User and system description of SAS2ARGUS

A SAS macro to execute τ-Argus

This is a description of a SAS macro - SAS2Argus - which is designed to facilitate risk assessment of tables and suppressing cells which are not able to publish regarding issues of integrity by establishing a "bridge" between SAS and τ-ARGUS.

This description is a brief summary of the macro that makes it possible, to put disclosure and cell value elimination in a process chain with use of SAS. The subject is complex and data/tables must often be prepared in a special way for it to be possible to protect in aspects of integrity of the table in a safe and rational way.

1.1 Recommended "pedagogy"

To understand the purpose and use of SAS macro SAS2Argus, it is essential to know how the function of τ-ARGUS how it is used. Concepts defined and

used in τ-Argus is also used in the SAS macro as far as possible to facilitate the understanding of how if "fit together". This description of "the bridge" between SAS and τ-Argus - how ambitious it would be - limp without the knowledge of the principles and methods used in τ-ARGUS.

A recommendation for a novice is to "start in τ-Argus" and get acquainted with the program and take advantage of the τ-ARGUS-manual. This smooth the progress significantly the understanding of the whole concept.

Are you also in need to get familiar and understand the implemented methods, in order to choose the "right", we also recommended the methodological handbook (manual) where these methods are described.

1.2 Choice of method

There are a variety of methods to choose from in statistical disclosure control that can be used to produce privacy-protected tables. The choice of method depends on many factors such as how data is used, how the method can be implemented from a practical perspective, and what protection it finally gives the table when published.

Methods can be divided into three categories:

1. Pre-tabular methods to adjust the micro data before the table request
2. Table redesign methods that modify the design of the table by defining the level of detail
3. Post-tabular methods that modify values in an already derived table

To "attack" an already made table - by applying various suppression methods on table values - are managed within the third category, that is, post-tabular methods. This is what the link between SAS and τ-ARGUS offers in its concept.

However, there are situations when it is much easier to solve this problem and consider other solutions earlier in the process chain. For example, define the table with larger geographic areas, to collapse groups and levels where the number of contributors are few. The advantages are several. The methods are easy to implement and provides understandable tables that are able to summaries and the technique is also easy to explain to the user. Sometimes it is not possible, if the definition of the tables' are established in advance (by i.e. international standard) or that this approach gives too much information loss.

Pre-tabular methods to adjust the values of micro data, prior to table the request, provide tables that you are able to summarize, but it can be more difficult to describe for the user, the used data adjustment methods.

Choice of method for protection of informers is still something that must be considered, as it may significantly affect both the work involved, the complexity of the solution and the final effect.

1.3 The purpose of the SAS macro

The application interface in τ-Argus, as used from SAS, works with text files that must be created. And that what's the SAS-macro does. Primary text files that need to be prepared to "put τ-Argus in work" are:

1. Data file with either micro data or aggregated data `[CSV]`
2. A metadata file that describes this data file `RDA]`
3. A batch file (command file) that describes the rules for the risk assessment and which type (s) of result file(s) to be produced.

The purpose of the macro is to facilitate the user to access the functionality of τ-ARGUS by automating the production of these necessary text files. With the use of the metadata information that is available in a SAS data set, or a SAS view against other data (such as SQL tables), much metadata information can be derived as data type, number of decimal places, and more. Then the user needs only to add information about variables roles, risk assessment method and prevention method.

In addition to these three primary text files, additional text files need to be created to describe hierarchies in the data, *(Hiearchy file)* `[HRC]`, if applicable. It is possible to define labels for values with text files, *(Code List file)* `[CDL]`. It is also possible to recode values in the data by use of text files. When it is obvious simpler to do those tasks in SAS by defining labels and/or recode data, so is this possibility to label and recode not implemented within the macro. There is another text file defined where you can set properties on the cells before secondary suppression is performed in a priori file *(The a priori file)*. All of these are described in τ-ARGUS-manual.

1.3.1 Concepts

To facilitate the mapping between the concepts used in the τ-ARGUS and concepts defined in the SAS macro SAS2Argus, the same concepts have been used within the macro, although some terms maybe are unusual for SAS-users, such as `explanatory`[1] Similarly, all the suffixes in the file types that are defined and referenced in τ-Argus, is also used in the SAS-macro to facilitate the reading of τ-ARGUS-manual and understand the mapping between τ-Argus and SAS2Argus.

The macro consists of a main macro - SAS2Argus - and a set of utility macros that are called within the main macro. The "normal" user should not really have to "worry" about these sub macros, but are listed and described at

[1] The word used in τ-Argus for describing dimensions in the table

the end of this document, to some extent describe the components of main macros function.

1.4 The main macro - SAS2Argus

The user needs only to understand the use of the comprehensive SAS macro, SAS2Argus, which then calls the execution of a sequence of other macros which controls syntax, generates all the necessary input files and start a batch job of τ-ARGUS. The function of the macro is thus mainly to get SAS to establish the text files that τ-ARGUS requires to be run through its application interfaces (API) available in τ-Argus, and the macro might also import the results to SAS after execution. With an understanding of the macro, and how the parameter is set and the function of τ-ARGUS, it is relatively easy to perform risk assessment and/or cell suppressing with use of the macro. In place it is an effective “workbench” that facilitate the work significantly and gives the opportunity to test a variety of different parameters and methods.

The alternative is otherwise to execute τ-ARGUS through the graphical user interface (GUI) - which may be a recommendation for a "novice" when it can provide a better understanding of all issues around disclosure, in general, and the features available in τ-Argus in particular. But you will still need to fabricate the text files in the format and content in which τ-ARGUS expects to find them.

2 SAS2Argus - Description

Here is a description of the macro SAS2Argus. How to set up the SAS session to access the functionality of the SAS macro, to "reach" the capabilities of τ-ARGUS. We begin by first describing the parameters the macro utilize, cause this in itself describes much of the potential of the concept.

2.1 Parameters

The macro is a so-called 'Named style macro "- that is, it has a set of named parameters that are assigned values as arguments.

Note that the following description of the parameters is a gross set of the macro's possible parameters and that in practice only a few need to be specified in a typical scenario. Most parameters describes roles for variables, variables that also must exist, or be established, in advance for to later be able to reference those. Some of these roles is unusual and rarely used. A number of parameters are assigned default values, if value is not specified explicitly. Required parameters are marked in **color.**

The parameter list is divided into the following categories:

- General parameters (system parameters)
- Parameters that define the input data into τ-ARGUS
- Parameters for risk assessment and secondary suppression
- Variables and their roles
  - General for both micro data and aggregated data
  - Specific for aggregated data
  - Specific for micro data
- Selection/choice of the output of τ-Argus

As it is known as a "Named style macro" all ***parameters*** are separated by commas, so that commas should ***not*** be used to separate, for example, listed variable names ***within*** a parameter.

To make the context of the parameters more understandable, we here give a brief example of how an invocation of the macro might look like, with no other comments:

```
%sas2argus(InTable     = APPdat.Freqdata,
           Jobname     = Tab_1,
           Explanatory = sex fam age ink,
           Frequency   = Resp,
           SafetyRule  = FREQ(5,30),
           Out         = inter(1),
           RunArgus    = 1,
           SAS         = 2,
           Debug       = 0
)
```

Note that the parameter description below you will also find some useful instructions:

Parameter	Description
General parameters (system parameters)	
JOBNAME	A name that is used as a prefix for all text files that are created for τ-ARGUS and used by τ-Argus in a "job" / execution. Default, unless specified, is **SAS2ARGUS.** This makes it easier, in that sense that you are able to “see" which files that "belong together" in an identifiable context.
RUNARGUS	An option that allows to control if: 0. Only text files are created by the macro. τ-Argus is not executed. 1. Text files are created and τ-Argus is executed **(default)** 2. Don’t create any text file - execute only τ-ARGUS on already created text files. This makes it possible to produce the text files first, edit the text files manually and finally execute the manually edited text files. To overcome exceptional situations not supported by the macro for example.

DEBUG	An option for providing a way to get more information incorporated in the SAS log: 0. No additional information is written in the SAS log 1. Information is written to the SAS log and the log of τ-Argus is also included the SAS-log **(default)** Facilitates debugging and documentation as all available information from execution is found in the SAS log.
HELP	Describes the macro and its parameters in the SAS log: 0. No information in the log **(default)** 1. The macro is described in the log and the macro stops (no execution). The macro is indeed documented in the script code, but this is an easy way to get access to a brief description
SAS	An option that controls imports from τ-ARGUS to SAS: 0. No import from τ-ARGUS to SAS **(default)** 1. Imports the results report in HTML format from τ-ARGUS and includes it in the SAS internal browser. Also imports the output from the τ-Argus in what is called the intermediate format to the SAS WORK. (This is the only format suitable for import into a SAS dataset or table.)

Parameters that define the input data into τ-ARGUS (One of these may be selected, either INPUT or INTABLE)	
InData	Specifies the name of SAS data sets of micro data. Note that this also can be an SQL table. All data sources that SAS supports with access methods can be used. **Must be specified,** either **InData** or **InTable.**
InTable	Specifies the name of SAS data sets for already aggregated data. Note that this also can be an SQL table. All data sources that SAS supports with access methods can be used. However, the data must often "be prepared" in some way and aggregate data must at least have information about the frequency in each cell to be useful as input to risk assessment and suppression.

Risk assessment and secondary suppression	
SafetyRule	Specifies the method of risk assessment to be used. This/these arguments are not checked in a "preventive way" by the macro. Study the τ-ARGUS-manual for valid arguments. Must be defined, unless an the exception in the case cell status (variable name: `Status`) is available and thus the risk assessment is already made. See section 2.3 Risk assessment – SAFETY RULE where this parameter is described.
Suppress	Specifies the method for suppression to be used. This/these arguments are not checked in a "preventive way"by the macro. Study the τ-ARGUS-manual for valid arguments. If this argument is omitted, it means that only a risk assessment is done. See Section 2.4 Secondary Suppression - SUPPRESS where this parameter is described.

The information in the `safety rule` and `suppress` "ports" in the command file `[ARB]` to "tell" τ-Argus what to do. The rest of the information from the parameters below, is used to create the data file

[CSV], with the net content of required / defined roles / variables, and metadata description of it [RDA].

Variables and their roles - Generic for both microdata and aggregated data	
Explanatory	Specifies the name / names of the so-called explanatory variables or dimensional variables that "spans the table." **Must be defined.** Along with the argument two options has been implemented. If the variable is hierarchical, one can in a subsequent brackets add a description of how it is hierarchical, or in which text file that description can be found. Note that variable names specified with spaces as separators if there is more than one in a list. See Section 2.2 Hierarchies where this is described.
Response	This specifies the name of response variable. **Must be defined.**
Shadow	The name of any shadow variable. A company's turnover could be such a "help variable". If not specified, then τ-ARGUS uses the `Response` variable.
Cost	Identification of potential cost variable. This is a weight that can be used when respondents has given permission in advance to publish their values by putting a high cost on observations with permission. If not specified, then τ-ARGUS uses the `Response` variable.
Lambda	Transformation Parameters used in a "simplified Box Cox function" as an exponent of the cost `(COST)`. **Default = 1.**

Variables and their roles - specific for aggregated data	
Frequency	The name of the variable that describes the frequency. **Must be defined** for aggregated data. Otherwise peculiar result can be produced since τ-ARGUS tries to compute the frequency.
LowerLevel	The name of the variable indicating the lowest level of "protection intervals".
UpperLevel	The name of the variable indicating the highest level of "protection intervals".
MaxScore	The name of variables that holds the single highest contributors in each cell. Used in magnitude tables when the dominance rule is applied to pre-aggregated tables. The largest contributors can be computed with PROC MEANS. There is a utility macro that can do this; **Calculate_TopN.sas.**

Status	The name of any variable that indicates status.: Status (value) can then typically be:
	`S = Safe`
	`U = Unsafe`
	`P = Protected`
	Note that in the case of consent (approval) is not advisable to put the status indicator to `S` (SAFE) for cells / observations with consent the "price" can be high with regard to secondary suppression of the table. Use `COST` instead.

TotCode	A constant that specifies/indicates which value that represent the total of an aggregate table. Default används tecken ” `T` ”. Default character is `'T'`.

For aggregated data, it is important not to forget to specify parameter `frequency`, otherwise the τ-ARGUS trying to aggregate already aggregated data to determine the frequency of the number of contributions in each cell.

The `Status` variable is "manufactured" by τ-Argus, when a risk assessment is made, by use of the rules specified in the parameter `safety rule`[2], but can of course created manually as well.

Variables and their roles - the specific micro-data	
Weight	The name of the variable that contains weight.
Holding	The name of the variable that contains information about the corporate group. When observations belonging to the same corporate group should be grouped together in the input file.
Request	The name of the variable that indicates the status when the respondent has requested protection of data or not. Inverse of consent. If consent is relevant, see commentary on `status` .

`Holding` and `Request` is the result of user requirements and the most widely used in *Business Statistics*, and *Foreign Trade Statistics.*

Selection of output from τ-Argus	
Out	Possible choices: **TABLE** () => VarName delimiter (,) Primary (x) Secondary (-) **PIVOT** (0) => VarName No Status **PIVOT** (1) => VarName Status **CODE** (0) => NoName delimiter (,) Primary (-) Secondary (x) No Status **CODE** (1) => NoName delimiter (,) Primary (Part) Secondary (x) No Status **CODE** (2) => NoName delimiter (,) Primary (-) status (1,5,11,14) **CODE** (3) => NoName delimiter (,) Primary (Part) Status (1.11) **SBS**() => NoName delimiter (,) Exp, 0, Exp, 0 .. zero(deleted) Status(V,D,A) zero (deleted) Status (V, D, A) **INTER** (0) => NoName delimiter (;) status only (S, M, U) **INTER** (1) => NoName delimiter (;) Status (S, M, U) If the parameter SAS=1 then we import PIVOT and INTER (if choosen). The “easiest” way may be to experiment in order to understand all the different options for output. The most informative and useful may Intermediate considered to be. **Comment:** *VarName* means that variable names can be found in the file. *NoName* means that no variable names are found in the text file. Characters used as delimiters are listed in parentheses after *Delimiter*. Characters that replaces primary suppressed values are given in parentheses after *Primary.* Characters that replaces secondary suppressed values are given in parentheses after *Secondary. Status / NoStatus* indicates whether the cell status is reported in the output or not.

It is easy to understand that the amount of output format has a historical explanation, as it is in the "τ-ARGUS-sphere" are many stakeholders who had different preferences. SBS is such a special format applied for *Business Statistics.* Most formats is not able or are inappropriate, to import into the SAS as a table. They are best suited to present in tabular form in spread sheets as Excel. The most informative and useful format is the Intermediate, as it is possible to work interactively with, i.e. edit in SAS and send back to τ-ARGUS for a second time, and the intermediate is also easy to tabulate in SAS.

2.2 Hierarchies

How to describe hierarchies may need to be explained. If one has to deal with hierarchical data, these can be of two types:

- Levels in a value domain
- Hierarchies consisting of merger of different values

The first is exemplified by the *Counties, municipalities, parishes* in a six-digit code that can be said to describe level 1 in the first two positions, level 2 in the next two positions, and finally level three in the last two positions.

The second is exemplified by *specifying, for example, the county which forms a country region* in list form. How such a list could look like is described in the manual of τ-ARGUS, and the text file (Hierarchy file) `[HRC],` must be created manually. However, when established, the filename can be specified as an addition to the parameter `Explanatory.`

In the macro SAS2Argus one can describe the first type of hierarchy as follows:

Alt 1. `Region(2 2 2)`

If the case of *county, municipality, parish.* The concept of τ-ARGUS for this is `<HIERLEVELS>.`

The other option available is to specify a file name as follows:

Alt 2. `Region(Region.hrc @)`

... If the hierarchy is described in a file named `Region.hrc.` The second argument `'@'`, (within the parentheses, which is an option in the option), represents the character to be used as so-called "lead string". The concept within τ-ARGUS for this is `<HIERCODELIST>,` and `<HIERLEADSTRING>` respectively. See the τ-ARGUS-manual to better

understand how hierarchies are handled, if this would be the case in your context.

There is much we don't have to write to get a metadata description of the hierarchies, as we can deduce the following from these two types of options given together with the arguments:

`<HIERARCHICAL>` - The name of the variable that is hierarchical

... and either option 1:

`<HIERLEVELS>` - The grouping of a string

... or option 2:

`<HIERCODELIST>` - The name of the file that defines the hierarchy
`<HIERLEADSTRING>` - Special characters to interpret in the file

2.3 Risk assessment - SAFETY RULE

In a disclosure context it could be considered as relatively easy to define which cells/domains that are unsafe to publish. If it is possible to formulate a rule for which cells/domains that are not safe to publish, these cells/domains are fairly easy to identify with relatively simple tools/technology and to also suppress.

In τ-ARGUS the risk assessment rule is specified in the parameter `SafetyRule`. The same concept/word is used within the SAS macro. The argument given here should be given in the style described in τ-ARGUS-manual. There is no check of the specified arguments done by the SAS macro. Information about any inaccuracies in the arguments can then be seen in the τ-ARGUS log. (Tip: Set parameter `debug=1`, then the log from τ-ARGUS is included in the SAS log.)

The parameter `SafetyRule` can take many arguments, and defines the primary suppression, or identification of primary unsafe cells. Note that several primary suppression principles can be chosen, these are then separated by the "|". The following rules can be set: `P, NK, ZERO, FREQ, REQ, WGT, MIS` and described in the table below.

Additional arguments are typically set in subsequent brackets:

SAFETYRULE	Description
P	Percent Rule where additional arguments specified as `P(p,n)` where n is optional and the default set to 1. `P(20,3)` represent percentage rule, where `p=20` and `n=3`.
NK	Dominance Rule where additional argument is specified as `NK(n,k)` where n represents the number of items that may not account for more than k percent of the contribution of the cell/domain.
ZERO	Margin of zero-cells in which additional argument sets the ZERO (`ZeroSafetyRange`) and refer to the size of the margin.
FREQ	The Frequency Rule in with additional argument is specified as FREQ (`MinFreq`, `FrequencySafetyRange`) where `MinFreq` specifies the minimum acceptable frequency and `FrequencySafetyRange` indicates the safety margin for not be able to derive the suppressed frequency.
REQ	"Request" rule - the request for confidentiality[3]. `REQ(Percent1,Percent2,Safetymargin)`. For example, if an informant accounts for 70% and in that case have asked for protection. This requires an additional variable in the table that indicates which items are requested /not requested confidentiality with value `1` or `0`. Variable name (for the role) is indicated by parameter `Request` .
MIS	Missing. If `MIS=0` (which is the default) means that the cells with a code for non-response is still regarded as uncertain whether any `SafetyRule` is violated. If `MIS=1` then the cell i always considered as secure, if at least one contributor has a missing value, then it may be regarded that the contributor with miss cannot be identified. In the SAS macro '9' has been defined as representative value for missing value[4].
WGT	If `WGT=0` (which is default) then weights is not used when aggregating tables or in calculating the `SafetyRule`. The name of the variable is indicated in the parameter Weight .
MAN	"Manual safety margin" (default = 20%). This manually-set safety margin is used only when the status is provided for each cell or when an a priori file[5] is used to set the option that a particular cell is set manually to be uncertain (Manual Unsafe).

All rules can occur multiple times, separated by "|". Example:

```
NK(3,70)|FREQ(3,30)|MIS(1)
```

Explanation:

- `NK(3,70)` implies the dominance rule, i.e., if 3 (or less) observations contribute for 70 percent of the cells value, the cell is regarded as uncertain.

- or `FREQ(3,30)` implies that if the frequency is less than 3 and an intruder, (who knows his own contribution in the cell), should not be not able to reveal another contributors value by less than 30 percent margin. In that case the cell is considered to be uncertain.
- or `MIS(1)` means that if any contribution to the cell is missing, the cell can be considered safe.

The first two `P` and `NK` is then assumed to be applied to the individual level and the subsequent `P` and `NK` is then assumed to be applied to the group level (managed / defined by the parameter `Holding`). The first `FREQ` and `REQ` is assumed to be applied for the individual level and subsequent to the consolidated level.

`ZERO` can be entered only once for each security rule.

2.4 Secondary Suppression - SUPPRESS

Here the arguments, that can be used for parameter `Suppress`, is described. It has the same "syntax" as the `SafetyRule,` that is, additional arguments are given in a subsequent brackets. Note that the first of these additional parameters specify the table number `(TabNo)` and as the SAS macro only handles one table at a time, then this additional parameter always be 1.

The following methods can be used for secondary suppression:

SUPPRESS	Description
GH	`GH(TabNo,AprioriBoundPercentage,ModelSize)` GH-miter, or as it is usually called The Hypercube Method. Together with NET, the only methods available in τ-ARGUS without access to commercial optimizers. The secondary suppress "mechanically" without regard to the optimization and is not recommended as a method since it often causes great loss of information in the table.
MOD	`MOD(TabNo,MaxTimePerSubtable)` A partial method that breaks down a hierarchical table of several non-hierarchical tables, protects them, and finally composes a fully protected hierarchical table. With the MaxTimePerSubtable parameter it is possible to limit the optimizer to work with each subtabell. Is indicated in minutes.
OPT	`OPT(TabNo,MaxComputingTime)` A method that protects a hierarchical table without breaking it down into smaller tables. By setting the maximum time it is possible to limit the time for the optimizer to work. Anges i minuter. Indicated in minutes.
NET	`NET(TabNo)` Network Solutions to be used for large two-dimensional tables with a hierarchy. Requires some special circumstances but no optimizer.

SUPPRESS	**Description** (cont.)
RND	RND(TabNo,RoundingBase,Steps,1,Time,Partitions, StopRule) - RoundingBase refers to the basis for rounding. - Steps means the number of step allowed **(default = 0)** - The fourth argument is a constant (for future extension) - Partitions **(default = 0).** 1 means the inverse. - StopRule (**default=3**) 1 = Rapid-only 2 = First feasible solution 3 = Optimal Solutions

2.4.1 A priori file

An a priori file is a simple text file that can be created in an editor. The information in A priori, the file used *after* risk assessment, but *before* secondary suppression. The file updates including Status, Cost, LowerLevel and UpperLevel and can also put a new value on the level of individual cells when other knowledge of individual cells is at hand. That is, we are given an opportunity to bring existing knowledge to the table after the initial risk assessment process. The current SAS macro has no parameter, (which in itself would be easy to add), to deal with an a priori file, but you can edit the batch file [ARB] by adding the argument:

<APRIORI> ” Filename ”, TabNo , Separator . Each line begins with comma separated values that identify each individual cell and is followed, after another comma, with the following codes and values to provide information with a priori knowledge of specific cells:

Code	Parameter	Explanation
S	-	Status changes to the safe
U	-	Status changed to unsafe
P	-	Status changed to protected
C	New value for cost (COST)	A low cost, increases the likelihood that the cell becomes a candidate for secondary suppression. A high cost increases the likelihood of the same.
PL	Two new values for the LowerLevel and UpperLevel separated by comma.	If the range for a particular cell in advance is likely to be less than the normal range, it may be stated here to get a new "protection level"

Note: The possibility to change the status of a cell is of course limited. This means that a cell that is primary unsafe should not be changed to protected. Nor can a protected cell is changed to unsafe.

The cost `(COST)` must always be a positive value.

It is recommended to be restrictive to set cell status to `protected`. If the intention is to avoid the cell be subjected to a secondary suppression, it is better to put a high cost for the cell. If, nevertheless, would be secondary suppressed by the algorithm in τ-Argus, there are good reasons for this.

See also the τ-ARGUS-manual to get a description of file formats, possible arguments and examples.

3 SAS2Argus - Usage

To initiate the macro we only need a "pointer" to tell SAS where to find the "resources". τ-ARGUS have also to be installed of-course.

```
/*============================================================================
 Initial path location on disk
 ----------------------------------------------------------------------------*/
%let PATH_ini=C:\SAS2Argus;

/*============================================================================
 Path to the generic part and version and the application part. Also give
 the full path to tau_Argus executable [PATH_exe]
 Note: If PATH_tmp is not set => we route to SAS Work as the working area.
       Preferable PATH_tmp is set during test and trouble shooting to be
       able to examine files.
 ----------------------------------------------------------------------------*/
%let PATH_sys=&Path_ini.\1.1;
%let PATH_app=&Path_ini.\Demo;
%let PATH_tmp=&Path_ini.\Demo\APPtmp;
%let PATH_exe=C:\Program\TauArgus\tauARGUS.exe;
```

There is a small program that initiates the path to the SAS programs. The only general SAS programs used are `A01 Init Session.sas` which in turn specifies the path to SAS macros, according to the principle that is used in SAS by the `AUTOSOURCE` option. This means that all macros are compiled in the time they are requested.

```
/*============================================================================
 Init the SAS session
 ----------------------------------------------------------------------------*/
options notes nomprint nomlogic;
filename SASpgm "&PATH_sys.\SASpgm";
%include SASpgm("A01 Init Session.sas");
```

Then we make one final check that everything is in place with a macro call:

```
/*============================================================================
 Macro:      System_Parameters
 ----------------------------------------------------------------------------
   Checks if:
      - full PATH to tau-Argus exe is present so we can find it
      - EXE file for tau-Argus is present so we can use it
      - PATH_tmp is set explicit. Otherwise we route to SAS WORK-path
   and reports finally the used tau-ARGUS work path to the SAS log
 ----------------------------------------------------------------------------*/
%System_parameters;

/*============================================================================
 Here ends the initialisation of the SAS session for use of the bridge
 between SAS and tau-Argus
 ============================================================================*/
```

After the initialization of the general part follows the specific part that preferably begins by defining a `libname` to any data source / database so that we can access data:

```
/*=========================================================================
 Here beneth follows the specific part - the production part of the concept.
 =========================================================================*/

/*=========================================================================
 Libname to data
 -------------------------------------------------------------------------*/
libname APPdat "&PATH_app.\APPdat";
```

Note that `libname` can also be an OLEDB connection to SQL server.

3.1 SAS2Argus - Examples of syntax

Here are some examples of syntax, which calls the macro.

3.1.1 Aggregated Table

Here is an example of an aggregated table `(InTable)` which risk is assessed by the frequency rule, `FREQ (5,30),` meaning that there must be at least 4 observations in each cell, accounting to 30% margin of safety.

```
/*=========================================================================
 EXAMPLE 1 - FREQUENCY
 -------------------------------------------------------------------------
 Comment: An aggregated table only primary suppressed
 -------------------------------------------------------------------------*/
%sas2argus(InTable     = APPdat.TauFreq,
           Jobname     = Example1,
           Explanatory = sex fam          age ink,
           Frequency   = Resp,
           SafetyRule  = FREQ(5,30),
           Out         = table() pivot(1) code(3) inter(1),
           RunArgus    = 1,
           SAS         = 2,
           Debug       = 1
)
```

It is important - for an aggregate table - to indicate `frequency,` as seen in the syntax above, as it is essential information for τ-ARGUS when working with a pre-aggregated table to do a risk assessment.

The naming of files based on `Jobname` - used as a prefix to all files "as part of an execution". The following files are produced in the above call for τ-ARGUS:

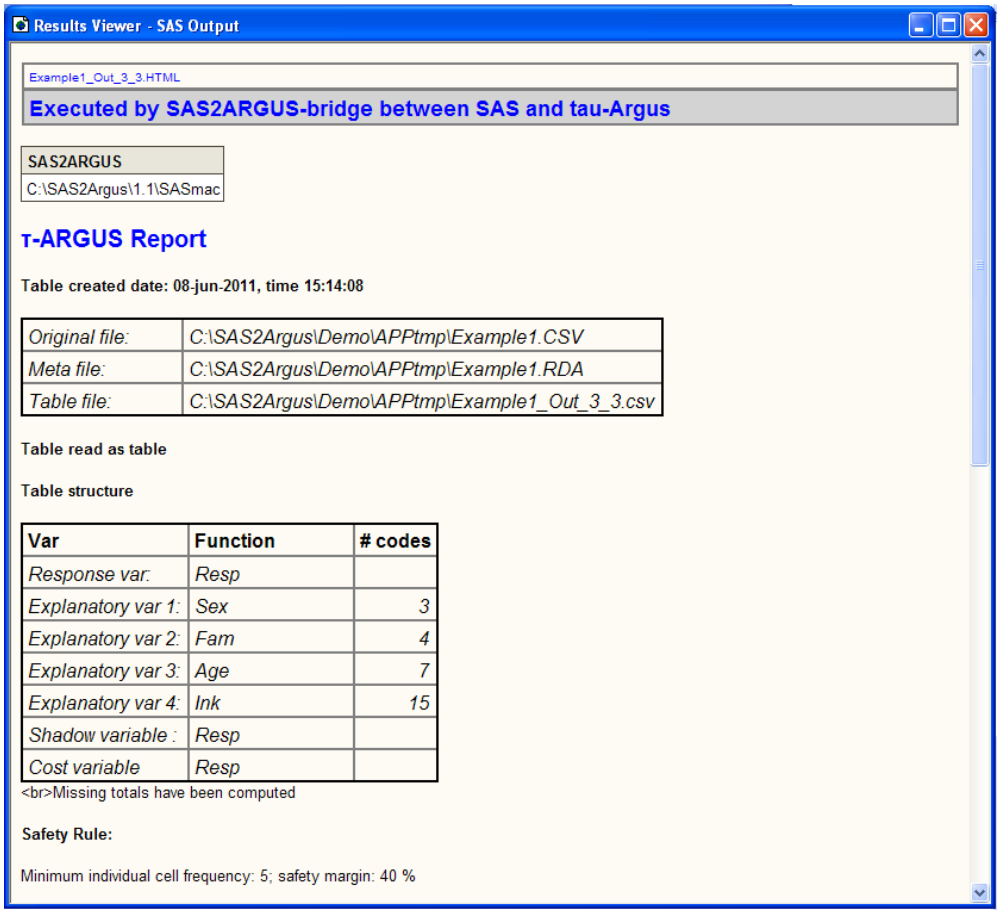
`Example1.csv` - The data file is tab-delimited
`Example1.rda` - Metadata file
`Example1.arb` - Command file

There are also some helpful macros that facilitates the opening and "to look at" both input and output files. `Open_Editor` opens any text file in the SAS Enhanced Editor:

```
%open_editor(Example1_out_5_1.rda)
```

Here, in this example, we open the file `Example1_out_5_1.rda` in the SAS Enhanced Editor. It is the metadata description for the named associated data file `Example1_out_5_1.csv` produced by τ-ARGUS. From the name, we can deduce that is a outputfil Type `5.1` from `example1`. That is the argument `Inter(1)` results in the 5th output format with the gross amount of information and status.

With the parameter `SAS=2` then the result is “returned” / imported to the SAS session and the produced data file can be found in the SAS Work library. The Result Report in the form of an HTML file from the τ-Argus, can be found in SAS internal browser (a bit depending on the settings of the SAS):



When parameter `Debug=1` is set then the `ARB` file (batch file) is also written in the SAS log:

```
NOTE: ===================================================================
NOTE: Macro: [FETCH_ARB]
NOTE: -------------------------------------------------------------------
NOTE: Here follows the established ARB-file for tau-ARGUS
NOTE: ===================================================================
//===================================================================
// This job based on dataset: APPdat.TauFreq
//-------------------------------------------------------------------
<LOGBOOK>         "C:\SAS2Argus\Demo\APPtmp\Example1.LOG"
<OPENTABLEDATA>   "C:\SAS2Argus\Demo\APPtmp\Example1.CSV"
<OPENMETADATA>    "C:\SAS2Argus\Demo\APPtmp\Example1.RDA"

<SPECIFYTABLE>    "Sex""Fam""Age""Ink"|"Resp"|"Resp"|"Resp",1

<SAFETYRULE>      FREQ(5,40)
<READTABLE>       1
<WRITETABLE>      (1,1,0,"C:\SAS2Argus\Demo\APPtmp\Example1_Out_1_0.csv")
<WRITETABLE>      (1,2,1,"C:\SAS2Argus\Demo\APPtmp\Example1_Out_2_1.csv")
<WRITETABLE>      (1,3,3,"C:\SAS2Argus\Demo\APPtmp\Example1_Out_3_3.csv")
<WRITETABLE>      (1,5,1,"C:\SAS2Argus\Demo\APPtmp\Example1_Out_5_1.csv")
NOTE: ===================================================================
NOTE: END of tau-ARGUS ARB-file
NOTE: ===================================================================
```

The log file from τ-Argus in also found/written in the SAS log.

3.1.2 Workflow

The process of protecting the integrity of tables, and the effort of work needed, depends on the initial layout of the tables. It can range from very simple work and already suitable oriented data, to quite complicated that brings in work to reorganize the data in a way that is suitable for disclosure control. A bit of "where do we come from" and "where are we going".

Variables needed in the various "roles" in the elimination process must be available and each cell needs to be described by properties that has a unique ID, that is the variables describing the dimensions need a unique value for each cell. Other properties may be in excess of this weight, cost, shadow variable, and so on.

If table data is not oriented in “row-dimension” you have to restructure the data. If we start from a table of income statistics, which contains both a variety of statistical measures, different dimension variables, partly overlapping, and a number of different income classes.

Initially, a table could look like this:

radtext	Antal totalt	Andel med inkomst	Median	Medel	kol5	kol6	kol7	kol8	kol9	kol10	kol11	kol12	kol13	kol14	kol15	kol16	kol17	kol18
Summa kvinnor och män	21241	68	167,6	246,7	6804	1307	738	786	916	1152	1673	2416	1770	1291	883	963	273	269
därav 20-24 år	1398	85,8	120,6	140,6	199	206	178	169	175	144	131	98	44	35	17	2	0	0
25-34	2694	91,2	203,1	222,7	237	154	139	174	228	298	348	400	294	178	132	89	11	12
35-44	3648	91,3	252	276	318	124	111	119	158	245	437	640	465	370	239	272	90	60
45-54	3745	89,7	267	297,5	384	99	66	97	123	212	383	687	528	352	266	334	104	110
55-64	4292	76,4	202,2	264,6	1012	259	136	162	182	215	350	571	427	348	223	260	64	83
65-	5464	14,8	10,6	71,8	4654	465	108	65	50	38	24	20	12	8	6	6	4	4
20-64	15777	86,4	222	257	2150	842	630	721	866	1114	1649	2396	1758	1283	877	957	269	265
Gifta/sambo	11398	72,9	192,5	264,2	3093	642	349	361	470	592	931	1405	1098	829	572	659	191	206
därav 20-24 år	103	92,2	115,5	125,3	8	15	24	13	13	10	11	5	1	1	2	0	0	0
25-34	1239	96,6	211,7	219,1	42	64	68	95	132	155	173	177	130	94	62	38	7	2
35-44	2334	95,8	272,5	284,3	97	61	66	79	97	174	283	413	322	266	168	197	65	46
45-54	2230	94,8	294,1	310,2	116	61	34	47	74	104	218	424	363	226	178	226	69	90
55-64	2699	79,9	218	272,8	542	163	93	83	119	126	226	374	274	236	159	192	48	64
65-	2793	18,1	14,2	78,7	2288	278	64	44	35	23	20	12	8	6	3	6	2	4
20-64	8605	90,6	250,4	276,2	805	364	285	317	435	569	911	1393	1090	823	569	653	189	202
Ensamstående föräldrar	1212	84,2	206,5	245,2	191	58	43	57	70	96	145	212	134	74	45	59	13	15
därav 20-24 år	28	92,9	71,4	76,9	2	10	5	3	7	1	0	0	0	0	0	0	0	0

If table data for disclosure control initially looks like this, where we for our example only have interest for the columns that are shaded (in grey), and holds information on the number of people in different income classes. These values are oriented in the column headed (from `kol5` to `kol18`):

	radtext	Tot	kol5	kol6	kol7	kol8	kol9	kol10
2	Summa kvinnor och män	21241	6804	1307	738	786	916	1152
3	därav 20-24 år	1398	199	206	178	169	175	144
4	25-34	2694	237	154	139	174	228	298
5	35-44	3648	318	124	111	119	158	245
6	45-54	3745	384	99	66	97	123	212
7	55-64	4292	1012	259	136	162	182	215
8	65-	5464	4654	465	108	65	50	38
9	20-64	15777	2150	842	630	721	866	1114
10	Gifta/sambo	11398	3093	642	349	361	470	592
11	därav 20-24 år	103	8	15	24	13	13	10
12	25-34	1239	42	64	68	95	132	155
13	35-44	2334	97	61	66	79	97	174
14	45-54	2230	116	61	34	47	74	104
15	55-64	2699	542	163	93	83	119	126
16	65-	2793	2288	278	64	44	35	23
17	20-64	8605	805	364	285	317	435	569
18	Ensamstående föräldrar	1212	191	58	43	57	70	96
19	därav 20-24 år	28	2	10	5	3	7	1
20	25-34	139	16	5	8	14	15	22
21	35-44	415	50	19	16	18	26	28
22	45-54	404	33	8	8	14	17	36
23	55-64	150	27	10	2	7	4	9
24	65-	76	63	6	4	1	1	0
25	20-64	1136	128	52	39	56	69	96

…In this scenario you need to restructure the data file - so that the cells can be identified by its properties.

We then get the entire table in list form where each cell is represented on one row:

	radtext	Sex	Fam	Age	Ink	Resp
1	Summa kvinnor och män	T	T	T	T	21241
2	Summa kvinnor och män	T	T	T	kol5	6804
3	Summa kvinnor och män	T	T	T	kol6	1307
4	Summa kvinnor och män	T	T	T	kol7	738
5	Summa kvinnor och män	T	T	T	kol8	786
6	Summa kvinnor och män	T	T	T	kol9	916
7	Summa kvinnor och män	T	T	T	kol10	1152
8	Summa kvinnor och män	T	T	T	kol11	1673
9	Summa kvinnor och män	T	T	T	kol12	2416
10	Summa kvinnor och män	T	T	T	kol13	1770
11	Summa kvinnor och män	T	T	T	kol14	1291
12	Summa kvinnor och män	T	T	T	kol15	883
13	Summa kvinnor och män	T	T	T	kol16	963
14	Summa kvinnor och män	T	T	T	kol17	273
15	Summa kvinnor och män	T	T	T	kol18	269
16	därav 20-24 år	T	T	20_24	T	1398
17	därav 20-24 år	T	T	20_24	kol5	199

`Radtext` is in the above example, although redundant information for τ-ARGUS, but otherwise the table is now usable as input to the disclosure process as set out above and used as input in our first example of a SAS2Argus call.

There are numerous techniques available to "tweak" the data and depends on the choice of tools, SQL (case, pivot ...), SAS (transpose, group data step ...).

We have now created dimensional variables, (from the only information available in `radtext)` and have introduced a constant `"T"` that denotes the totals, that is, the table marginal totals.

The overlapping age classes `(20-64)` are *not* managed. If we would handle this, which would be an example of an hierarchy and then this fact also had to be described for τ-ARGUS.

If we now define a call to the macro to perform a risk assessment of the table with the criterion that there must be at least 5 observations in each cell in the line-oriented data we now prepared. We define the name of the dataset and the following roles, rules and system parameters:

```
%sas2argus(InTable     = APPdat.TauFreq,
           Jobname     = Example1,
           Explanatory = Sex Fam Age Ink,
           Frequency   = Resp,
           SafetyRule  = FREQ(5,40),
           Out         = inter(1),
           RunArgus    = 1,
           SAS         = 2,
           Debug       = 1
)
```

Note that for an aggregated frequency table we don't need to enter `Response` variable. The `Frequency=Resp` is enough for to make an risk assessment based on just frequency.

System parameter/option `SAS=2` means that we also import importable data sets to SAS. The so called intermediate table is obtained as output with the argument: `Out=Inter(1)`, and as this table is suitable to import to SAS, after the execution, we import it and it will be found in SAS WORK and looks like this after execution:

	Sex	Fam	Age	Ink	Resp	Freqvar	Resp_shadow	Resp_cost	Statusvar	Lowerprotlevel	Upperprotlevel
89	T	T	65_99	T	5464	5464	5464	5464	S	1	1
90	T	T	65_99	kol5	4654	4654	4654	4654	S	1	1
91	T	T	65_99	kol6	465	465	465	465	S	1	1
92	T	T	65_99	kol7	108	108	108	108	S	1	1
93	T	T	65_99	kol8	65	65	65	65	S	1	1
94	T	T	65_99	kol9	50	50	50	50	S	1	1
95	T	T	65_99	kol10	38	38	38	38	S	1	1
96	T	T	65_99	kol11	24	24	24	24	S	1	1
97	T	T	65_99	kol12	20	20	20	20	S	1	1
98	T	T	65_99	kol13	12	12	12	12	S	1	1
99	T	T	65_99	kol14	8	8	8	8	S	1	1
100	T	T	65_99	kol15	6	6	6	6	S	1	1
101	T	T	65_99	kol16	6	6	6	6	S	1	1
102	T	T	65_99	kol17	4	4	4	4	U	1.6	1.6
103	T	T	65_99	kol18	4	4	4	4	U	1.6	1.6

Note the `Statusvar` with value `S=Safe` and `U=Unsafe` (when frequency is below 5).

The name of the output type, `Intermediate` suggests by the name, that this is the most appropriate format to work with, as you can either edit and send it back to τ-Argus or forward it in the process chain to set up the presentation tables.

3.1.3 Micro data Table – Magnitude table

The previous example was an aggregated frequency table. The example here is a magnitude table, which we let SAS aggregate before we continue with the disclosure. If we wish to apply the so-called `NK-rule, N` objects may not contribute more than `S%` of the cell contents, we also need to identify the major contributors in each cell first.

This can be accomplished using this syntax of `PROC MEANS`:

```
/*=========================================================================
 Summarize and select four TopN values for the explanatory variables.
 -------------------------------------------------------------------------*/
proc means data=APPdat.TAUmicro Missing CompleteTypes NWay NoPrint;
   class Age Omk;
   var Resp;
   output out=APPdat.TAUmicro_agg
          sum=Resp
          idgroup(max(Resp) out[4] (Resp)=Resp_TopN)/NoInherit;
run;
```

...where the four major donors will be written to the output dataset as "told" in the syntax. Or, if you have trouble remembering the syntax, use the macro `Calculate_TopN` (which is one of utility macros):

```
/*-------------------------------------------------------------------------
 Run the macro CALCULATE_TopN.
 -------------------------------------------------------------------------*/
%Calculate_TopN(Indata=APPdat.TAUmicro,
                ClassVar=Age Omk,
                Var=Resp,
                TopN=4);
```

By "leaning" against the metadata that we have available in SAS, we can define a format to tell how many decimal places we want to work with. Then the macro will "see" this and we get this into account within the specification when the metadata file is created:

```
/*=========================================================================
 Assign a SAS-format for "telling" SAS2ARGUS how many decimals we should
 deal with when establishing the files for tau-ARGUS.
 -------------------------------------------------------------------------*/
proc datasets library=APPdat NoList;
   modify TAUmicro_agg;
   format Resp: 8.3;
run; quit;
```

Then we execute the main macro SAS2Argus:

```
/*=========================================================================
 EXAMPLE 2 - Magnitude
-------------------------------------------------------------------------*/
options nonotes nosource nomprint nomlogic nosymbolgen;
%sas2argus(InTable     = APPdat.TAUmicro_agg,
           Jobname     = Example2,
           Explanatory = Age Omk,
           Response    = Resp,
           MaxScore    = Resp TopN_1 Resp_TopN_2 Resp_TopN_3 Resp_TopN_4,
           SafetyRule  = NK(3,90),
           Suppress    = GH(1,40,0),
           Out         = table() pivot(1) code(3) inter(1),
           RunArgus    = 1,
           SAS         = 2,
           Debug       = 1
)
```

Here we indicate, beside `Explanatory` and `Response`, also the with PROC MEANS produced `MaxScore` variables names. The `Safety rule` is here `NK(3,90)`, the so-called `NK-rule` stating that three companies may not contribute more than 90% of the contents of domain/cell. Secondary Suppression, `GH(1,40,0)`, is here done with the hypercube method (or GH-miter as it is called). Where `1` stands for "Table 1" (and is set in general as we execute only one table at a time via the macro). `40` meaning that the max precision should be at least 40% for any intruder to be able to reveal a value of a suppressed cell against the remaining margin totals.

Hypercube method together with the NET method, are the only methods available as a secondary suppression methods, if you do not have access to an optimizer. A purchased optimizer provides more options. The Hypercube method "puts out" a lot of table cells in the secondary suppression phase, as it works "mechanically", and is not recommended as first choice of method. The Modular approach is better, and to recommend, as it tries to find the optimal solution that supress least secondary cells in the secondary suppression.

3.1.4 Micro data Table – Magnitude table

A final example shows a micro-data table that contains population and the county, municipality, parish, `region(2 2 2)`[2] represent one hierarchy and age classes, `Age(&PATH_app\APPdat\Age.hrc)`, is the second hierarchy of classes. In the latter case is the prepared file that describes the age group hierarchy in a file with a name, along with the path specified as an argument (in parentheses). If `HierLeadString` not listed (as here) it is assumed to be `'@'`. Finally there is a third explanatory variable, gender `(Sex)`. The `Response` variable `(Count)` is in this case a column consisting of '1'.

```
/*-------------------------------------------------------------------------
 EXAMPLE 3 - HIERARCHY
-------------------------------------------------------------------------*/
options nomprint nosource;
%sas2argus(InData      = APPdat.Population,
           Jobname     = Example3,
           Explanatory = Region(2 2 2) Age(&PATH_app\APPdat\Age.hrc) Sex,
           Response    = Count,
           SafetyRule  = NK(3,75)|P(25,100,1)|FREQ(5,30),
           Suppress    = GH(1,30,0),
           Out         = table() inter(1),
           RunArgus    = 0,
           SAS         = 1,
           Debug       = 1
)
```

[2] No commas in the list of numbers!

Several `safety rules` can be specified as seen. See the τ-ARGUS-manual for a detailed description of the arguments that's are possible.

`RunArgus=0` means that only the text files that τ-Argus need to be produced. τ-Argus is not executed. These text files can be reused in the interactive interface.

`SAS=1` has no effect here since τ-ARGUS will not be executed because of the previous arguments. Would otherwise meant that the Result Report and the log had been "imported" and presented within the SAS session.

4 SAS2Argus - Structure

SAS2Argus is composed of a main macro that uses a set of "utility macros."

Macro	Description
Sas2Argus.sas	The main macro which the user calls with parameter set and executes
Macros that are called by the macro SAS2Argus:	
SAS2Argus_Help.sas	Prints usage description of the macro to the SAS log
Clean_Parameters.sas	“Cleaning up” parameters
Check_Parameters.sas	Checks the parameter values
Variable_Roles.sas	Sets variables roles
Variable_Properties.sas	Sets variables properties
Variable_Meta.sas	Checks that the specified variables exist in the specified data table (SAS, SQL, Excel ...)
Write_Datafile.sas	Creates text file from table data
Write_Jobfile.sas	Creates batch file
Fetch_Arb.sas	Includes batch file in the SAS log
Remove_File.sas	Removes any log file from previous runs
Read_Datafile.sas	Imports text file from the τ-ARGUS to SAS
Present_HTML.sas	Presents results report in SAS internal browser
Argus2SAS.sas	Imports the output of τ-ARGUS to SAS by executing Read_Datafile.sas and Present_HTML.sas. The inverse of the concept
Fetch_Log.sas	Incorporates the log from τ-Argus in the SAS log
Macro that controls the initialization of the SAS-session:	
System_Parameters.sas	A macro that controls the initial parameters required to set up the functionality in SAS
Other Useful using macros (utility macros):	
Calculate_TopN.sas	A utility macro which aggregates (PROC MEANS), and presents the largest contributors in each cell
Check_Dataset.sas	A utility macro to facilitate verification of the existence of a data set, view, table on SQL server ...
Check_Outstring.sas	A utility macro that controls that the right arguments is specified for the output of τ-Argus
Open_Editor.sas	A utility macro that opens any text file in the SAS program editor in SAS
Open_Excel.sas	A utility macro that opens any CSV file in Excel from SAS

4.1 Context

To describe the context, it is easiest to, in condensed form, present the main macro SAS2Argus and how it in turn executes the "underlying" utility macros for those who want to understand the function from a system perspective.

```
%macro SAS2Argus(

/*-----------------------------------------------------------------
 Need for help?
 -----------------------------------------------------------------*/
%if &help. %then %do;
  %SAS2Argus_help;
%end;

/*-----------------------------------------------------------------
 "Clean" the parameters
 -----------------------------------------------------------------*/
%let Explanatory = %Clean_Parameters(&Explanatory.,p);
...
...

/*-----------------------------------------------------------------
 Check parameters for data and that the dataset exists
 -----------------------------------------------------------------*/
%Check_parameters;

/*-----------------------------------------------------------------
 If parameter RUNARGUS is set to 2 => Rerun τ-Argus with already created
 files meaning that there is no need to establish any CSV or RDA or ARB-
 files. This is the scenario when the user could changed the status on
 certain domains (cells) from SAFE to UNSAFE or from UNSAFE to SAFE and
 with a new request to τ-Argus to handle secondary cell suppression.
 -----------------------------------------------------------------*/
%if &runargus. ne 2 %then %do;

  /*-----------------------------------------------------------------
   Establish the Metadata for the ROLES OF VARIABLES at macro invocation.
   -----------------------------------------------------------------*/
  %Variable_Roles

  /*-----------------------------------------------------------------
   Establish the Metadata for the ACTUAL VARIABLES in the dataset given at
   macro invocation.
   -----------------------------------------------------------------*/
  %Variable_Properties(&_dataset.)
```

```
/*-------------------------------------------------------------
   Check conformity between designated variable roles and actual variables
   in the dataset.
   -------------------------------------------------------------*/
  %Variable_Meta(&_dataset.)

  /*-------------------------------------------------------------
   Create the textfiles .RDA (meta) and .CSV (data ) according to the
   information established so far and found in _Variable_Meta
   -------------------------------------------------------------*/
  %Write_Datafile(Dataset=%bquote(&_dataset.),Datafile=&PATH_tmp.\&jobname.)

  /*-------------------------------------------------------------
   Create the Command file (BAT) for τ-Argus as a manifest of what we
   want τ-Argus to do (suffix .ARB)
   -------------------------------------------------------------*/
  %Write_Jobfile(Jobfile=&PATH_tmp.\&jobname..ARB)

 %end;


 /*-------------------------------------------------------------
  If debug=1 then include the ARB-file in the SAS log
  -------------------------------------------------------------*/
 %if &debug. %then %do;
   %Fetch_Arb(Arbfile=&PATH_tmp.\&jobname..ARB)
 %end;

 /*-------------------------------------------------------------
  Execute τ-Argus
  -------------------------------------------------------------*/
 %if &RunArgus. %then %do;

   /*-------------------------------------------------------------
    Remove the LOG for this "job"
    -------------------------------------------------------------*/
   %Remove_File

/*-------------------------------------------------------------
   Execute τ-Argus
   -------------------------------------------------------------*/
  data _null_;
   _cmd = """&PATH_exe"" ""&PATH_tmp.\&jobname..ARB""";
   call system(_cmd);
  run;
```

```
  /*-------------------------------------------------------------------
   If SAS=1 then "import" both TEXT files and HTML files produced by
   τ-Argus to this SAS-session.
   -------------------------------------------------------------------
   %Argus2SAS executes:
       %Read_Datafile - A macro that reads delimited (CSV) text files
                        with use of supplied metadata file (RDA) and
                        establish SAS datasets.
       %Present_HTML  - A macro that presents the HTML-file produced by
                        τ-Argus in the SAS internal browser.
   -------------------------------------------------------------------*/
  %if &SAS. %then %do;
    %Argus2SAS
  %end;

  /*-------------------------------------------------------------------
   If debug=1 then include the HTML-files from τ-Argus in the SAS
   internal browser and the LOG-file from τ-Argus in the SAS log
   -------------------------------------------------------------------*/
  %if &debug. %then %do;
    %Fetch_Log
  %end;

 %end;

%mend SAS2Argus;
```

All included macros are well commented in order to increase readability, make the code/function easier to understand and to facilitate system maintenance.