LP in the standard Cplex or Xpress format. As object function the sum of all suppressed cells is used, but all but one variable is multiplied by zero.

The opens Xpress or Cplex and starts the optimisation process.

Subsequently the coefficient for each variable is changed from 0 to 1 and the new problem is solved. Both Xpress and Cplex have provisions for this.

If a cell is not properly protected, this is written in the output file. The last field for each variable is 1 (not properly protected) or 0 (OK)

## Index

1.	Introduction.....	1
2.	μ-ARGUS.....	1
2.1.	General Overview.....	1
2.2.	μ-ARGUS.VB user interface .....	2
2.2.1.	frmMain.frm .....	3
	frmRecordLinkage.frm.....	4
2.2.2.	Frm Open file.....	5
2.2.3.	SpecCombi.frm .....	5
2.2.4.	frmMicroAggregation.frm .....	6
2.2.5.	frmNumericVars.frm .....	7
2.2.6.	frmOutput.frm .....	7
2.2.7.	frmRecordLinkage.frm .....	8
2.2.8.	frmRiskChart.frm .....	9
2.2.9.	frmShowTables.frm.....	9
2.2.10.	frmSpecPRAM.frm .....	10
2.2.11.	frmSpecmetadata.frm .....	10
2.2.12.	frmSyntheticData.frm .....	11
2.2.13.	frmGlobalRecode.frm .....	12
2.2.14.	μ-ARGUS modules .....	12
2.2.15.	MuTau modules.....	12
2.3.	Technical documentation MuArg.OCX .....	12
2.3.1.	Events .....	14
2.3.2.	Functions .....	14
2.3.2.1.	SetNumberVar .....	16
2.3.2.2.	SetVariable .....	16
2.3.2.3.	ExploreFile.....	16
2.3.2.4.	SetNumberTab.....	17
2.3.2.5.	Settable .....	17
2.3.2.6.	ComputeTables.....	17
2.3.2.7.	UnsafeVariable .....	18
2.3.2.8.	UnsafeVariablePrepare.....	18
2.3.2.9.	UnsafeVariableCodes .....	18
2.3.2.10.	UnsafeVariableClose .....	19
2.3.2.11.	GetTableUC .....	19
2.3.2.12.	DoTruncate.....	19
2.3.2.13.	DoRecode.....	20
2.3.2.14.	ApplyRecode .....	20
2.3.2.15.	GetMaxnUC .....	21
2.3.2.16.	MakeFileSafeClearOptions .....	21
2.3.2.17.	GetMinMaxValue .....	21
2.3.2.18.	GetVarCode .....	21
2.3.2.19.	SetRound.....	22
2.3.2.20.	SetCodingTop .....	22
2.3.2.21.	SetCodingBottom.....	22
2.3.2.22.	SetSuppressPrior .....	23
2.3.2.23.	SetWeightNoise .....	23
2.3.2.24.	SetPramVar .....	23
2.3.2.25.	SetPramValue.....	23
2.3.2.26.	ClosePramVar .....	24
2.3.2.27.	GetBIRHistogramData .....	24
2.3.2.28.	SetBIRThreshold.....	24
2.3.2.29.	MakeFileSafe .....	24

2.3.2.30.	GetVarProperties .....	25
2.3.2.31.	BaseIndividualRisk .....	26
2.3.2.32.	CleanAll .....	26
3.	$\tau$ -ARGUS .....	26
3.1.	General overview.....	26
3.2.	Description of the individual modules.....	28
3.2.1.	$\tau$ -ARGUS VB user-interface.....	28
3.2.1.1.	dlgGHMiterSpec.frm .....	29
3.2.1.2.	dlgMinRoundBase.frm .....	30
3.2.1.3.	dlgSelectTable.frm .....	30
3.2.1.4.	dlgSpecifyTables.frm .....	31
3.2.1.5.	frmAbout.frm .....	32
3.2.1.6.	frmChangeView.frm .....	33
3.2.1.7.	frmGlobalRecode.frm .....	34
3.2.1.8.	frmInfo.frm .....	36
3.2.1.9.	frmLinkedModular.frm.....	36
3.2.1.10.	frmMain.frm .....	37
3.2.1.11.	frmModularOptions.frm .....	38
3.2.1.12.	frmNetwork.frm .....	38
3.2.1.13.	frmNetwork2D1H.frm .....	39
3.2.1.14.	frmOpenTable.frm.....	39
3.2.1.15.	frmOpenTableSet.frm.....	40
3.2.1.16.	frmOption.frm.....	41
3.2.1.17.	frmReadHistory.frm .....	42
3.2.1.18.	frmSaveTable.frm.....	43
3.2.1.19.	frmSpecifyMeta.frm.....	44
3.2.1.20.	frmSummary.frm .....	47
3.2.1.21.	frmTableMeta.frm .....	48
3.2.1.22.	frmViewTable.frm .....	49
3.2.1.23.	MakeAP.frm .....	52
3.2.2.	$\mu$ -ARGUS and $\tau$ -ARGUS forms .....	52
3.2.2.1.	dlgArgusInput.frm .....	52
3.2.2.2.	dlgOpenFile.frm .....	53
3.2.2.3.	frmBatch.frm .....	53
3.2.2.4.	frmSplash.frm.....	53
3.2.2.5.	frmViewReport.frm.....	53
3.2.3.	$\tau$ -ARGUS modules.....	54
3.2.3.1.	Module1.bas .....	54
3.2.3.2.	TauArgusData.bas .....	54
3.2.3.3.	TauFunctions.bas.....	54
3.2.4.	$\mu$ -ARGUS and $\tau$ -ARGUS modules.....	55
3.2.4.1.	ArgusFunctions.bas.....	55
3.2.4.2.	MuTauArgusData.bas.....	56
3.2.4.3.	SPSSFunctions.bas.....	56
3.2.4.4.	Exec.bas .....	57
4.	Global description of structure of Modular Approach (HiTaS) source code ...	58
4.1.	AHiTaS (MainCall.cpp) .....	58
4.2.	FullJJ (FullJJCall.cpp).....	60
5.	Technical documentation NewTauArg.Dll.....	61
5.1.	Events .....	61
5.2.	Functions .....	61
5.2.1.	Introduction .....	61
5.2.2.	CleanAll.....	63
5.2.3.	SetNumberVar .....	63

5.2.4.	SetVariable .....	63
5.2.5.	SetHierarchicalDigits .....	64
5.2.6.	ExploreFile .....	64
5.2.7.	SetNumberTab.....	65
5.2.8.	SetTable .....	65
5.2.9.	SetTableSafety .....	65
5.2.10.	ComputeTables.....	66
5.2.11.	DoRecode.....	66
5.2.12.	DoActiveRecode .....	66
5.2.13.	SetVarCodeActive .....	67
5.2.14.	ApplyRecode .....	67
5.2.15.	UndoRecode.....	67
5.2.16.	GetTableRow.....	67
5.2.17.	GetToalTableSize .....	68
5.2.18.	GetTableCell .....	68
5.2.19.	GetTableDimensions.....	68
5.2.20.	GetVarNumberOfCodes.....	68
5.2.21.	GetVarCode .....	69
5.2.22.	GetVarCodeProperties.....	69
5.2.23.	UnsafeVariable .....	69
5.2.24.	UnsafeVariableCodes.....	69
5.2.25.	GetMaxnUC .....	70
5.2.26.	SetTableCellStatus .....	70
5.2.27.	WriteGHMITERSteuer.....	70
5.2.28.	WriteGHMITERDataCell.....	70
5.2.29.	SetSecondaryGHMITER .....	70
5.2.30.	PrepareHITAS .....	71
5.2.31.	WriteJJFormat.....	71
5.2.32.	SetSecondaryJJFORMAT .....	71
5.2.33.	UndoSecondarySuppress.....	71
5.2.34.	GetStatusAndCostPerDim.....	71
5.2.35.	WriteCSV.....	72
5.2.36.	WriteCellRecords .....	72
5.2.37.	SetSecondaryHITAS.....	72
5.2.38.	GetCellStatusFreq .....	72
5.2.39.	SetHierarchicalCodelist.....	73
5.3.	Error Codes .....	73
5.4.	CellStatussen .....	74
5.5.	Program constants.....	74
6.	The Audit-program INTERVALLE .....	76