Paper 081-2010

Automated Data Cleaning Linking Data Manager to Study Coordinator: Error Finding by Variable and Error Fixing by Subject

Christine T. Román, The Gertrude H. Sergievsky Center, Columbia University Medical Center, New York NY
Emma K. T. Benn, The Gertrude H. Sergievsky Center, Columbia University Medical Center, New York NY
Dale C. Hesdorffer, The Gertrude H. Sergievsky Center, Columbia University Medical Center, New York NY
Anthony Marmarou, Department of Neurosurgery, Virginia Commonwealth University, Richmond, VA
Shlomo Shinnar, Comprehensive Epilepsy Management Center, Albert Einstein College of Medicine, New York, NY

ABSTRACT

Many researchers using the SAS® System software receive data sets with incomplete data or data errors. Data cleaning can become an extremely tedious task for the Coordinators who must return to medical records or case report forms (CRFs) if the data managers list errors by variable. Instead it is easier for Coordinators to clean data when errors are listed by subject across all CRFs. Our series of macros encompass a step-by-step process that allow for error searching by variable, while outputting errors by subject.

INTRODUCTION

Before researchers can analyze their data it is necessary to clean the data. This process involves intensive communication between data managers and Study Coordinators and is a crucial step before undertaking data analysis. Coordinators receive reports of data errors from data managers and must return to the CRFs or medical records to check and potentially correct errors identified by logic checks within or across forms and by checks for missing data. Typically, such errors are reported by variable; however, this is very cumbersome for Coordinators who must then check one variable at a time across subjects. We have constructed a series of macros that allow data managers to find errors by variable in a given data set and output the errors by subject for the Study Coordinators.

The study that gave rise to this need, the Consequences of Prolonged Febrile Seizures in Childhood (FEBSTAT), is a prospective multicenter study designed to address the relationship between febrile status epilepticus and the subsequent development of mesial temporal sclerosis and mesial temporal lobe epilepsy. Ultimately a total of 200 children ages 1 month to 5 years presenting with a febrile seizure lasting 30 minutes or more will be recruited and followed over time. The data collection effort is large, with 9 forms at baseline, 5 forms at one month, and 5 forms at one year, together adding up to approximately 200 pages of forms. The number of forms and variables makes data cleaning by variable and not by subject, tedious and overly time consuming for Coordinators, and itself prone to error. This gave rise to the creation of the data cleaning approach by subject described here.

Our procedure enhances the %split1 Macro, introduced by Selvaratnam Sridharma at the SUGI 28 Proceedings, and a macro returning the number of observations in a data set, introduced by Jack Hamilton at the SUGI 26 Proceedings. Combined, these macros can divide larger data sets into smaller data sets, count the number of errors in each of the smaller data sets, and eliminate data sets without errors. Then, through the addition of a final merge, data managers can focus on subjects with errors using a data structure that is convenient for the Study Coordinators.

MOTIVATING EXAMPLE

Imagine a data set, ***o.demo*** (Figure 1), containing the following information for four children: idnum, sex, income (parental income), age (in years), weight (in pounds), and height (in inches). Missing information is apparent when the value of the response is -999.

idnum	sex	income	age	weight	height
1p1	0	1	5	-999	-999
1p2	1	-999	4	45	40
1p3	0	-999	-999	37	38
1p4	0	2	4	40	42

Figure 1: Dat set *o.demo*, **prior to applying macros**

The %miss Macro

This macro is designed to identify missing information by variable and then create a new data set, **&data,** for each variable, **&a**.

```
%macro miss(data,a);
data &data; set o.demo(keep = idnum &a);
if &a = -999;
proc sort data = &data;
by idnum;
run;
%mend miss;

%miss(set1,sex);
%miss(set2,income);
%miss(set3,age);
%miss(set4,weight);
%miss(set5,height);
```

The **%miss** macro created five different data sets with two variables, **idnum** and **&a,** from ***o.demo***. Since the variable **sex** has no missing data, we expect the **sex** data set to have no observations. It is easy to see this here because ***o.demo*** has four subjects and five meaningful variables. However, a typical data set would be composed of more variables and more subjects, making it difficult to point out missing data at first glance. Since our interest is to merge these data sets, having a data set with no observations could be problematic. Instead of manually looking for those data sets with observations, we designed a macro, the **%obs macro,** to delete data sets with no observations and keep those data sets with observations. The retained data sets are referred to as **V&i**, where **i** represents the number assigned to each data set created by the **%miss macro.** It is important to note that the data sets containing observations are being stored in a common library other than the library where ***o.demo*** is located; we chose the “work" library to be our common library.

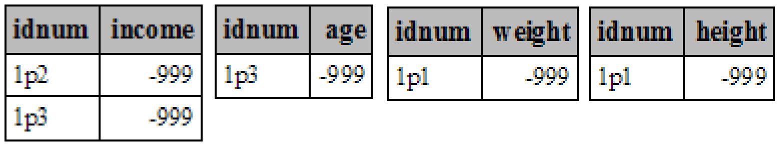
The %obs Macro

```
%macro obs(data,i);
%let dsid=%sysfunc(open(&data));
%put dsid=&dsid;
%let hasobs=%sysfunc(attrn(&dsid, any));
%put hasobs=&hasobs;
%if &hasobs=0 %then %do;
%let rc=%sysfunc(close(&dsid));
%put rc=&rc;
%end;
%if &hasobs=0 %then %do;
proc datasets library=work memtype=data;
delete &data;
run;
quit;
proc datasets library=work details memtype=data;
run;
quit;
%end;
%else %do;
%let rc=%sysfunc(close(&dsid));
%put rc=&rc;
%if &hasobs ne 0 %then %do;
proc datasets library=work memtype=data details;
change &data=V&i;
run;
quit;
%end;
%end;
%mend obs;

%obs(set1,1);
```

```
        %obs(set2,2);
        %obs(set3,3);
        %obs(set4,4);
        %obs(set5,5);
```

The above macro, **%obs**, consists of a series of macro functions which we have denoted as, **dsid**, **hasobs**, and **rc**. The macro variable **dsid** opens all the data sets created by the **%miss** macro and assigns them a data set id; **hasobs** verifies whether the opened data sets contain observations; and **rc** closes the data sets specified by **hasobs**. If the data set specified by **dsid** has no observations, **hasobs** takes a value of 0. If the data set has variables and no observations, or neither variables nor observations, it takes a value of -1. If the data set has both variables and observations, it takes a value of 1. However, for our scenario, **hasobs** will only take on a value of 0 or 1.

This step is then followed by a series of **%do** loops. In the first loop, **rc**, closes the data sets with no observations and, via a the **DATASETS** procedure, these closed data sets are deleted. At the end of this loop, only those data sets with observations remain in the work library. In the second loop, **rc** closes the data sets that contain observations, and via a **PROC DATASETS** step, these data sets are renamed and saved to the work library. The resulting data sets (**V2-V5)** are now ready to be merged (see Figure 2).



idnum	income
1p2	-999
1p3	-999

idnum	age
1p3	-999

idnum	weight
1p1	-999

idnum	height
1p1	-999

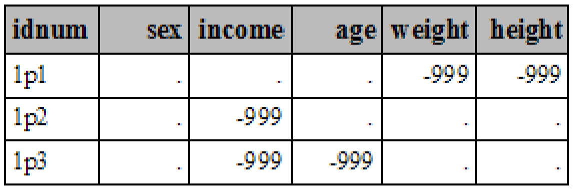
Figure 2: Data sets (V2-V5) created by the %obs **macro**

Our next macro, the **%merge macro**, was designed to merge the data sets, **V2-V5**, by idnum**.** A **DATA** step would be a practical method to carry out this task. However, keeping in mind that we typically work with numerous data sets, we used the **SQL** procedure to merge the data sets (a response to a posting on http://www.listserv.uga.edu/archives/sas-l.html).

The %merge Macro

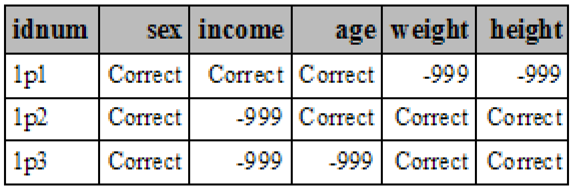
```
        %macro merge(b);
        proc sql noprint;
        select MemName Into:mylist Separated By ' '
        from Dictionary.Tables
        Where Libname = 'WORK'
        And Memtype = 'DATA';
        quit;

        proc format;
        value corr
               . = "Correct";
        run;

        data &b;
        merge &mylist;
        by idnum;
        format _numeric_ corr.;
        run;
        quit;

        proc datasets library=WORK memtype=data;
        save &b;
        run;
        quit;
        %mend merge;

        %merge (missing);
```

There are several important things to note about the **%merge** macro. First, all the data sets created by **%obs** are saved in the work library which must not contain data sets other than those needed to be merged. Thus, the **%merge** macro consists of a **PROC SQL** step that allows SAS to create a table with a list of all the data sets that exist in the work library. Once the list is created, **&mylist**, SAS merges all the data sets on the list by **idnum**. Second, the resulting data set created by **%merge**, which we denoted as ***missing***, is a single data set that contains all of the subjects for whom missing data exist. We expect ***missing*** to contain three subjects (1p1, 1p2, and 1p3) and five variables (idnum, income, weight, age, and height). However, not all subjects will have missing data for the same variable. The data set ***missing*** will display a period (.) for the subjects that had no missing data for a particular variable (see Figure 3). By applying the **corr format**, the word "Correct" will be displayed in place of the period. This will indicate to the Coordinators that there were no errors found for this variable for this subject (see Figure 4).



idnum	sex	income	age	weight	height
1p1	.	.	.	-999	-999
1p2	.	-999	.	.	.
1p3	.	-999	-999	.	.

Figure 3: Data set *missing* **before applying the** *corr* **format**



idnum	sex	income	age	weight	height
1p1	Correct	Correct	Correct	-999	-999
1p2	Correct	-999	Correct	Correct	Correct
1p3	Correct	-999	-999	Correct	Correct

Figure 4: Data set *missing* **after applying the** *corr* **format**

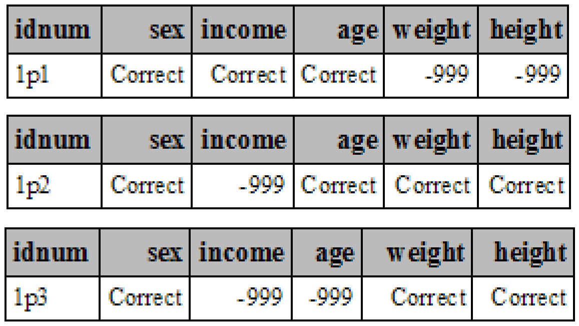
In order to split this data set into three smaller sets, we designed the **%split1A** macro, which is an adaptation of the %split1 macro.

The %split1A Macro

```
%macro split1A();
data _null_;
if 0 then set missing nobs=count;
call symput('numobs',put(count,8.));
run;

data %do J=1 %to &numobs; missing_&J %end; ;
set missing;
%do I=1 %to &numobs;
if %eval(&i-1)<_n_<= %eval(&I)
then output missing_&I;
%end;

%do I=1 %to &numobs;
data missing_&I; set missing_&I;
call symput("b",idnum);
proc print data = missing_&I;
title1 Missing Data For Idnum = &b;
run;
quit;

data o.missing_&b; set missing_&I;
run;

%end;
%mend split1A;

%split1A();
```

The **%split1A** macro begins by counting the number of observations in the large data set, ***missing***. The large data set with **&numobs** observations is split into **&numobs** smaller data sets each with a single observation.

Now that we have created a data set by subject, our split macro proceeds by outputting each data set via the **PRINT** procedure. This macro is then finalized with a **DATA** step where each data set is saved in a common library. The final output and each data set are named using the subject's idnum, **missing_1p1**, **missing_1p2**, and **missing_1p3** (see Figure 5).



idnum	sex	income	age	weight	height
1p1	Correct	Correct	Correct	-999	-999

idnum	sex	income	age	weight	height
1p2	Correct	-999	Correct	Correct	Correct

idnum	sex	income	age	weight	height
1p3	Correct	-999	-999	Correct	Correct

Figure 5: Data sets created by the %split1A **macro**

LIMITATIONS

This series of macros is not designed for data sets with repeated observations, however, it is possible to modify such data and apply these macros. If the data set contains multiple references to the same subject, a variable that is unique for each observation could be created. By creating unique variables, our series of macros will then apply.

CONCLUSION

Our series of macros, **%miss, %obs, %merge**, and **%split1A** encompasses a step-by-step process that allows the data manager to search for errors by variable while outputting the errors by subject. Thus, our macros maximize the efficiency of the data cleaning process for both data managers and Study Coordinators.

REFERENCES

Hamilton, Jack. 2001. “How Many Observations Are In My Data Set?” *SUGI Proceedings;* paper 95-26.

Shinnar, S., Hesdorffer, D. C., Nordli, D. R., et.al. 2008. “Phenomenology of prolonged febrile seizures: Results of the FEBSTAT study. *Neurology* 2008;71;170-176.

Sridharma, Selvaratnam. 2003. “Splitting a Large SAS® Data Set.”*SUGI Proceedings 28*; paper 75-28.

ACKNOWLEDGMENTS

This work was supported by Grant NS 43209 from NINDS. The authors would like to thank Malka Schlesinger, MS, for her hard work in the initial phases of data cleaning for the FEBSTAT study.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Christine Román
The Gertrude H. Sergievsky Center, Columbia University Medical Center
630 W 168th St #19-319
New York, NY 10032
Work Phone: (212) 305-7848
Fax: (212) 305-2426
E-mail: croman@sergievsky.cpmc.columbia.edu
Web: http://www.cumc.columbia.edu/dept/sergievsky/