

Consistent Variables + Consistent Checks = Cleaner Data

Greg Silva, Biogen Idec, Maidenhead, UK

ABSTRACT

Data cleaning is one of the most important functions of a Biometrics group. Without clean data, programs have to be written defensibly, analysis cannot be completed correctly and deadlines are missed. This paper will outline a case for standardizing variable names across studies. Once names are consistent, then a method for standardizing checks using a set of SAS macro will be presented.

INTRODUCTION

Often statistical programmers are given the “small” picture – write some programs for a study. It may even be split into smaller pieces like tables or listings. Clinical programmers are often placed in a specific drug program or even indication, and do not see all of the issues that run across studies and drug programs.

By developing standard variables via a “global” library of names, programmers throughout a Biometrics department can start taking advantage of reusable code. This paper will be discussing a crucial component of Biometrics – the checking and cleaning of clinical trials data.

STANDARD VARIABLES

Earlier versions of SAS, as well as FDA electronic submissions requirements left the Biometrics programmer with limited naming options: 8 characters or less, case insensitive. With some companies, the Data Management group is separate from the Statistical group, this leads to even more disconnect with names. Add to this the statisticians defining derived variables, and you have the recipe for alphabet soup.

For example, in medical history the variable for Nervous System on one study may be NERVOUS which could be a text field for explaining the condition. On an anxiety study, the same name may be used for a quality of life question that rates the level of nervousness from 0 to 10. It could also be a question with a YES/NO expected. One variable name: three meanings and three data types. What’s a programmer to do?

By developing a naming convention for studies, and having that convention cover all studies, a company can lay down a foundation for efficiency. If a programmer can rely on NERVOUS only ever meaning one thing, then he or she can easily move across studies as priorities change. Statisticians can reuse specifications from existing studies for new studies. Data managers can spend more time with the data, instead of defining the data.

Once standards are in place, reuse can occur. A report containing known variables can be reused. A calculated variable can run across studies with original programming and checking recycled. Data checks can be run across studies, and stored for future studies.

STANDARD CHECKS

One of the easiest way to check data is to do a simple compare, i.e., is age <= 65? By defining the “simple” checks, the programmer can concentrate on the bigger issues. The analysis of the data and the consistency across data sets can be explored more fully. He or she can also go home at a reasonable hour.

By defining a company wide data dictionary, a company can set the groundwork for standard data checks. Having one person or a small committee decide on what the meanings and attributes of all variables are will allow more people to build familiarity with variables across studies.

With a standard of one name for each unique variable across all studies developed, companies can start developing data checks for these variables. Using SAS, the programmer can start building a library of data checks. Some discussion on how to define a check with multiple variables will be needed, but the end result will be a set of standard checks that can be run across studies and drug programs.

Putting a library of the data checks in place will allow for studies to reuse these checks. A check on the supine systolic blood pressure for an Oncology study will most likely be the same as for an Antifungal study. The data cleaners can rest assured that a check is in place for many of the common variables on a study.

Going a step further and making sure that each data check is named after the variable being checked, will allow for the development of a system that will match the variable with the data checks associated with that variable. Making this a rule allows for a system to be built that will automatically assign data checks to the data being checked.

STANDARD MACROS

The following section present the flow of a macro driven system that allows for standard data checks to be set up across studies, based on the “one variable – one meaning” rule. It also relies on a hierarchy of macro catalogues.

By defining levels of macro catalogues – one company wide, one drug program specific, one study specific, the programmer can access all of the “standard” checks, and can build custom checks. A sasautos statement that loads macros from the company level to the study level will allow programmers to customize as needed.

An example would be the standard age check: is the patient between 18 and 65 years old. For a paediatric study, a study level macro could be written to check that patients are less than 18 or perhaps less than 16 years old. The standard would still be available for all, but study specific checks could override these.

The system consists of two SAS programs: compile_macros.sas and gen_code.sas. The compile_macros program builds a macro catalogue of the actual edit checks. The gen_code program takes the list of all variables for a study and merges the data checks that exist for these variables. The result is a standalone program that is “customized” to the study: it runs checks for the variables defined in the study.

The compile_macros program (appendix 1) contains the few macros that set up the checking. The OPENDS macro is a data step stub that opens the requested data set and sets attributes for the variables that will contains the results of the edit checks. The close data check macro (CLOSEDC) contain the run statement that ends the data checking step, and stores the resulting data in a common data set.

The AGE and DIASTOL macros are examples of the data checks. The variable is checked, and if the condition is met, then a standard set of variables are defined that explain the error and provide information on the error. This information is stored in the final data set.

The power of the system is in the gen_code program. The first steps are to pull the names of the checks out of the catalogues defined. These names are merged with the entire list of variables in a given study. If a variable is in a study, and the variable has a check, it will be stored in the edits data set.

The data null at the end of the macro builds a program that checks each data set in a study. The result of this data null is found in Appendix 3. The general structure is: open a data set, invoke all data checks that have been defined for the variables in the data set, close the data check data set and store it for reporting.

The final result is a data set with one observation for each issue found in a study. The layout of the data set, and the messages produced are consistent across all studies. A single program can be added to produce the report in a Word document or even to the Web for review across the company.

CONCLUSION

By defining variables in a standard way across all studies, a simple process of checking data in studies can be developed. Using SAS macro catalogues to order the invocation of macros, programmers will be free to add study specific checks, without overwriting standard data checks.

An added benefit to this system is the standardization of variable names across studies. This allows for less time to be spent training programmers and statisticians on the mechanics of a study – everyone that has worked on any study will know that AGE is the age of the patient at the start of the study.

The end result of this system is a set of “run and forget about ‘em” data checks. This library of checks will allow people to spend more time on the unique issues that arise with each study. From data manager to statisticians, more eyes can be looking at the bigger picture of a study.

CONTACT INFORMATION

Comments and questions are appreciated. I anticipate my phone number will be changing within the year, so please contact me via email.

Greg Silva
Biogen Idec
Thames House
Foundation Park
Maidenhead, SL6 3UD, UK
Greg.silva@biogenidec.com

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.

Appendix 1: Macro compile_macros

```
Options MAUTOSOURCE MSTORED SASMSTORE=datachk ;
libname data 'your_data_library' ;

%*** Open dataset. ;
%macro opens(dsname) / store ;

data &dsname (keep = patid _dsname _errmsg _errtxt _varname) ;
  set data.&dsname ;
  length _errmsg $50
         _errtxt $100
         _varname
         _dsname $ 8 ;
%mend ;

%*** Save the checking results. ;
%macro storedc(dsname,study) / store ;

run;

proc sort data = &dsname ;
  by dsname patid ;
run ;

data alldsn ;
  set alldsn
      &dsname ;
run ;
%mend ;

%macro age(hi=80,lo=18) / store ;
  if ( age < &lo or age > &hi ) then do ;
    _errmsg = "AGE: value abnormal (<&lo, >&hi)" ;
    _errtxt = put(age, 8.) ;
    _varname = "AGE" ;
    _dsname = "DEMOG" ;
    output ;
  end ;
%mend ;

%macro diastol(hi=120,lo=40) / store ;
  if (diastol > &hi or (diastol < &lo and diastol > .)) then do ;
    _errmsg = "DIASTOLIC BP: value abnormal (<&lo,>&hi)" ;
    _errtxt = put(diastol, 4.) || ", VISIT=" ||visit ;
    _varname = "DIASTOL" ;
    _dsname = "VITALS";
    output ;
  end ;

  if (systolic < diastol) then do ;
```

```

        _errmsg = "DIASTOLIC BP: greater than Systolic BP" ;
        _errtxt = put(systolic, 4.) || "/" || put(diastol, 4.) || ", VISIT=" || visit ;
        _varname = "SYSTOLIC" ;
        _dsname = "VITALS";
        output ;
    end ;
%mend ;

```

Appendix 2: Macro Gen_code

```

%macro gen_code(stdyroot=, protocol=) ;

%*** Get all of the pre-defined macros in a data set. ;
proc catalog cat=datachk.sasmacr ;
    contents out=checks (keep = name) ;
run;

%** Get all of the variables from all of the data sets in a study. ;
proc datasets library = data;
    contents data = _all_ noprint
        out = work.vars (keep = memname name) ;
run ;

proc sort data = checks ;
    by name ;
run ;

proc sort data = vars ;
    by name ;
run ;

%*** Merge all the variables together. ;
data edits ;
    merge checks    (in = checks)
          vars      (in = vars) ;
    by name ;
    if (checks and vars) ;
run ;

proc sort data = edits ;
    by memname ;
run ;

%** Generate the data check program. ;
data _null_ ;
    length text $100 ;
    set edits end = eof ;
    file "data_checks_&protocol..sas";

    by memname ;

    if (_N_ = 1) then do ;
        put "options SASMStore=datachk MautoSource MStored ;" / ;
        put 'data alldsn;' ;
        put ' stop ; ' ;
        put 'run; ' //;
    end ;
    if (first.memname) then do ;
        put "%***** Data Checks for: " memname " *****;" ;
        text = '%opens(' || compress(memname) || ')' ;
        put text ;
    end ;
    text = '%' || compress(name) ;
    put @3 text ;

```

```

if (last.memname) then do ;
  text = '%storedc(' || compress(memname) || ",&protocol)" ;
  put text //;
end ;
if (eof) then do ;
  put '                                     ' ;
  put 'libname edit ".";                 ' ;
  put 'data edit.edtchk_&sysdate9. ;     ' ;
  put '  set alldsn ;                     ' ;
  put 'run;                               ' ;
end ;
run ;
%mend ;

```

Appendix 3: Macro generated by gencode

```
options SASMStore=datachk MautoSource MStored ;
```

```
data alldsn;
  stop ;
run;
```

```
%***** Data Checks for: AE *****;
```

```
%opens (AE)
  %ACTION1
  %AEINFECT
  %ONSEDATE
  %PT_CODE
  %RELAT1
  %SERIOUS
  %SEVERITY
  %SYMPTOM
  %TREATMEN
%storedc (AE,ABC-123)
```

```
.
.
.
```

```
%***** Data Checks for: VITALS *****;
```

```
%opens (VITALS)
  %DIASTOL
  %EXAMDATE
  %PULSE
  %SYSTOLIC
  %TEMPERAT
  %TUNITS
%storedc (VITALS,ABC-123)
```

```
libname edit ".";
```

```
data edit.edtchk_&sysdate9. ;
  set alldsn ;
run;
```