

A Modular Approach to Develop Complex Data Errors Checking Program in Clinical Trial Environment

Mirjana Stojanovic, Laura Archer

Cancer Center Biostatistics, Duke University Medical Center, Durham, NC

ABSTRACT

This paper describes a modular approach to developing a complex data error checking program. A module is a collection of functions that perform related tasks. We use a series of SAS® macros to develop each module. SAS macros are used intensively because they reduce code volume and improve program reliability and readability. SAS programs generated from SAS macros are dynamic and flexible. In this way the application of the program is much more flexible than in the traditional design as one monolith program. Our program works for many studies with no intervention into program code. In implementation of these checks, we have developed and use a specification file in which the user indicates the modules, and specific errors within the modules, to be performed. Then, where necessary, the user will provide additional information within the specification file regarding the tables and variables to be used in performing the checks. The driver program then pulls together all necessary modules and runs the checks. Reports for each section are produced in RTF, Excel and PDF formats. The size of the report is dependent on the numbers of sections (modules) to be used as well as on the numbers of specific questions defined in the specification file. Using a modular design we are able to reduce the time it takes to run study specific checks and simplify later maintenance of the program. This paper is intended for programmers with a sound foundation in SAS macro programming.

INTRODUCTION

Our goal of developing application software using SAS software was to check study data for completeness, consistency and availability in Cancer and Leukemia Group B (CALGB) studies in a unified and standardized way. External ORACLE data bases are maintained by data management staff, and used to store CALGB data. They follow all guidelines for storing study forms in machine readable form as well as security measures to prevent any unauthorized access including modification. A team of experts discussed the scope of checks as well as which ORACLE tables and variables should be checked. The product of their discussion was documented as "Generic Data Checks" which details the request for SAS program/macro that will perform data checks for all CALGB studies. The request was that said program should be maximally flexible so users (biostatisticians) would be able to easily choose which checks need to be performed.

Cleaning data for clinical research studies often consumes the significant portion of time (and money). If data are entered manually or by using optical scanning devices it is not reasonable to expect that all data will be entered correctly. Many times human error or illegible data on source forms produce problems as well as source forms not being filled out correctly. Our goal was to have reports with summarization of errors allowing data coordinators to identify and remedy the problems quickly. In that way the length of the process to have data with good quality for data analysis was minimized.

All data checks were divided into seven sections based on criteria like:

- Baseline data checks
- Death checks
- Case status checks
- Treatment status checks
- Follow-up data checks
- Adverse Event data checks
- Delinquency based on master tables checks

Further each section was divided into number of checks (questions) for detail checking of data within table or between two or more tables (crosschecking).

Macros Section1 to Section7 are the main parts of the application. In order to change the available checks all the programming that needs to be done is to add, delete, or modify part of SAS code in each section or in utility macros (tools). So we are able to find inconsistencies and discrepancies between values of variables in different tables. As a powerful tool we chosen SAS macros developed in house. By using a specification file the user was able to do all checks (several tens of checks) or just one check without any modification of the program. SAS

macros allow “checks on demand” which include dynamic generation of SAS code depending on number of checks requested. To simplify development of application software we developed so call ‘utility macros’ which were used repetitively in all sections. Further we will explain and give short description of the job of each macro.

PROGRAM DESIGN

Structure of Data_Error_Check SAS program:

```
%include "H:\Data Checks\Study\50303\DATA_CHECK_SPECFILE.sas";
%include "H:\Data Checks\Common\SECTIONFORMATS.sas" ;
%include "H:\Data Checks\Common\MACROS_4_DATA_CHECK.sas" ;

%include "H:\Data Checks\Common\SECTION1.sas" ;
%include "H:\Data Checks\Common\SECTION2.sas" ;
%include "H:\Data Checks\Common\SECTION3.sas" ;
%include "H:\Data Checks\Common\SECTION4.sas" ;
%include "H:\Data Checks\Common\SECTION5.sas" ;
%include "H:\Data Checks\Common\SECTION6.sas" ;
%include "H:\Data Checks\Common\SECTION7.sas" ;
%include "H:\Data Checks\Common\DATA_ERROR_SHELL.SAS" ;

%DATA_ERROR_CHECK
```

End user – statistician would see and possibly modify **just the following few lines.**

```
options ls=130 ps=48 nocenter;
```

```
* location of the data files and reports ;
```

```
libname db "H:\data CHECKS\Study\XXXXXX\" ;
```

```
* location of your data_check_specfile.sas;
```

```
%include "H:\data CHECKS\Study\XXXXXX\data_check_specfile.sas" ;
```

```
%DATA_ERROR_CHECK;
```

PARTS

Specification file (for each study)

Data_error_check.sas program

Macros_4_data_check (tools)

Common macros as tools (described below):

1. %EXIST
2. %COUNTOBS
3. %MISS_FORM1-5
4. %CONV_DATE

1. Macro for checking the existence of SAS data set. %macro EXIST (dsn);

2. Macro for checking existence and number of obs in data set.

%macro COUNTOBS (datastor=, count=_count_); * Macro designed by Frank Dilorio;

3. Macro for checking existence of forms versus Master data set.

%macro MISS_FORM (dataset=, Clin_Rev=, form_code=, seq=) ;

proc sort data=db.&dataset.(keep=patid clin_review) out=outdata ;

by patid ;

where clin_review in (&Clin_Rev.) ;

run ;

data temp_miss ;

merge master(in=in1) outdata(in=in2) ;

by patid ;

if (in1 and not in2) and

(today() - regis_dt > 91) then do ;

```

length form_code $ 8 table_name $ 10 miss_DS 3 ;
miss_DS = &seq. ;    * Specific form is missing ;
table_name = "&dataset." ;
form_code = "&form_code." ;
output ;
end ;
run ;

%countobs(datastor=temp_miss, count=_count_);

%IF &_COUNT_ %THEN %DO;
    * Data set with missing forms ;
    data missing_forms ;
        set missing_forms temp_miss ;
        by patid ;
    run ;
%END ;
%mend MISS_FORM ;

```

Other MISS_FORMx macros are similar to previous one with slightly different goal.

What we want to perform?

1. Checks for the presence of baseline data.
2. Checks of death data.
3. Checks of Case Status
4. Checks regarding treatment status.
5. Checks of Follow-up data.
6. Checks of Adverse Events (AE) data.
7. Delinquency checks based on master file data.

Generic Data Checks

The Generic Data Checks document specifies the data check that is provided by the macros. In implementing these checks, we have developed a specification file in which the user indicates the categories of checks to be performed. Then, where necessary, the user will provide information regarding the tables and variables to be used in performing the checks.

Goal of checking forms and data

- ❖ Finding inconsistency between forms
- ❖ Finding missing values on specific forms
- ❖ Finding forms which should not exist
- ❖ Finding missing forms
- ❖ Finding incompleteness in forms

Short description of specification file

- ❖ Spec file in essence is sequence of many %let macro statements.
- ❖ With these macro statements we assign study specific data set names, variable names and variable values which we will use in data checking.
- ❖ This way was used to make macros for requested sections as flexible as possible.
- ❖ Based on macro values for each section and each question (0,1) SAS macro preprocessor will decide which section and which question will be used or commented out.

Filling out data check specfile

```

%let Report_Location = H:\Data Checks\study\30306;
* Please never end previous statement with '\';
%let STUDY_NUM=;    * Insert Study number ;
* Specify the name of the master file data set ;
* This file must have one record per patient;
%let master=master;
%let PET_Study=;    * PET study (1=Yes, 0=No) ;
%let leukemia_study=; * Leukemia study (1=Yes,

```

* OFF TREATMENT VARIABLES;

```
%let offtrt=;          * Name of off treatment form SAS data file or 0 if na;
%let offtrt_reason=;    * Variable indicating if reason off treatment is death;
%let offtrt_death=;     * Variable indicating if reason off treatment is death;
%let offtrt_death_value=; * Value for death ;
%let offtrt_date=;      * Date off-treatment ;
%let offtrt_form_code=;  * Form Number for off treatment form ;
%let offtrt_reason_value=; * Value indicating patient did not start treatment;
%let last_date_of_prot_trt=; * Last date of protocol treatment ;
Etc.
```

Macro SECTION 1

The purpose of macro section1 is to produce baseline Data Checks including checking for the existence of data forms by 3 months after registration. The master data set is compared to other forms (ONSTUDY, NEWELG [eligibility], and SAMPLE etc.) to find missing, or incomplete forms. The macro also compare other forms with master to find so called phantom cases (typo in patient_id, wrong study number etc).

If master data set is not present complete processing of macro section1 will be aborted since comparison is not possible.

There are 34 questions in Baseline section.

* These are the flags for each question.

1 = Yes, check that question, 0 = No. ;

```
%let S1Q1 = ; * Q1 On-study form missing ;
%let S1Q2 = ; * Q2 Supplemental on-study form 1 missing ;
%let S1Q3 = ; * Q3 Supplemental on-study form 2 missing ;
%let S1Q4 = ; * Q4 Supplemental on-study form 3 missing ;
%let S1Q5 = ; * Q5 Supplemental on-study form 4 missing ;
```

....

Continues to 34th question.

Macro SECTION 2

Macro section2 compares death form with master and vice versa. If patient exists in Death form and same patient is still alive in master, that is a problem and that case will be on report. Similar if patient in master is dead and there is no observation in death data set that situation will be reported too.

These are the flags for each question.

1 = Yes, check that question, 0 = No;

```
%let S2Q1 = ; * Q1 = Patient status_id=8, death form not present ;
%let S2Q2 = ; * Q2 = Patient status_id=8, date of death on death form disagrees with patient status date or is
               missing/incomplete ;
%let S2Q3 = ; * Q3 = Death form present, patient status not updated ;
%let S2Q3_1 = ; * Q4 = Death form date does not agree with patient status date;
```

Macro SECTION 3

The goal of macro section3 is to compare master with follow-up, long term follow-up, off treatment, treatment summary and to present missing forms or discrepancies in data values. Example of checks.

3.2 If the patient is off study is there:

- 3.2.1 A follow-up form with a date of progression/relapse, or
- 3.2.2 A long term follow-up form with a date of progression/relapse, or
- 3.2.3 A treatment summary form with a date of progression/relapse, or
- 3.2.4 Patient status_id=8 (dead) and patient status_dt=case_rec status_dt, or
- 3.2.5 For PET studies (study.comm_id=6), if an off-treatment form present?

Error message will be "Patient is off study but no date of progression or death found" if no one of these conditions is met.

Macro SECTION 4

Section 4 looks for errors in treatment information. One check looks to see if an off treatment form or treatment summary form is present that the date and reason off-treatment completed. The macro also checks that if a follow-up form indicates a patient is off treatment that the appropriate treatment summary or off treatment form is present. Finally the section checks that if there is a long term follow-up form present that a treatment summary or off treatment form is present.

Macro SECTION 5

The section 5 macro checks progression and response data. It checks that on the first form that a progression is listed as best response that there is a date of progression on this form. It also checks to confirm that best response stay the same or improve over time. Note, this will be tricky for leukemia studies where the best response is recorded for the particular treatment period. For leukemia studies, treatment type is recorded on the follow-up form, so best response should stay the same or improve within treatment type. The check also looks to see if a patient is still in remission that a date last known in remission is indicated.

Macro SECTION 6

Section 6 is checks adverse event (AE) data. The macro checks that if the patient is still on treatment that there are AE forms submitted within a specified time period, e.g., at 6 weeks is at least 1 AE form in the database? The time period is to be specified by the user. The macro also checks for a gap between the to date of one form and the from date on the next form more than 7 days (i.e., is there a form missing). Finally the macro checks that if a patient is off treatment that AE forms are submitted within a specified time period. If the patient is off study then forms are not required past the off-study date.

Macro SECTION 7

Delinquency based on master table.

%let DELINQUENCY =;* 1 = Yes perform these checks, 0 = No;

BIG PICTURE

How this macro works?

Modular programming is used.

Program DATA_ERROR_CHECK.SAS is a driver program which puts together all necessary modules and runs checks. Section MACROS are stand-alone macros and independent of each other. Questions inside one section are independent of each other.

DATA_ERROR_SHELL.sas

Macro data_error_shell selects which macros to be run.

```
%macro DATA_ERROR_SHELL ;
```

```
%IF "&BASELINE." = "1" %THEN %DO ;
```

```
    %SECTION1 ;
```

```
%END ;
```

```
%IF "&DEATH_SECT." = "1" %THEN %DO ;
```

```
    %SECTION2 ;
```

```
%END ;
```

```
Etc.
```

In that way program DATA_ERROR_CHECK.SAS will be dynamic – it will compile and execute only those sections and corresponding data and proc steps which user needs. Specification file **must be** updated by user for each study. He/she should make all decisions (which section to include or exclude) and which questions to include or exclude. He/she should update data set names and form codes which are specific for that study.

EXAMPLE OF REPORT

1	patid	Problem	Form	Table
2	0	Treatment summary form present; reason off treatment not completed	C-1320	E10114
3	0	Treatment summary form present; reason off treatment not completed	C-1320	E10114
4	0	Follow-up form indicates patient is off-treatment; treatment summary form not present	C-1320	E10114
5	0	Treatment summary form present; reason off treatment not completed	C-1320	E10114
6	0	Treatment summary form present; reason off treatment not completed	C-1320	E10114

CONCLUSION

By using modular design task of building complex program for editing data for all CALGB studies was achieved. Change in program by adding new checks (edits) was relatively easy task. Modular design saved time in coding, debugging, maintaining and improved code. Our estimate was that monolith program would be 3-4 times longer than this one and very difficult for maintenance.

REFERENCES

Gratt, Jeremy and Adams, John (2006) "Large Scale Standard Macros - A Methodical Approach to Development and Implementation" PharmaSUG 2006 paper AD014.

Widel, Mario and Zhou, Jay (2006) "Techniques for Creating Reviewer-Friendly SAS® Programs" PharmaSUG 2006 paper TT19.

Michaels, Phillip (2004) "Building A SAS® Application to Manage SAS Code" SESUG 2004, paper AD08.

Ratcliffe, Andrew "Methodical SAS programming"

D. L. Parnas "The Modular Structure of Complex Systems" Computer Science and Systems Branch
U.S. Naval Research Laboratory, Washington D. C., USA

Peng, Fuping, and Perdomo, Carlos (2203)"A Modular Approach to Develop Patient Profile Application"
PharmaSug 2003, paper Ad003.

Carpenter, Arthur L. (2004), *Carpenter's Complete Guide to the SAS Macro Language, Second Edition*, Cary, NC: SAS Institute Inc.

Carpenter, Arthur and Smith, Richard (2002), "Library and File management: Building a Dynamic Application," *Proceedings of the Twenty-seventh Annual SAS Users Group International Conference*, Paper 21-27

Cheng, Edmond (2008) "Better, Faster and Cheaper Sas Software Lifecycle" NESUG 2008 paper Po21

Litzsinger, Michael and Riddle, Michael "A Modular Approach to Portable Programming" SUGI 27 PO34.

ACKNOWLEDGMENTS

We would like to thank Mr. Jeff Johnson MSc, Senior statistician, Duke University Medical center, NC for his advices, support and efforts to help in creation of this macros. His knowledge about CALGB database helps us tremendously answering all of our questions, and resolving all the issues we had at that time.

CONTACT INFORMATION

Your comments are greatly appreciated and encouraged.

Contact the author at:

Mirjana Stojanovic
Duke University Medical Center Duke University Medical Center
phone (919) 668-9337
E-mail: mirjana.stojanovic@duke.edu

Laura Archer
Duke University Medical Center
E-mail: laura.archer@duke.edu

TRADEMARK INFORMATION

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.