ABSTRACT
Data files often contain missing data. All too often these missing values are represented by numeric codes (e.g. 99, -9). Unfortunately, the use of numeric codes creates a situation in which the programmer understands that the codes represent missing data, but SAS® views the codes as non-missing numeric values. This can lead to the accidental inclusion of missing codes in numeric calculations and other similar errors. To avoid this possibility, programmers often convert the numeric codes to the standard SAS missing value code (represented by a period) or write code that explicitly excludes the missing codes from calculations and reports. SAS special missing values present a better solution to this problem. They allow programmers to represent different types of missing values in a manner that allows SAS to recognize them as missing and treat them accordingly.

ALL MISSINGS ARE NOT CREATED EQUAL
Data files often contain several types of missing data. In survey research, for example, missing data may arise from any of the following sources:

- skip patterns
- respondent refused
- respondent did not know
- interviewer forgot to ask
- illegible response
- invalid (out-of-range) data

To an analyst the distinctions among types of missing values are often important. A high rate of refusal on sensitive questions may provoke concerns regarding biased data. Items missing due to skip patterns (e.g. number of cigarettes smoked by a non-smoker) may sometimes be recoded to zero. Trends in missing data may suggest a need for further interviewer training, or further revision of the instrument. How should these missing values be represented in a SAS data file?

THE FOLLY OF NUMERIC MISSING CODES
All too often missing values are represented with numeric values. The following sample data file includes three variables, each with different numeric codes for missing data.

The use of numeric missing codes causes a breakdown in communication between a programmer and SAS. The domain of values which can be represented in a SAS numeric variable can be divided into two mutually exclusive sets: non-missing data (theoretically all real numbers between negative infinity and positive infinity) and missing data. SAS has different rules for dealing with missing data and non-missing data. Programmers often think in terms of the same two sets, recognizing that missing data and non-missing data must be handled differently. The use of a numeric value to represent missing data creates a conflict in which the programmer views the data as part of the missing set, and SAS views the data as part of the non-missing set.

What are some of the problems that result from such coding?

- accidental inclusion of missing codes in numeric calculations
accidental inclusion of missing codes in format ranges
- missing codes are counted as non-missing by functions N(arg), NMISS(arg), MISSING(arg), etc.
- codes used for missings are inconsistent across variables, complicating reporting and analysis
- method will not work for date and time variables

The following program, designed to produce a simple report, illustrates several errors that may result from the use of numeric missing codes.

```sas
proc format;
  value AnnIncF
    low=49999 =’<$50k’
    50000-high=’>=50k’
  ;
run;

data NumMiss2;
  set NumMiss;
  MoInc=AnnInc/12; *compute monthly income;
run;
proc print data=NumMiss2 noobs;
run;
proc means data=NumMiss2 n nmiss mean
  maxdec=2;
  var Gender Age AnnInc;
run;
proc freq data=NumMiss2;
  tables AnnInc/nopercent nocum;
  format AnnInc AnnIncF.;
run;
```

The program produces the output depicted at the top of the next column.

A review of this output reveals a number of errors. The computation of monthly income (MoInc) in the DATA step treated the values 999999998 and 999999999 as non-missing data, resulting in three records having a value for MoInc of approximately $80 million. The PROC MEANS also included the missing codes in its calculations, and displays nonsensical results. Finally, the PROC FREQ included the missing codes for AnnInc in the category ">=50k" because the values 999999998 and 999999999 fall within the range '50000 - high'.

Typically programmers will choose one of two methods for dealing with numeric missing codes. They may engage in a search-and-destroy mission, converting all of the numeric missing values to SAS standard missings (represented by a period). Such practice results in a loss of information, as the modified file will not distinguish different types of missing data. Alternatively, programmers may accommodate the numeric missings by writing extra lines of code to explicitly exclude them from calculations (e.g. via conditional logic). When the missing codes vary across variables (as is common), such logic will have to vary as well, adding to the complexity of programs.

Both of these "solutions", however, are far from ideal. The former results in a loss of information. The latter places the burden on the programmer to write extra code, imposing a definition of missing set which is different from that understood by SAS. All of this would be much simpler if SAS could
incorporate the missing values into the missing set automatically, and therefore treat them appropriately.

**SPECIAL MISSINGS TO THE RESCUE**

Rather than lose the benefits of discriminating among types of missing values, or writing needlessly complex code to accommodate the use of numeric missing codes, programmers can make use of special missing values. These special missings (there are 27 of them: _.A_.B_.C..._.X_.Y_.Z) are understood by SAS to be part of the set of missing values. They may be used in any numeric variable, including dates and times.

Special missing values behave in a DATA or PROC step very much like standard missing values. Each special missing value has a defined rank in the numeric sort order: _. < . [standard missing] < .A < ... < .Z < [negative numbers] < 0 < [positive numbers]. This allows them to be used in BY-group processing, CLASS statements, etc. When used in calculations they propagate as standard missings (with the expected note to the log). Like standard missings, they are treated as FALSE in Boolean logic. Variable x may be tested for standard or special missings with: if MISSING(x). Testing for special missings alone can be done with: if MISSING(x) and x ^=.

Using the previous NumMiss data file as a starting point, it is easy enough to convert the numeric missing codes to special missing values:

```sas
data SpecMiss;
  set NumMiss;
  /******************************************************************************
  note: MISSING statement is not needed to use special missing values.
  ***************************************************************************/
  if      Gender=8 then Gender=.R; /*Refused*/
  else if Gender=9 then Gender=.M; /*Missing*/
  if      Age=888 then Age=.R;
  else if Age=999 then Age=.M;
  if      AnnInc=999999998 then AnnInc=.R;
  else if AnnInc=999999999 then AnnInc=.M;
  run;

proc format;
  value AnnIncF
    low-49999 ='<$50K'
    50000-high='>=$50K'
    .R='Refused'
    .M='Missing' ;
  run;
```

Running the reporting code on this file will produce the results shown in the figure at the top of the next column.

Note that the unformatted special missing values print as "R", "M" rather than ".R", ".M". It is necessary to refer to them in code as ".R" and ".M" so that SAS can identify them as special missing values rather than variable names. In the calculation of MoInc, the special missing values propagated as standard missings. If instead MoInc should take the special missing value from AnnInc, the code could be slightly modified to:

```sas
if NOT MISSING(AnnInc) then MoInc=AnnInc/12;
else MoInc=AnnInc;
```

The special missing values were correctly excluded from calculations of both the PROC MEANS and PROC FREQ. If the programmer wants to include rows for the missing values in the PROC FREQ output, the MISSPRINT or MISSING option added to the TABLES statement will produce the table on the top of the next page:

The code used to generate this report is identical to that used earlier. We have simply corrected the data, converting the missing codes to a form that SAS will recognize as missing. After that, SAS takes care of the rest.

In summary, the benefits of using special missing values include:
They are correctly treated as missing in DATA step and PROC step calculations.

- They maintain distinctions among different types of missing values.

- They allow consistent coding of missings across variables, files, and projects.

- They may be used in any numeric variable (including SAS dates and times).

- If all missings in a file are special missings, any standard missings generated in processing indicate an error (misspelled variable name, merge with no match, etc.).

- It is easy to convert from special missings to the standard missing, if needed.

- if MISSING(VAR) then VAR=.

HOW TO GET THERE FROM HERE

When the decision has been made to migrate from standard missings to special missings, the first step should be to design a coding scheme. One of the benefits of special missing values is that they are consistent across not only variables but also across data files and even across projects. It is worth investing time thinking about the types of missing values that exist in your data, and the number of different missing codes you want to support. Once a coding scheme has been developed, the conversion process itself is not at all difficult. Fairly basic macros can replace the hard coding employed in the example above. The following rudimentary macro, for example, takes as parameters a list of variables, the numeric missing codes used in those variables, and the special missing values being created.

```
%macro misscnv (varlist= /* X Y Z */, nummiss= /* 888 999 */, specmiss= /* .S .R .M */);
%local i j var numcode speccode;
%let i=1;
%let var=%scan(&varlist,&i);
%do %while (&var ne %str());
  select(&var);
  %let j=1;
  %let numcode =
    %scan(&nummiss ,&j,%str());
  %let speccode =
    %scan(&specmiss,&j,%str());
  %do %while (&numcode ne %str());
    %when (&numcode) &var=&speccode;
    %let j = %eval(&j+1);
    %let numcode =
      %scan(&nummiss ,&j,%str());
    %let speccode =
      %scan(&specmiss,&j,%str());
  %end;
  otherwise;
  end;
  %let i = %eval(&i+1);
%let var=%scan(&varlist,&i);
%end;
%mend;
```

```
data SpecMiss;
  set NumMiss;
  %misscnv(varlist=Gender ,nummiss=7 8 9 ,specmiss=.S .R .M )
  %misscnv(varlist=Age ,nummiss=777 888 999 ,specmiss=.S .R .M )
  %misscnv(varlist=AnnInc ,nummiss=999999997 999999998 999999999 ,specmiss=.S .R .M )
run;
```

CONCLUSIONS

SAS special missing values shift the burden of handling missing data off the programmer and onto SAS. They allow programmers to represent missing data in a manner that both simplifies coding and decreases the probability of embarrassing coding mistakes.

ACKNOWLEDGMENTS

Many thanks to Sigurd Hermansen, Ian Whitlock, and Mike Rhoads for their help in developing this paper.

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.

CONTACT INFORMATION

Quentin McMullen
Westat
1650 Research Boulevard
Rockville, MD 20850
quentinmcmullen@westat.com
Comments and questions greatly appreciated!