

Is Your Output Telling the Truth? Tips and Tricks in Verifying SAS Outputs

Angelo Tinazzi, SENDO-Tech S.r.l., Milan, Italy
Sonia Colombini, SENDO-Tech S.r.l., Milan, Italy
Lisa Comarella, CROS NT, Verona, Italy
Marta Zanus, CROS NT, Verona, Italy

ABSTRACT

The process of producing tables, listings and graphs (TLGs) for a study, either for supporting report writing or scientific papers, is made of several steps, including the production of well designed specifications (Statistical Analysis Plan, Derived Variable specification, etc.). Each one of these steps is performed by different roles (programmer, eventually an output reviewer, statistician, report writer, principal investigator, etc.). This could be enough to guarantee a good quality of the output results.

However, because of the peculiarity of different studies and the complexity of clinical databases (nr. of tables, variables, complex tables relationship, data-conventions, etc.), it is not always the case that TLGs are delivered 100% error free.

The single programmers' skills and (clinical) knowledge/understanding are vital vehicles to ensure that TLGs are well programmed and these can be only achieved with continuous professional update and of course with experience gathered over the years. These can not be replaced by a single article or a presentation. However this paper will provide a possible 'output-verification' approach based on our experience and suggest a possible grid/checklist of single activities and steps to be performed before releasing a single output.

It is not the purpose of this paper to discuss the system validation of a statistical analysis reporting system.

INTRODUCTION

Producing analysis outputs by means of tables, listings or graphs, can be considered an 'art'. Delivering reliable results remains the main objective of such art, but the environment has changed. As in all the steps of the drug development process, the new regulatory requirements have also impacted the "output and analysis programming" processes.

The available SAS® literature, mainly from SAS User-Groups, is full of examples and most of the regulatory implications have already been addressed, as well as specific suggestions for SAS programmers, and in particular:

- the requirements for System Validation [1] [2] [3]
- the implication of the 21CFRPart11, and in particular the change control management [4]
- regulatory agency guidelines for statistical tables layout defining for example margins and font size [5]
- experiences and suggestions for 'Good Programming Practice' [6]

Furthermore, possible solutions for a regulated programming environment have also been presented [7] [8].

However only in a few cases the importance of data knowledge [9] and the discussion on how to do a proper "Quality Control" (QC) of the output, have been addressed, except for some technical solutions for making some sort of automatic QC [10]. Finally, an additional important issue is the recruitment and proper training of SAS programmers [11] [12].

Though all the arguments treated in the above mentioned articles are important for creating a regulated programming environment, are they enough to guarantee that the data analysis outputs are reliable and provide the correct answers?

SAS PROGRAMMING AND OTHER STANDARD REQUIREMENTS

We like to enforce the requirements for SAS Good Programming Practice (SAS GPP) already reported in most of the articles referenced in the 'Introduction' section; Table 1 reports the most important ones (for an extensive and accurate review of SAS GPP, see Frank Dilorio book [13]). In addition, the following aspects have to be considered:

- repetitive tasks, wherever possible, should be implemented as a macro routine; when macros are generally applicable, so that they can be used in other projects also by other programmers, a macro library can be created (and maintained) [14]
- every organisation should have Standard Operating Procedures (SOPs) in place and possibly set-up its own standard output templates for Tables, Listings and Graphs
- there should be adequate 'functional requirements' specified in Statistical Analysis Plan (SAP), including the

PhUSE 2006

definition of Derived Datasets and algorithm specifications for derived variables.

- two separate instances for test and production environment should be maintained and it is suggested to have a system, for example a Concurrent Version System – CVS [15], enabling the committing process (moving a program from test to production environment). However, if you are not able to implement such a system, or even better an Output Reporting System Application, you should at least maintain a version of each program and changes appropriately documented (figure 1).

Rule	Description
General	<ul style="list-style-type: none"> • Set-up a standard folders structure • Document your program by means of standard headers (see example in figure 1) • Make appropriate comments to the most relevant sections of your program • Maintain appropriate documentation of testing (e.g. SAS log and Output)
Coding Conventions	<ul style="list-style-type: none"> • Use indentation, blank lines and spaces to arrange the code clearly • Write only one SAS statement per line • Do not reuse a variable name in a data step • Do not reuse a dataset name in a SAS program (or project) • Use meaningful variable and dataset names (avoid for example dataset name such 'one' or 'a'). • Apply standards naming conventions, such as CDISC model, or company standards • Chose data type carefully in order to avoid the risk of truncation • Use RUN and QUIT at the end of each data or proc step • Delete temporary datasets after programming execution • Do not overwrite input datasets • Do not use SAS keywords (e.g. statement) for variable and dataset names • Set-up main variables characteristics by means of label, length and format (e.g. with an ATTRIB statement) • Do not 'hard-code' data on SAS programs • Keep only those variables you need, by either KEEPing or DROPPing variables
Data Control	<ul style="list-style-type: none"> • Debug your program (and data) to show intermediate results, by using PROC PRINT or PUT statement • Prevent data-driven issues, by executing code conditionally (e.g. division by zero)
Additional Common Tips	<ul style="list-style-type: none"> • Carefully evaluate the use of mathematical operators versus functions (e.g. + operator versus a SUM function)

Table 1: Good Generic and Specific SAS Programming Standard Rules

OUTPUT VERIFICATION APPROACHES

INDEPENDENT REPRODUCTION OF STATISTICAL ANALYSES

An independent reproduction of statistical analyses is like the double data-entry process. In this method an independent statistical programmer reproduces the set of TLGs.

A 'primary' programmer should program all the outputs applying the complete set of layout specifications defined for the final outputs, while the 'secondary' programmer (back-up programmer) can make quick programming avoiding adding additional SAS lines of code to control the output layout. Outputs produced by the two programmers, are then compared in order to verify that both programmers obtained the same results.

Of course this technique may not be applicable in all situations. Small companies for example may have limited resources and may not have the possibility to allocate more than one SAS programmer to the same project; therefore alternative techniques should be put in place.

CRITICAL REVIEW OF OUTPUTS

Just as the Quality Assurance Unit make independent reviews of the conduct of a given study, a lead programmer/lead analyst (or even a statistician) makes an independent Quality Control (QC) of the outputs produced as well as of the program logs. An additional step in the QC process, can be the code review to verify that programs are written according to coding standards and company policies.

It is also recommended to have a person with business intelligence to QA the outputs. For instance, the report writer or anyone with clinical expertise who is responsible for interpreting and using the data (e.g. for a publication in scientific

PhUSE 2006

journal), can assess if the value of average dosage of certain drug is correct or if the incidence rate of a particular adverse event is in line with the safety profile of the drug being studied.

```
*-----*;  
* Program ... : LOCSET.SAS *;  
* Scope ..... : Load User Settings *;  
* Version ... : 1.1 *;  
* Author .... : Angelo Tinazzi - SENDO Tech S.r.l. *;  
* Date Created : 27DEC2004 *;  
* Project .... : Final Analysis of Study SENDO10A289 *;  
* Macros Used. : %READ *;  
* Input ..... : *;  
* Output ..... : *;  
*REVISION HISTORY: *;  
*-----*;  
*Version |Date |Author |Change Description *;  
*-----*;  
*1.1 |31JUL2006|Angelo Tinazzi|Added setting of macro variables *;  
* | | |such as macro for HRPLOT *;  
*-----*;
```

Figure 1: A Possible Standard SAS Program Header

SOME USEFUL METHODS TO VERIFY SAS OUTPUTS

TAKE THE FULL CONTROL ON DATA

The risk of obtaining misleading results due to lack of understanding of the input data is high. Statisticians, Programmers and Lead Analysts, must be well aware of the possible contents of each variable, of the rules applied to data collection and management and of any ad-hoc convention applied in a later phase of the study. In fact, Data Management conventions are often source of mistakes because in some cases they allow to collect data that were not originally expected and therefore they are often not self explanatory; knowing in advance what conventions have been applied can reduce the risk of mistakes deriving from wrong assumptions on the data collected. For example assumptions for calculating times when conventions for reporting incomplete dates have been applied or, in oncology, when assumptions are applied to determine the number of anti-tumor treatments when multiple drugs are given concomitantly or in sequence as part of a unique regimen.

GIVE YOUR DATA THE LAST CHANCE TO BE CHECKED AND CLEANED

Although you have established a good Data Management Process within your organization, sometimes data are not 100% clean when passed from Data Management to Statistics (Output Programming). Data issues can still exist by the time data have been locked, or extracted from the Clinical Database to SAS for an interim analysis, and they may possibly later be detected by SAS programmers or by whoever is performing the review of the outputs.

OUTPUT LAYOUT ACCORDING TO PRE-DEFINED SPECIFICATIONS

☺ USEFUL PROGRAMMING TIP

A good programming approach is to code metadata, for example by means of macro variables, so that repeated terms are consistent throughout the outputs. For example you can assign a macro variable to each population label and use it whenever you need (for example in a TITLE statement).

```
%LET POPITT=ITT Population;  
title1 "Table 1 : XXXX";  
title2 "&POPITT";
```

CONTROLLING DENOMINATORS

It is important to identify soon in your set of TLGs, population denominators, such as number of patients enrolled, or number of patients randomized for each treatment arm, as well as specific population setting, such 'Intention to Treat', 'Efficacy Population', 'Safety Population'.

☺ USEFUL PROGRAMMING TIP

```
***Population denominator;  
data _null_;  
set pop end=last;  
retain denitt deneval densaf 0;  
if itt='Y' then denitt=denitt+1;  
if eval='Y' then deneval=deneval+1;  
if saf='Y' then densaf= densaf +1;  
if last then do;  
call symput('denitt',denitt);  
call symput('deneval',deneval);  
call symput('densaf',densaf);  
end;  
run;
```

STRATIFICATION CONSISTENCY

Stratum, such dose levels or treatment arms, should be defined and maintained consistently across outputs and in accordance with SAP.

CONTROL LISTINGS

A control listing is a good and simple tool to verify algorithms or mathematical calculations. This tool is especially useful when algorithms are complex, made of several steps and performed across visits or in any case at different periods.

For example in oncology a common approach to evaluate the treatment compliance is the calculation of the dose-intensity and relative dose-intensity [16]; the algorithm is described step by step in table 2.

Calculate the BSA (m2) (Weight(kg) ^{0.425} *(height(cm) ^{0.725})*71.84)/10000
Calculate the Dosage (mg/m2) Total Dose (mg) / BSA (m2)
Calculate Duration of Period/Cycle (days) (Date of first inception at cycle n - Date of first inception at cycle n-1) + 1
Calculate the Dose-Intensity (mg/mw/week) (Dosage (mg/m2) / Duration (days))*7
Calculate the Dose-Intensity and Relative Dose-Intensity (%) (Dose Intensity/Intended Dose Intensity)*100

Table 2: Dose-Intensity and Relative Dose-Intensity Algorithm

An ideal control listing for easy check of each of the above steps, should report all the calculations so that both errors in the algorithm or data-driven errors can be 'easily' detected (figure 2).

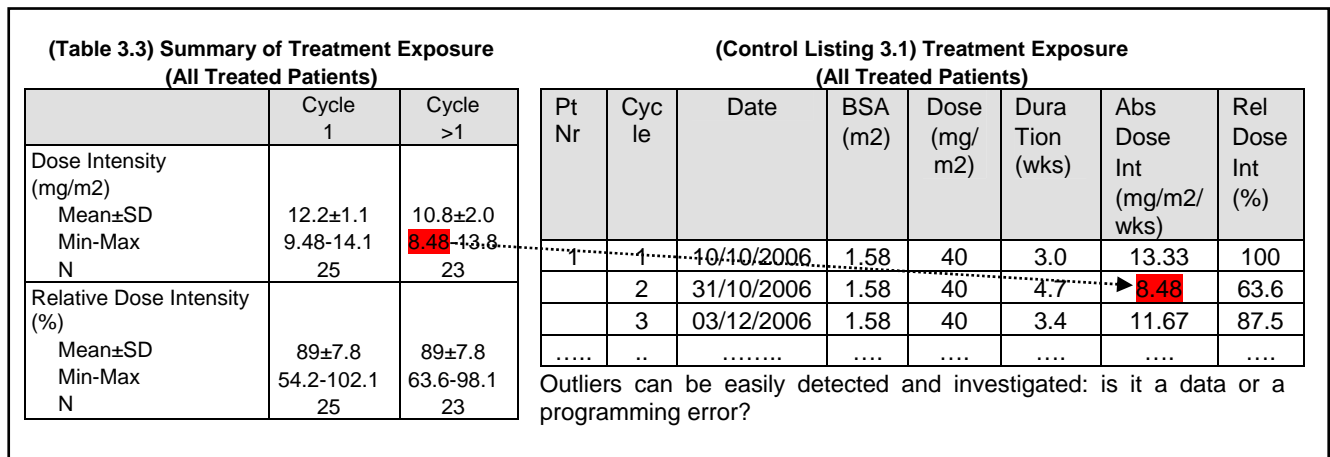


Figure 2: Checking summary results with control listing

PLAN CROSS TABLES CHECKS

Plan in advance relationship across tables and listings. Tables sharing some data should be cross-checked. For example in a safety analysis a table containing an overview of safety profile by means of summary of number of adverse events, should be checked against the table reporting the details of adverse events by MedDRA® System Organ Class and Preferred Term.

Table 3 reports an example of possible pre-defined cross table checks.

In the first example table T3.1 should report statistics on treatment compliance by cycle, from cycle 1 to cycle N, where N is the maximum number of cycles administered, which is a data reported in table T1.2.

In the second example T4.2 reports summary results from a survival analysis; in this table the nr. of events (deaths) should be equal to the nr. of deaths reported in table T4.1. If any difference is found, further investigations are needed in order to verify if there is any subject died without knowing the date, resulting in a 'time' variable not appropriately calculated (is missing).

TLG Number	Title	Cross Table Check
T1.2	Maximum Number of Cycles Received	
T3.1	Treatment Compliance per Cycle	Check with T1.2, for maximum nr. of cycles received
T4.1	Summary of Reason for Death	
T4.2	Overall Survival	Check with T4.1, for total number of deaths by group. If different, checks for missing in 'time' variable
T5.1	Summary of Haematological Toxicity Severity	
T5.2	Haematology Time to Recovery	This table is applicable where haematological toxicity is present and that were followed until recovery. The number of cases included in T5.1 must be equal to the number of cases in T5.2 with toxicity grade > 0.

Table 3: Example of plan of cross-tables checks

An additional cross table check is the one between table T5.1 and table T5.2. In this case haematological safety profile is investigated by means of worst severity CTC grade (a grading system ranging from 0 to 4, where 4 is the worst severity grade) and time to recovery. Table T5.2 can be considered a sort of an in-depth analysis to better investigate the analysis already performed in table T5.1; while with table T5.1 we want to see how severe the toxicity was, with table T5.2 we also want to see the duration of such toxicity (that is time elapsed between the toxicity started, that is when toxicity grade is >0, and the day when haematological parameter was recovered within laboratory normal range). Therefore, the number of cases included in table T5.2 must be equal to the number of cases in table T5.1 with grade>0. It may happen that table T5.2 includes less cases than those with grade>0 in table T5.1. In this case a control listing showing date of sampling and toxicity grade allocated to each result, can be of help in understanding where the 'mistake' is; by using this control listing we may discover that some toxicities have not been followed until resolution (grade > 0 at last assessment), therefore the time to recovery was not calculated.

PLAN AND DOCUMENT THE REVIEW PROCESS

The methods reported so far, by means of suggestions and tools, need to be of course planned in advance for each output you expect to program. To help the review process of the outputs, it is useful to have a possible pre-defined grid of items to be verified; the same grid can be used to document the review process applied to each output (program). Table 4 is an example of grid-list of possible items to be verified.

WANT TO CHECK MORE?

In many situations simple statistics such as mean, minimum and maximum, or numbers and percentages, can be either calculated ad-hoc by a SAS data-step or obtained from a SAS procedure output (e.g. statistics obtained from a PROC SUMMARY output dataset). In this circumstance it is necessary to verify even a simple mean value and it is also suggested to code and validate a SAS macro routine performing such operation, so that the validation can be done once.

But what should we do when a result is obtained entirely from a SAS standard package procedure? Should we verify for example that the log-rank test obtained from a PROC LIFETEST is appropriately calculated?

We 'guess' that the answer from a company who is going to submit data to a regulatory agency to get a new drug approval can be "I want to manually check every single number I'm going to submit", even a mean calculated with a standard PROC MEANS or even worst an odds ratio obtained from a PROC PHREG for multivariate survival analysis. It is our opinion that SAS is enough "tested" to guarantee our 'business' even in such extremely important situations; we should of course establish some sort of 'boundary' beyond which no further verification are needed.

However, back to the need of having standards in place (by means of SOP or company policies for SAS GPP or Statistical Analysis), it is suggested to validate the company approach to the most common analysis methodologies applied, as in the example of the multivariate survival analysis.

For example in oncology, where 'time-to-event' outcomes analyzed by means of multivariate survival techniques are commonly used (e.g. Cox analysis using PROC PHREG), we may implement a company policy detailing the way the programmer or the statistician have to code a PROC PHREG so that the multivariate survival analysis model is appropriately processed with no or less risk of getting inappropriate results due to inappropriate model selection. In this situation a good approach is to validate the PROC PHREG code with literature data, so that you are able to repeat standard analysis models as in the literature. Figure 3 reports a potential coding approach for multivariate survival analysis using the Cox model; this model of code has been validated using literature data (those reported in the reference book) and results compared with published results. If the programmers, or the statisticians, while performing an analysis of a 'time-to-event' outcome apply the model reported in figure 3, we may have enough 'warrant' that the results are obtained according to company policies and, more important, according to literature standards.

CONCLUSIONS

"System Validation" of output reporting systems is crucial for the quality of the biometric processes. Moreover, the implementation of standards for output reporting together with standard data-dictionary for data management (including standard forms, query programs, etc.), are a good starting point to guarantee that the last step of a clinical trial, that is producing and communicating the final results, is performed accurately.

Though we are not that far from an ideal world where 'few pushes on the keyboard' will make possible the set-up of everything is needed for managing and analyzing clinical study data, it is our opinion that programmers background and skills, as well as the ability and the experience gathered on the field, remain the key aspects to guarantee the quality and the 'trust-ness' of data outputs, by means of tables, listings or graphs.

	Item to be Verified	Description
LAYOUT	Formatting	Margins are set-up according to regulatory requirements, such as paper margins, font-size and style, etc.
	Tiles and footnotes	Titles, footnotes and notes according to specifications (e.g. SAP)
	Column Header	Misspelling and row splitting
	Variable Formatting Value	Verify that all variable values are appropriately decoded
	Page Break	Verify that summary data of the same group are not split across multiple pages. Whenever possible, try to control output appearance
	Sorting Order	Verify that sorting order is appropriate according to pre-specification and consistent according across tables and listings (e.g. dose level, investigator site, patient nr.)
POPULATION	Denominator and %	Total numbers of subjects in each group is accurate. Percentages are in line with denominators
	Data Sub-setting	Check specific population setting, for example 'Treated Male Patients'
SAS LOG	Errors	Check and resolve all errors raised in the SAS Log
	Warnings	Check warning messages. The following warnings can be accepted but they need to be verified: <ul style="list-style-type: none"> • SQL Join, when the same variable are present in both tables (datasets) and a '*' operator is used • Inappropriate DROP and KEEP • PROC APPEND warning message when the two datasets have not the same characteristics
	Notes	<ul style="list-style-type: none"> • Missing values generated • Mismatched data types • MERGE messages reporting numbers of observations involved in the MERGE statement
	Variable Un-initialized	If any, check that they are not used in any expression/calculations.
	Macro	Make sure there are not unresolved macro variables
OUTPUT CONTENTS	Outliers	Minimum and Maximum values are in line with variable 'domain'.

Table 4: Grid List of Items to be checked while reviewing SAS Outputs

```
proc phreg data=<input dataset>;
  model <time variable>*< censor variable>(<censor value>) =
    <independent variable 1> <independent variable 2> .....
    <independent variable n> / rl ties=exact selection=b;
run;
```

Figure 3: A PROC PHREG Company Model

REFERENCES

1. Howard N., Gayari M. Validation, SAS, and the Systems Development Life Cycle: An Oxymoron?. PharmaSUG, Paper DM09, 2000
2. Truong S. Effective ways of validating SAS Programs. PharmaSUG, Paper HW07, 2004

PhUSE 2006

3. Lawton A. Defining a Validation process for end-user (data manager/statisticians) SAS Programs. PharmaSUG Paper FDA058, 2003
4. Garbutt D. Implementing CFR 21 part 11 for SAS without tears or joins. PhUSE, Paper AS09, 2005.
5. Light S., Gilbert P. Using SAS to Enhance Clinical Data Summaries: Meeting eSub Guideline. SUGI 27, Paper 116-27, 2002
6. Adams J.H. Good Programming Practice in Clinical Trial – a Check Program. PharmaSUG, Paper AD17, 2006
7. Williams T. A programming development environment for SAS programs. PharmaSUG, Paper AD011, 2003.
8. Gerber P., Ludwig M. QualityView – a program database and validation documentation tool. PhUSE, Paper RC03, 2005
9. Chuang S. Know your data while writing the SAS code. PharmaSUG, Paper DM12, 2005
10. Morrill J, Austin D.J. Consistency Check: QC Across Outputs for Inconsistencies. PharmaSUG, Paper TT14, 2006
11. Zirbed D. 10 Things SAS Programmers Don't Know – But Should. SUGI 27, Paper 240-27, 2002
12. Howard N. The Ultimate Match Merge: Hiring the Best SAS Programmer. SUGI 25, Paper 245-25, 2000
13. Dilorio F.C. The Elements of SAS Programming Style. Published by SAS Institute, 2001. An articles is also available at <http://www.codecraftersinc.com/pdf/Style.pdf>
14. Mo A. Developing, Managing and Evaluating a Standard Macro System. SUGI 31, Paper 018-31, 2006
15. Williams T. Versions Control on the Cheap. A User-Friendly, Cost-Effective Revision Control System for SAS. SUGI 31, Paper 014-31, 2006
16. Longo D.L. et al. The Calculation of Actual or Received Dose Intensity. A Comparison of Published Methods. Journal of Clinical Oncology, vol 9, no. 11, pp 2042-51, 1991

ACKNOWLEDGMENTS

We would like to thanks Irene Corradino for her valuable input and Daniela Nava for her review.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author at:

Angelo Tinazzi

SENDO Tech S.r.l.

Via Visconti di Modrone 12

Milan - 20122

Office Phone: 0039 02 76420426

Fax: 0039 02 76017484

Email: tinazzia@sendo-org.it

Web: www.sendofoundation.org

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.