ABSTRACT

An important responsibility for a biostatistician supporting clinical trials is to ensure the quality of data. The Central Unit of the Collaborative Antiviral Study Group, located at the University of Alabama at Birmingham, oversees many antiviral pediatric and adult multi-center clinical trials, most of which are long-term trials. The major part of our data quality monitoring is to check if every case record form (CRF) is collected, and endpoint values are entered correctly according to the protocol. Macro is an efficient way to handle this kind of tasks. However, since the number of variables for unique ID is not the same for different CRFs, and the endpoints are scattered in different CRFs, many macros or ordinary codes are needed to accommodate these situations. In this paper, we describe a simple approach to increase the efficiency of macros for this circumstance by using dummy variable and placeholder property of macro parameter.

Keyword: clinical trial, data quality monitoring, macro, dummy variable, macro parameter, placeholder.

BACKGROUND

The Collaborative Antiviral Study Group (CASG) was established by the National Institute of Allergy and Infectious Diseases, National Institutes of Health in 1972. Since its inception the University of Alabama-Birmingham has served as the Central Unit for the conduct of multi-center, controlled clinical trials of antiviral therapies for non-HIV diseases. Many of the trials last 5 or more years. The associated therapeutic studies are designed to result in licensure of compounds for improved health of people suffering from diseases such as herpes simplex encephalitis, neonatal HSV infection, and VZV infections, as well as, hepatitis B and C, human papillomavirus, influenza, hantavirus pulmonary syndrome, and enteroviral diseases of the newborn.

The Biostatistics Unit of the Comprehensive Cancer Center of the University of Alabama at Birmingham also serves as the Biostatistics Unit supporting CASG. We are involved in clinical trial protocol design and CRF design; our services include randomization scheme generation, data entry, DSMB report, data quality report, IND report, and statistical analyses.

Since the whole working code is too long, only some relevant pieces are quoted in this paper for demonstration purpose. The code structure, dataset and variable names remain unchanged.

PROBLEM

In order to generate the periodical data quality reports for some clinical trials, we go through two steps: first, data double entry and comparison; and second, checking the database for CRFs and study endpoints against the protocols. Basically, the checking is to make sure that the CRFs and study endpoints have been collected and entered into the database within certain time windows according to each protocol. In the process of programming, we met the following problems:

1. A set of CRFs for one patient may have more than two hundred pages. Although we group the same kind of forms into one dataset, there still more than a dozen of datasets in the database for one clinical trial. The number of variables for a dataset key, a group of variables that uniquely identify every observation in one dataset, is various among the datasets. To run proc compare after double data entry, we may write similar macros like sample code 1 to handle the situation in which dataset key has two variables, and use sample code 2 to handle the situation in which dataset key has three variables, and so on.

Sample code 1:
/* compare contents for two-variable IDs */

```verbatim
%macro content2(form,id1,id2);
  data temp;
  set in.&form;
  proc sort;
    by &id1 &id2;
  run;
  data vtemp;
  set in.v&form;
  proc sort;
    by &id1 &id2;
  run;
  proc compare data=temp compare=vtemp;
    id &id1 &id2;
    title1 "CSAG 502 (HCV)";
    title2 "Check Variable Values for &form";
  run;
%mend content2;
```

Sample code 2:
/* compare contents for three-variable IDs */
To calculate the percentage of availability of study endpoints, we need to merge some datasets for certain endpoint. But for different endpoints, the number of datasets need to be merged are different. We may write several similar macros to accommodate all the possibilities, like sample code 3, to handle the situation when two datasets are involved, and use sample code 4 to handle the situation when three datasets are involved, and so on.

Sample code 3:
/* the same endpoint in 2 datasets */

%macro endpoint2(f1, f2, missing, outset, endpoint);
  * f1-f2: datasets containing the endpoints;
  ... data temp;
    set &f1 &f2;
  ...%mend endpoint2;
  %endpoint2(f44, f157, MissPCR, SumPCR, 'Secondary');

Sample code 4:
/* the same endpoint in 3 datasets */

%macro endpoint3(f1, f2, f3, missing, outset, endpoint);
  * f1-f3: datasets containing the endpoints;
  ... data temp;
    set &f1 &f2 &f3;
  ...%mend endpoint3;
  %endpoint3(f189, f190, f224, MissSur, SumSur, 'Primary');

3. The study endpoints have different variable names in CRFs; and for the same study endpoint, it locates in different CRFs for different checking time points. How can we uniformly indicate whether they are missing or not? Especially for the case, the CRF that contains the expected study endpoint is not in the dataset.

**SOLUTION**

The approach to make the improvement for problem 1 and 2 is the same – using placeholder macro parameter, which may be assigned "_Null_", or nothing. They are only assigned values when applicable.

For problem 1, we can use one macro to handle the situation that different datasets have different number of variables for the unique ID in PROC COMPARE, see sample code 5. In this case, id1 – id6 are placeholder parameters.

Sample code 5:
/* compare contents */

%macro content(form,id1,id2,id3,id4,id5,id6);
  data temp;
    set in.&form;
  proc sort;
    by &id1 &id2 &id3 &id4 &id5 &id6;
  run;

  data vtemp;
    set in.v&form;
  proc sort;
    by &id1 &id2 &id3 &id4 &id5 &id6;
  run;

  proc compare data=temp compare=vtemp;
    id &id1 &id2 &id3 &id4 &id5 &id6;
    title1 "CSAG 502 (HCV)"
    title2 " Check Variable Values for &form"
  run;
  %mend content;
  %content(Form10t,pin,pageno);
  *------ (1);
  %content(Form22t,pin,pageno,examdy);
  *------ (2);
  %content(Form43c,pin,startmo,startdy,pred,solum,fk506);
  *------ (3);

Here six parameters used in the macro definition for %content(…). That means the maximum possible number of variables for id statement in PROC COMPARE is six for all the situations. But it is not necessary to assign values to all these six parameters, like (3), every time the macro is used. If only two variables are needed, you just plug the two variables in, like (1). It also works for three variables, like (2).
For problem 2, similarly, we use the following macro to cover all situations, see sample code 6. In this case, f1 - f6 play the role of placeholder.

Sample code 6:
/* check the endpoint */

%macro endpoint(f1,f2,f3,f4,f5,f6,missing,outset,endpoint);
  * f1-f6: datasets containing the endpoints;
...  
data temp;
  set &f1 &f2 &f3 &f4 &f5 &f6;
...%
%mend endpoint;

%endpoint(f44,f157,_Null_,_Null_,_Null_,_Null_,MissPCR,SumPCR,'Secondary');
*------ (a);
%endpoint(f189,f190,f224, , , ,MissSur,SumSur,'Primary');
*------ (b);
%endpoint(f49,f171,f205,f239,f273,f306, MissSF36,SumSF36,iteSF36,'Secondary');
*------ (c);
%endpoint(f1=f44,f2=f157,missing=MissPCR,outset=SumPCR,endpoint= 'Secondary');
*------ (d);

Basically, the explanation is similar to for sample code 5. To further demonstrate the features of placeholder, those parameters are put at the beginning of the parameter list instead of the end. In this arrangement, you must assign '_Null_' to hold the place of the parameter that are not applicable, like (a); or leave them blank using delimiter ',', to indicate those places are held, like (b). (c) is the situation that very parameter has a value. If you use the form like (d), you do not need to worry about matching the parameters or the sequence of the parameters within the parentheses. (d) runs the same as (a).

Problem 3 is another kind of problem that is often encountered in data quality monitoring programming. There are many ways to handle this. What I did in our SAS code is using one dummy variable to indicate the status of different study endpoint variables. You may use more than one if needed. Since the schedule of study endpoints is the same as the CRFs that contain the endpoints, the dummy variable 'Chk_Point' is initiated in the macro %setpage(...) in sample code 7.

Sample code 7:
/* set each page of CRFs */

%macro setpage(f, form, page_no, expday);
  ... if a variable should be checked, assign it to Chk_Point, which will be used later *;
...  
    then Chk_Point=totraw;
else if &page_no in (51,173,207,241,274,308,342,376)
    then Chk_Point=totraw;
else if &page_no in (44,157)
    then Chk_Point=pcr;
...%
%mend setpage;

Then, when checking the study endpoints, we can use the dummy variable 'Chk_Point' as a part of information to help make the decision whether certain study endpoint is missing or not, like sample code 8.

Sample code 8:
/* check the endpoint */

%macro endpoint(f1, f2, f3, f4, f5, f6, missing, outset, endpoint);
...  
  if Chk_Point=., &Blank<=0 &Pageno in
    then Missing=1;
...%
%mend endpoint;

In this way, we do not need treat the study endpoints variable by variable, page by page. We can cut off many redundant codes.

SUMMARY

There are some situations in SAS programming, in which we have to write long code as we did for the clinical trial data quality monitoring programs. If we find that some macros use the same pattern, we can group them together, and use one macro to accommodate all the variations by the placeholder macro parameter approach as illustrated in the paper. If we need certain characteristic indicator of a variable, dummy variable can be very useful. By trying some little things like these, we can make the macros more functional and the code more concise.

REFERENCE


CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contacted author at: