Paper 082-2008

Tools of Miss-Calculation: Managing Missing Values with SAS®

William C. Murphy
Howard M. Proskin and Associates, Inc., Rochester, NY

INTRODUCTION

Programmers working for a consulting company specializing in clinical trials can be confronted with databases that contain huge amounts of data. Several different lab results, physical exams, adverse event recordings, medication tracking, and various other examinations can lead to hundreds, if not thousands, of parameters in a typical study. It would be great if all of the measurements were recorded for each subject, but this never, ever happens. For whatever reason, there are always hundreds of missing values for the study scores. In fact, in a large study, you will not even find one subject without some missing data. However annoying these missing values are, we cannot ignore them. Luckily the SAS system provides us with tools to manage these voids in the data.

PRELIMINARIES

In the SAS system, numeric missing data are actually characterized into 28 different levels: the well-known '.', the '._', and the letters of the alphabet preceded by a period. Such missing information can be read by a DATA step employing the MISSING statement. Furthermore, this information can be tabulated by using the MISSING option in the PROC FREQ TABLE statement. Classes of missing can be incorporated into models and analyzed using PROC GLM or PROG MIXED, as long as the MISSING option is used on the CLASS statements.

These missing values, however, can still cause havoc in the analyses of data if not properly handled. If not properly coded, the missing values can lead to incorrect sums and means. Even if properly coded, undetected missing values can lead to erroneous conclusions about the number of observations to be analyzed. Furthermore, if errors in data collection mandate the removal of the data, coding could be tedious. The SAS system however provides a variety of tools and techniques for the detection and creation of missing values. These tools can help a programmer write clear and concise code for the management of missing values.

MISSING STATEMENT

You are requested to generate a report on all subjects in a study that had a missing value for their birth dates. So you write a DATA step to subset the data:

```
data MissingBirth;
       set  Raw;
       if birthdate = . then output;
       run;
```

This step is simple and seemingly effective code. However, the person who created Raw used special missing character to account for the different missing (*e.g.* .A for adopted and no original record, .Q for interviewer forgot to ask, *etc.*). So you still have missing birthdates in your output data. Rewriting the IF statement as

```
if birthdate <= .Z then output;
```

would solve this problem. Another research site sends you a new set of Raw data and you run your subsetting program. Of course, it doesn't work. This time the researchers entered the dates as text strings. So you could now write the IF statement as

```
if birthdate = ' ' then output;
```

But you have no way of knowing what kind of data you will get in the future. To cover all bets, you could write the IF statement as

```
if missing(birthdate) then output;
```

where we have employed the MISSING function. This function does not care weather its argument is character of numeric, nor does it care what type of numeric missing the argument is. The function produces a 1 for any missing argument and a 0 otherwise.

MISSING OPTION

You have just printed out a listing of the number of Adverse Events for each patient in a drug study. Since no data exist for patients without occurrences, the listing shows a '.' for these patients, even though it should say '0'. Now you could write a DATA step to change the missing values to zero, or you could format the value to zero. Since you don't need this information anywhere else, why not just change the global option:

```
option missing = '0';
```

The MISSING option allows you to change the character displayed when missing numeric data is encountered. So instead of '.' in our output, we now have '0' displayed. In fact, all displays of the data after the invocation of this option will be set to the new value. So remember to change the option back after your output statements or you might get unexpected results later in your program.

FINDING NO DATA

You're prepping a data set for analysis, but the statistician only wants a 'balanced' design. The subject must have a value for the analysis measurement for each and every treatment. So you must find out if any data are missing for any treatment. If there were five treatments, you could write

```
data AnalysisData;
      set RawData;
      array score {5} score1-score5;
      do Treament=1 to 5;
            if missing(score[Treament]) then delete;
            end;
      run;
```

where we look for any missing score among the 5 visits. Or we could write

```
data AnaylsisData;
      set Rawdata;
      if nmiss(of score1-score5)>0 then delete;
      run;
```

The function NMISS counts the number of missing values among its arguments. If it is greater than zero we get rid of the value to achieve a data set with all non-missing scores.

MAKING DATA VANISH

You have just received some new data from a lab and notice that some of the values are a little strange. So you contact the lab about these values and you find that they had instrument problems when running patients 37, 38, and 39 and their readings are not reliable. You are instructed to remove the lab values from the database for these individuals but leave the other values. So you proceed to write

```
data CorrectedLab;
      set OriginalLab;
      array NumericLab{*} wbc rbc monocytes …;
      array CharacterLab{*}  opiates cocaine urineketone …;
      if patient in (37, 38, 39) then do;
            do i=1 to dim(NumericeLab);
                  NumericLab[i] = .;
                  end;
            do i=1 to dim(CharacterLab);
                  CharacterLab[i] = ' ';
                  end;
```

```
                end
        run;
```

where you had to treat numeric lab value differently that character lab values. So you wrote a lot of code just to make some variable values disappear.

But the SAS system has a routine that can simplify this:

```
data CorrectedLab;
       set OriginalLab;
       if patient in (37, 38, 39) then
              call missing (wbc, rbc, monocytes,… opiates, cocaine, ketone,…);
       run;
```

where we have done away with the arrays and loops. CALL MISSING will set any variables in its argument to missing whether character or numeric. Just as with the NMISS function discussed above, you can use a range of variables in the CALL MISSING argument to further simplify the expression.

FINDING THE LAST MEASUREMENT

You are told to make a list of the last visit that a subject had in a study. So you just sort your data by the Visit variables and take the last one. You discover that all of the subjects finished the study and came for the last scheduled visit! So you check the data to make sure this is true. But you find that whoever created the data decided to have entries for all subject whether or not they showed up for a visit. Now your task is to find the last visit in the database after you remove the placeholder entries. You must remove observations in which all study measurements are missing. You write the code

```
data NoMissing;
       set DataWithMissing;
       array xxx{*}  Weight BP Pulse Test1-Test10 OtherVars;
       MissingCount=0;
       do i=1 to dim(xxx);
              MissingCount + missing(xxx[i]);
              End;
       if MissingCount=dim(xxx) then delete;
       run;
```

This code would appear to work, but wouldn't it be simpler if we wrote

```
data  NoMissing;
       set DataWithMissing;
       if n(Weight,BP,Pulse, of Test1-Test10,OtherVars)=0 then delete;
       run;
```

where we use the N function to count the number of non-missing values? If all values are missing then the N function will be zero. This is a lot simpler than our first example, but like the first lines of code, it does not take into account character values. However, we can check for character missing by writing the IF statement as

```
if missing(cats(Weight,BP,Pulse, of Test1-Test10,OtherVars)) then delete;
```

where we have used the CATS function to concatenate all the values of all the variables into one string. If there is nothing in the string formed by the combination, then there was nothing there in the individual variables. Note that the CATS function works on both numeric and string variables, but for missing numeric it will concatenate the missing character to the string. Therefore the string will never be missing if there are numbers in the argument. With this in mind we should write our logical statement as

```
if not n(...list of numeric variables...) and
       missing(cats(...list of character variables...)) then delete;
```

where the first part of the logical statement is true only when all numeric values are missing and the second part of the statement is true only when all character values are missing. Therefore, this code will remove all observations with missing measurements from the data. With the missing gone, finding the last visit is now trivial.

CONCLUSIONS

The array of tools that the SAS system supplies for handling missing values is large and varied. Using the global option MISSING we can control the appearance of missing values in the output; using the MISSING and NMISS functions we can detect and count the number of missing data points; and using the CALL MISSING routine we can set parameters to missing. These tools and others can make negotiating the voids in your data easy and effective. And the reports you make to a client complete and error free.

REFERENCES

SAS Institute Inc. 2002-2006, *SAS OnLineDoc® 9*.1.3 "Functions and CALL Routines: MISSING Function", "Functions and CALL Routines: NMISS Function", "Functions and CALL Routines: CALL MISSING Routine", "Functions and CALL Routines: N Function", "Functions and CALL Routines: CATS Function". http://support.sas.com/onlinedoc/913/docMainpage.jsp.

ACKNOWLEDGEMENTS

The author would like to thank Caroline Brickley of the SAS Institute Inc. for her helpful suggestions.

CONTACT INFORMATION

Your comments and questions are values and encouraged. Contact the author at

William C. Murphy
Howard M. Proskin & Associates, Inc.
300 Red Creek Dr., Suite 220
Rochester, NY 14623
Phone 585-359-2420
FAX 585-359-0465
Email wmurphy@hmproskin.com or wcmurphy@usa.net
Web www.hmproskin.com