# 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.

[User and system description of SAS2ARGUS 1](#_Toc521074323)

[1.1 Recommended "pedagogy" 1](#_Toc521074324)

[1.2 Choice of method 2](#_Toc521074325)

[1.3 The purpose of the SAS macro 3](#_Toc521074326)

[1.3.1 Concepts 3](#_Toc521074327)

[1.4 The main macro - SAS2Argus 4](#_Toc521074328)

[2 SAS2Argus - Description 4](#_Toc521074329)

[2.1 Parameters 4](#_Toc521074330)

[2.2 Hierarchies 9](#_Toc521074331)

[2.3 Risk assessment - SAFETY RULE 10](#_Toc521074332)

[2.4 Secondary Suppression - SUPPRESS 12](#_Toc521074333)

[2.4.1 A priori file 13](#_Toc521074334)

[3 SAS2Argus - Usage 15](#_Toc521074335)

[3.1 SAS2Argus - Examples of syntax 16](#_Toc521074336)

[3.1.1 Micro data 16](#_Toc521074337)

[3.1.2 Aggregated Table 19](#_Toc521074338)

[3.2 Open Editor 20](#_Toc521074339)

[4 SAS2Argus - Structure 21](#_Toc521074340)

[4.1 Context 22](#_Toc521074341)

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.

## Recommended "pedagogy"

To understand the purpose and use of the SAS macro SAS2Argus, it is essential to know the function of τ-ARGUS and how it is used. Concepts defined and used in τ-Argus are also used in the SAS macro as far as possible in order to facilitate the understanding of how if "fits together". This description of the "bridge" between SAS and τ-Argus - how ambitious it would be - limps without the knowledge of the principles and methods used in τ-ARGUS.

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

If you need to get familiar with and understand the implemented methods, in order to choose the right one, we also recommend the methodological handbook (manual) where these methods are described.

## Choice of method

There is 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 ​​– belongs to 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, or to collapse groups and levels where the number of contributors are few. The advantages are several. The methods are easy to implement and provide understandable tables that can be summarized, and the technique is also easy to explain to the user. Sometimes it is not possible to redesign the tables, for example if the definitions of the tables are established in advance (by i.e. international standard) or that this approach gives too much information loss.

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

Choice of method for protection of respondents 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.

## 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 is what the SAS-macro does. Primary text files that need to be prepared to "put τ-Argus to work" are:

1. A 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 access to 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 only needs to add information about variable 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. As it is obviously simpler to do those tasks in SAS by defining labels and/or recode data, this possibility to label and recode is not implemented within the macro. There is another text file defined where you can set properties of the cells before secondary suppression is performed in an a priori file *(The a priori file).* All of these are described in the τ-ARGUS-manual.

### Concepts

To facilitate the mapping between the concepts used in τ-ARGUS and concepts defined in the SAS macro SAS2Argus, the same concepts have been used within the macro, although some terms may be unusual for SAS-users, such as explanatory[[1]](#footnote-1) . Similarly, all the suffixes in the file types that are defined and referenced in τ-Argus, are also used in the SAS-macro. This facilitates the reading of the τ-ARGUS-manual and the understanding of 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 consider these sub macros, but they are listed and described at the end of this document.

## The main macro - SAS2Argus

The user only needs to understand the use of the comprehensive SAS macro, SAS2Argus. This macro calls the execution of a sequence of other macros, which control syntax, generate 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 parameters are set and the function of τ-ARGUS, it is relatively easy to perform risk assessment and/or cell suppression with use of the macro. In place it is an effective “workbench” that facilitates 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.

# SAS2Argus - Description

Here is a description of the macro SAS2Argus. We describe 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 utilizes, cause this in itself describes much of the potential of the concept.

## 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 describe roles for variables, and variables that also must exist, or be established, in advance in order to make it possible to refer to them later. Some of these roles are unusual and rarely used. A number of parameters are assigned default values, if value is not specified explicitly. Required parameters are marked in **colour.**

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 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 in 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:   1. 0. Only text files are created by the macro. τ-Argus is not executed. 2. 1. Text files are created and τ-Argus is executed **(default)** 3. Don’t create any text file – only execute 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:   1. No additional information is written in the SAS log 2. 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 the execution is found in the SAS log. |
| **HELP** | Describes the macro and its parameters in the SAS log:   1. No information in the log **(default)** 2. 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**  New choices with version 4.0 | An option that controls imports from τ-ARGUS to SAS:   1. No import from τ-ARGUS to SAS **(default)** 2. Imports the results report in HTML format from τ-ARGUS and includes it in the web browser. 3. Imports the output from τ-Argus to the SAS WORK. 4. Imports the results report and imports the output to SAS work. |

|  |  |
| --- | --- |
| ***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 for frequency tables, aggregate data must at least have information about the frequency in each cell to be useful as input to risk assessment and suppression. For magnitude tables, we need the input from the largest contributors in each cell. |

|  |  |
| --- | --- |
| ***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](#_Risk_assessment_-) 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](#_Secondary_Suppression_-) where this parameter is described. |

The information in the safety rule and suppress "ends up" in the command file [ARB] and is used 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 add a description of how it is hierarchical in subsequent brackets, or specify in which text file that description can be found. Note that variable names are specified with spaces as separators if there is more than one in a list. See [Section 2.2 Hierarchies](#_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, τ-ARGUS uses the Response variable. |
| **Cost** | Identification of potential cost variable. When secondary suppression is performed, a cell with high cost is less likely to be suppressed compared to a cell with low cost. 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.** |
| **Missing** | Indicate if code for missing should be specified in the metadata file. **Default=0.** |

|  |  |
| --- | --- |
| ***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 incorrect results 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 names of variables that hold 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  It is recommended to restrict the use of setting a cell to protected. If you want to prevent that a cell becomes secondarily suppressed, give it a high cost instead. |
| **TotCode** | A constant that specifies/indicates which value that represent the total of an aggregate table. Default character is 'T'. |

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

The Status variable is created by τ-Argus when a risk assessment is made, by use of the rules specified in the parameter safety rule2, but can of course be 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. Used 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. |

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 chosen). The “easiest” way may be to experiment in order to understand all the different options for output. The most informative and useful is considered to be Intermediate (INTER).  **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 replace primary suppressed values ​​are given in parentheses after *Primary.* Characters that replace 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 formats has a historical explanation, as in the "τ-ARGUS-sphere", there are many stakeholders with different preferences. SBS is such a special format applied for *Business Statistics.* Most formats are impossible or inappropriate to import into SAS as a table. They are best suited for presentation 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. The intermediate format is also easy to tabulate in SAS.

## 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)

In 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

## 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, and suppress, with relatively simple tools/technology.

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 the τ-ARGUS-manual. There is no check of the specified arguments done ​​by the SAS macro. Information about any inaccuracies in the arguments can 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 separated by the **"|".** The following rules can be set: P, NK, ZERO, FREQ, REQ, WGT and MIS. They are described in the table below.

Additional arguments are typically set in subsequent brackets:

|  |  |
| --- | --- |
| **SAFETYRULE** | **Description** |
| P | Percent Rule where additional arguments are 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 | Zero-cells are set to unsafe. The additional argument in ZERO (ZeroSafetyRange) refers to the size of the safety margin. |
| FREQ | The Frequency Rule in which additional arguments are specified as FREQ (MinFreq, FrequencySafetyRange) where MinFreq specifies the minimum acceptable frequency and FrequencySafetyRange indicates the safety margin for not being able to derive the suppressed frequency. |
| REQ | "Request" rule - the request for confidentiality3. REQ(Percent1,Percent2,Safetymargin). For example, if an respondent accounts for 70% of the cell value and in that case has asked for protection. This requires an additional variable in the table that indicates which items that have requested /not requested confidentiality with value 1 or 0. Variable name (for the role) is indicated by the parameter Request . |
| MIS | Missing. MIS=0 (which is the default) means that the cells with a code for non-response are still regarded as uncertain whether any SafetyRule is violated. If MIS=1 then the cell is always considered as safe, if at least one contributor has a missing value, then it may be regarded that the contributor with missing value cannot be identified. In the SAS macro ‘9’ has been defined as representative value for missing value4. |
| 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 file5 is used to set the option that a particular cell is manually set 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 able to reveal another contributor´s 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 are then assumed to be applied to the individual level and the subsequent P and NK are 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 safety rule.

## Secondary Suppression - SUPPRESS

Here, the arguments that can be used for parameter Suppress are described. It has the same "syntax" as the SafetyRule, that is, additional arguments are given in subsequent brackets. Note that the first of these additional parameters specifies the table number (TabNo) and as the SAS macro only handles one table at a time, this additional parameter will 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 method available in τ-ARGUS without access to commercial optimizers. The secondary suppression is done "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 into several non-hierarchical tables, protects them, and finally composes a protected hierarchical table. With the MaxTimePerSubtable parameter it is possible to limit the optimizer to work with each sub table. The maximum time 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. Indicated in minutes. |
|  |  |
| **SUPPRESS** | **Description (cont.)** |
| NET | NET(TabNo) Network Solutions to be used for large two-dimensional tables with a hierarchy. Requires some special circumstances but no optimizer. |
| 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 |

### A priori file

An a priori file is a simple text file that can be created in an editor. The information in the A priori file is used *after* risk assessment, but *before* secondary suppression. The file updates include 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 decreases 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 be changed to unsafe.

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

It is recommended to be restrictive when setting cell status to protected. If the intention is to avoid the cell to be subject to a secondary suppression, it is better to assign a high cost for the cell. If, nevertheless, it would be secondarily 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.

# 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]

--------------------------------------------------------------------------\*/

%let PATH\_sys=&Path\_ini.\1.3;

%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 program used is 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 ODBC connection to SQL server.

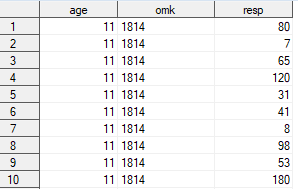
## SAS2Argus - Examples of syntax

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

### Micro data

The easiest way to use SAS2Argus is with micro data. You can use SAS2Argus both with magnitude and frequency tables.

The micro data should be prepared so that every observation (row) has a value for all the explanatory variables and for the response variable. Here is an simple example with two explanatory variables: age and omk. The response variable, which is a magnitude variable, is called resp. The microdata should look like this:



In Example 1 we show how the syntax in SAS2Argus could look like.

/\*==========================================================================

EXAMPLE 1 – Micro data

---------------------------------------------------------------------------

Magnitude table

Primary and secondary suppressed table built from micro data

--------------------------------------------------------------------------\*/

%***sas2argus***(InData = APPdat.TauMicro,

Jobname = Example1,

Explanatory = age omk,

Response = Resp,

SafetyRule = P(**20**,**1**),

Suppress = OPT(**1**,**2**),

Out = inter(**1**),

RunArgus = **1**,

SAS = **2**,

Debug = **1**)

For primary suppression we used the P-percent rule, with p=20%, and for secondary suppression we choose the Optimal method, with maximum execution time set to two minutes.

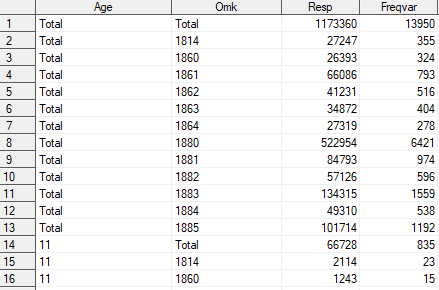
The naming of the files are based on Jobname, which is used as a prefix to all files "as part of an execution". The following files are produced in the above call for τ-ARGUS:

Example1.csv - The data file is tab-delimited

Example1.rda - Metadata file

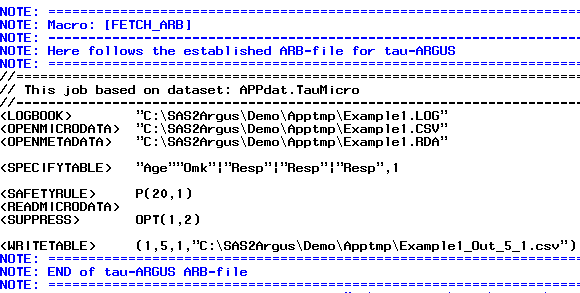
Example1.arb - Command file

With the parameter SAS=2 the result is “returned” / imported to the SAS session and the produced data file can be found in the SAS Work library. Here we see a part of that file from Example 1:



The so called intermediate table is obtained as output with the argument: Out=Inter(1). 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.

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



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

Next example shows a micro-data table with three explanatory variables. Region is a hierarchical variable with three levels: county, municipality and parish. Age is also a hierarchical variable where the hierarchical levels are described in a prepared file. If HierLeadString is not listed (as here), it is assumed to be '@'. Finally there is a third explanatory variable, gender (Sex).

/\*--------------------------------------------------------------------------

EXAMPLE 2 – Micro data - HIERARCHY

Magnitude table

Primary and secondary suppressed table built from micro data

--------------------------------------------------------------------------\*/

options nomprint nosource;

%***sas2argus***(InData = APPdat.Population,

Jobname = Example2,

Explanatory = Region(**2** **2** **2**) Age(&PATH\_app\Age.hrc) Sex,

Response = Resp,

SafetyRule = NK(1,50)|NK(2,80),

Suppress = GH(**1**,**40**,**0**),

Out = inter(**1**),

RunArgus = **1**,

SAS = **2**,

Debug = **1**

)

Several safety rules can be specified as seen. Here we have used a combination of two dominance rules. See the τ-ARGUS-manual for a detailed description of the arguments that are possible. We have used the Hypercube method for secondary suppression and also specified two output alternatives.

In next example we use the same micro data as in Example 2. But in this case the response variable is a frequency variable(Count), which is a column consisting of ‘1’. This will build a frequency table where every observation contributes with one. Here we use FREQ (3,30), meaning that there must be at least 3 observations in each cell, otherwise the cell is set to unsafe. With the second parameter we set the manual safety range to 30%.

/\*--------------------------------------------------------------------------

EXAMPLE 3 – Micro data - HIERARCHY

Frequency table

Primary and secondary suppressed table built from micro data

--------------------------------------------------------------------------\*/

options nomprint nosource;

%***sas2argus***(InData = APPdat.Population,

Jobname = Example3,

Explanatory = Region(**2** **2** **2**) Age(&PATH\_app\Age.hrc) Sex,

Response = Count,

SafetyRule = FREQ(**3**,**30**),

Suppress = GH(**1**,**40**,**0**),

Out = table() inter(**1**),

RunArgus = **1**,

SAS = **2**,

Debug = **1**

)

### Aggregated Table

We can also use SAS2Argus with an already built table. Here is an example of an aggregated magnitude table (InTable) where risk is assessed by the P-percent rule.

/\*==========================================================================

EXAMPLE 4 – Aggregated data

Magnitude table

Primary and secondary suppressed table built from aggregated table

--------------------------------------------------------------------------\*/

%***sas2argus***(InTable = APPdat. TAUmicro\_agg,

Jobname = Example4,

Explanatory = Age Omk,

Response = Resp,

MaxScore = Top1 Top2,

SafetyRule = P(**20**,**1**),

Suppress = MOD(**1**,**1**),

Out = inter(**1**),

RunArgus = **1**,

SAS = **2**,

Debug = **1** )

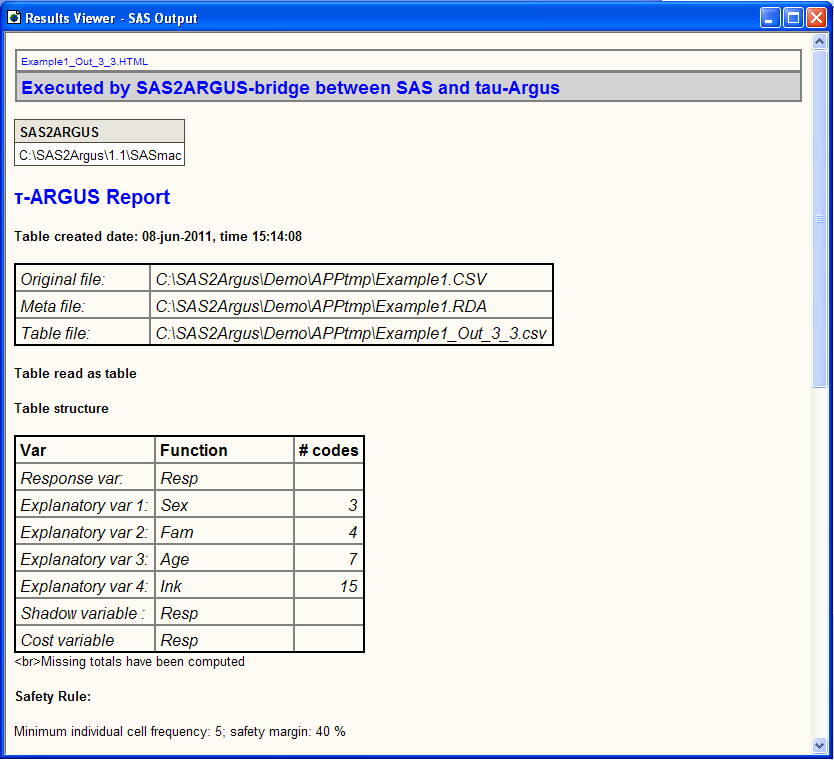
If you work with an aggregated frequency table instead of a magnitude table, it is important to indicate frequency, as this is essential information for τ-ARGUS for risk assessment.

## Open Editor

There are also some helpful macros that facilitates the opening of and looking 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 it is an output file 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. 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):



# 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 ...) |
| Variable\_Decimals | Fins out the exact number of decimals in data |
| 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. |
| Fetch\_Log.sas | Incorporates the log from τ-Argus in the SAS log |
| Check\_TauLog | Checks that the optimiser (Xpress) was run |
| ***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 are 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 |

## 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 chang 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.)

/\*-------------------------------------------------------------------

*Finds out the exact number of decimals in data*

--------------------------------------------------------------------\*/

**%Variable\_Decimals**(&\_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;

/\*-------------------------------------------------------------------

*A macro that checks the TauArgus log to see that we really did run*

*the optimiser (Xpress)*

*NOTE: Runs always independent of the settings of DEBUG*

--------------------------------------------------------------------\*/

**%Check\_TauLog**

%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.

1. The word used in τ-Argus for describing dimensions in the table [↑](#footnote-ref-1)