

ESSNet S D C

A Network of Excellence in the European Statistical System in the field of
Statistical Disclosure Control

Linking τ -ARGUS to SAS

Sarah Giessing

1 Introduction

This document offers some suggestions for how to facilitate exchanging data between SAS and τ -ARGUS.

The basic idea is to make ARGUS read/write SAS code files which the user then has to run in his SAS environment. Section 2 suggests how reading of a SAS input file could be organized. Section 3 is concerned with reading back ARGUS output into SAS format.

2 Reading SAS-input files

2.1 *The libname statement*

SAS organizes data in so called libraries. In principle, a library name is a short-name for a path/folder name. When opening a SAS project, the user starts by executing "libname" statements to assign short-names for those folders where the data he wants to work with, or which he intends to create shall be stored.

For simplicity, we should assume the ARGUS user to organize the SAS data sets he wants to use for data exchange with ARGUS in a single folder. He should provide a libname statement pointing to this folder in the ARGUS options screen.

Syntax for a simple libname statement

```
LIBNAME libref 'SAS-data-library';
```

libref

is a shortcut name or a "nickname" for the aggregate storage location where your SAS files are stored.

'SAS-data-library'

must be the physical name for the SAS data library. The physical name is the name that is recognized by the operating environment. Enclose the physical name in single or double quotation marks.

Examples

Example 1: libref in a Windows environment

```
libname essnet 'F:\myfiles\essnet';|
```

Example 2: libref for the same library in a Unix environment

```
libname essnet '/sas/user/myname/myfiles/essnet';
```

2.2 Name of the SAS input file

In order to work with a SAS input file, the user should select a SAS file (through ARGUS). The name extension of a SAS file is .sas7bdat.

2.3 Reading a SAS input file

In order to create structural information on the SAS input file that ARGUS can read, the user should run SAS-code (in his SAS-environment) which creates a meta-file with structural information on the SAS input file. This file must then be read into ARGUS. The information of this file should be used in the process of making the meta-file. Another SAS-code file will be created by ARGUS with the code for exporting the SAS data into .csv format. This code has to be executed again in the SAS-environment. The resulting .csv file then can be opened in ARGUS as usual.

Those steps could be handled by the following procedure:

- The user indicates if the SAS file is micro-data or tabular-data. He specifies a name for the file with the (Argus generated) SAS-code, and a name for the file with the (SAS generated) structure information (in the example of 2.3.1: “vars.txt”).
- ARGUS generates the file with the SAS-code, c.f. example 2.3.1
- The user runs this SAS code in his SAS environment.
- The user opens the resulting file (“vars.txt”, in the example) in ARGUS, i.e. ARGUS reads the file and creates a default .rda file. Using facilities of the Specify Metafile windows ARGUS the user can then edit this default .rda metafile.
- The user saves the modified metafile as .rda, and specifies
 - A name for a file with SAS-code that exports the data from SAS into .csv format, and
 - a name for this .csv-file
- ARGUS generates the file with the SAS-code (e.g. the macro-code followed by the macro-call), c.f. example 2.3.2
- The user runs the SAS-code in his SAS environment

2.3.1 Example: SAS-Code to retrieve structural information on the SAS input file

SAS-code

```
proc datasets library=essnet
              nodetails nolist;
```

Name of the libref given in the libname statement

```
contents data=i2_b4h
          out=work.vars
          (keep=name type length )
          varnum nods noprint;
```

Name of the SAS data file (without extension), c.f. 2.2

```
run;
proc export data=work.vars
```

SAS-data library, c.f. 2.1

```
outfile='/sas/user/myname/myfiles/essnet/vars.txt'
```

```
dbms=dlm;
```

```
delimiter=',';
run;
```

Name of the structure information file

Notes:

- 1.) In the case of the Example 1 libref statement of 2.1 the outfile parameter statement would be as follows:

```
outfile='F:\myfiles\essnet\vars.txt'
```
- 2.) Perhaps it would be safer to repeat the libref statement at the beginning of the code (e.g. before the proc datasets statement).

The Output: Structure information file vars.txt: Example

NAME,	TYPE,	LENGTH	dim3,	2,	1
dim2,	2,	5			
skip,	1,	8			
sales,	1,	8			
nace,	2,	7			

The structure information file contains one line with a header, and one line for each variable in the SAS-data set. Each line has 3 columns. Column “Name” gives the name of the variable, column “type” refers to the type (numeric: 1, character: 2) and column “length” is the variable length (only meaningful for character variables). In this example it is comma-separated, because in the proc export statement ‘,’ has been set as delimiter.

The default rda-file: Example

```
<SEPARATOR> ";"
<SAFE> "S"
<UNSAFE> "U"
<PROTECT> "P"
dim2 5
  <RECODEABLE>
  <TOTCODE> T
dim3 1
  <RECODEABLE>
  <TOTCODE> T
Nace 7
  <RECODEABLE>
  <TOTCODE> T
Sales
  <NUMERIC>
skip
  <NUMERIC>
```

Variable lengths for explanatory variables as given in the vars.txt information file

The edited rda-file: Example

```
<SEPARATOR> ";"
<SAFE> "S"
<UNSAFE> "U"
<PROTECT> "P"
dim2 5
  <RECODEABLE>
  <TOTCODE> T
dim3 1
  <RECODEABLE>
  <TOTCODE> T
Nace 7
  <RECODEABLE>
  <TOTCODE> T
  <HIERARCHICAL>
  <HIERCODELIST> "nace.hrc"
  <HIERLEADSTRING> "@"
Sales
  <NUMERIC>
```

2.3.2 Example SAS code to export data from SAS

The file with SAS-code to exports the data from SAS into .csv format should contain the macro-code first, followed by the statement for the macro call.

a.) Example for macro-code

```
%macro SAS2tab(myfile);
data _null_;
set essnet.i2_b4h end=EFIEOD;
mf=symget('myfile');
%let _EFIERR_ = 0; /* set the ERROR detection macro variable */
%let _EFIREC_ = 0; /* clear export record count macro variable */
file outdat filevar=mf delimiter=';' termstr=CRLF DSD DROPOVER
lrecl=32767;
format dim2 $5.;
format dim3 $1.;
format Nace $7.;
format sales best15.;
do;
EFIOUT + 1;
put dim2 $ @;
put dim3 $ @;
put Nace $ @;
put sales;
end;
if _ERROR_ then call symputx('_EFIERR_',1); /* set ERROR detection
macro variable */
if EFIEOD then call symputx('_EFIREC_',EFIOUT);
run;
%mend;
```

Name of the SAS data file
(without extension), c.f. 2.2

Separator for the csv-file

Variable lengths for explanatory
variables as stated in the (edited)
.rda -file.
Put **best15.** with numeric
variables

Note: A “\$” in the “put” statements indicates that the respective variables are numeric. “@” is a line-hold specifier, e.g. don’t use the line-hold specifier at the end of the last put-statement in the list.

b.) Example macro-call

- In Windows environment:
%SAS2tab('F:\myfiles\essnet\i2_b4h.tab');
- In UNIX environment:
%SAS2tab('/sas/user/myname/myfiles/essnet/i2_b4h.tab');

Name of (output) csv file

Note:

- 1.) Put the macro-call after the %mend statement of the macro-statement.
- 2.) Eventually repeat the libref statement between the %mend statement and the macro-call.
- 3.)

3 Reading ARGUS output into SAS-files

The following procedure could be used to read ARGUS output back into SAS:

- The user selects an ARGUS output format and specifies

- A name for a file with SAS-code that imports the data from ARGUS into SAS
- a name for the SAS-file
- if he wants the SAS-file to be stored temporarily or permanently
- The user runs the SAS-code in his SAS environment

3.1 Example SAS code to import data into SAS

The file with SAS-code to import the data from SAS into .csv format should contain the macro-code first, followed by the statement for the macro call.

a.) Example for macro-code

This is an example to import data from the ARGUS format “CSV file for pivot table, AddStatus”.

Only non-zero cells are imported. Total codes are set to the codes given in the corresponding .rda-file

```

%macro CsvStrich2SAS(mydirectory,myfile,mysasfile,perm);

proc format ;
    invalue strich  '-'=0 other=best32.;
run;

%if &perm %then %do;
libname essnet  '/sas/user/myname/myfiles/essnet';
%let lib=essnet;
%end;
%else %let lib=work;

%let myfile=&mydirectory.&myfile;

data &lib..&mysasfile  ;
mf=symget('myfile');
%let _EFIERR_ = 0; /* set the ERROR detection macro variable */
%let _EFIREC_ = 0; /* clear export record count macro variable */
infile indat filevar=mf delimiter = ',' termstr=CRLF MISSOVER DSD
lrecl=32767 firstobs=2 ;
informat dim3 $5.;
informat dim3 $8.;
informat nace $5.;
informat value strich. ;
informat status best2. ;

input
  dim2 $
  dim3 $
  nace $

```

Libname statement, c.f. 2.1

Separator of the csv-file

Variable lengths for explanatory variables as stated in the (edited) .rda -file.

```
value
status ;

if dim2="Total" then dim2="T";
if dim3="Total" then dim3="T";
if Nace="Total" then Nace="T";

if (status ne 14) then output; /*output for non-zero cells only*/
if _ERROR_ then call symput('_EFIERR_',1); /* set ERROR detection
macro variable */
run;
%mend;
```

Insert total codes as
stated in the .rda

b.) Example macro-call

- In Windows environment:

```
%SAS2tab(F:\myfiles\essnet\,i2_b4h_mitp_it1.csv,i2_b4h_mitp_it1,0);
```
- In UNIX environment:

```
%SAS2tab(/sas/user/myname/myfiles/essnet/,i2_b4h_mitp_it1.csv,i2_b4h_mitp_it1,0);
```

Path to SAS-data library, c.f. 2.1

Name of SAS output file

Name of ARGUS csv file

Note:

- 1.) Put the macro-call after the %mend statement of the macro-statement.
- 2.) In this example the output SAS-file will contain the total-codes given in the (edited) .rda (instead of "Total").
- 3.) The example assumes that the user has chosen to save the SAS-output file only temporarily (perm=0).
- 4.) The proc format statement defining the format "strich", together with the informat setting for the variable VALUE is necessary to replace the missing symbol "-" of the ARGUS output by 0-values. If ARGUS would directly put in 0-values for zero or missing data of the response variable, this proc format statement would not be needed. The informat for variable VALUE should then be set to best 32. directly (e.g. the corresponding statement in the macro would be " informat value best32. ; ")