Paper 134-2018

The Missing Link: A Robust Macro for Recoding General Missing Data Values

Hunter Glanz, Statistics Department, California Polytechnic State University, San Luis Obispo;
Josh Horstman, Nested Loop Consulting

ABSTRACT

No matter the industry or field of study, missing values inevitably complicate a great many datasets and analyses. Deciding what to do with them in an analysis often depends on the analysis itself. Regardless of the uses and analyses, the data need to be clean and tidy first. Tidy data include, among other things, properly coded missing values. Unfortunately, domain-specific software, electronic surveys, and other data collection and organization tools do not necessarily code missing values in the ways that SAS wants: a period for numeric values and a blank for character values. SAS accepts a short list of other special characters as missing values also, but these aren't guaranteed any more than a period or blank. To make matters worse, non-numeric flags such as "NA" can cause SAS to read our numeric data into a character variable, rendering the information difficult to analyze. Most SAS users would not complain about re-coding missing values as long as the variable types were not altered by the employed missing value code, but this is not always the case. In this article we provide a robust macro for re-coding a given missing value code, to the period and blank that SAS prefer, while also converting variables back to their intended types.

INTRODUCTION

Suppose your boss gives you a data file stored in a simple comma-separated value (CSV) format. Along with this data file, you have access to a codebook which documents the numerous numerical and character variables in the data file. Fortunately, the codebook mentions that missing values only exist in a small handful of explicitly specified columns and are coded as -99. To work with these data in SAS, you know you will need to recode these values to a period or a blank depending on the variable type [3]. This task is made easy by the following facts:

- You know which variables contain missing values and their types
- The number of variables with missing values is small
- The missing value code of -99 will not change the intended types of each variable when this data file is read into SAS (i.e. numeric variables will be read in correctly as numeric and likewise for character variables)

We would consider ourselves lucky if working with a new dataset involved so few hurdles! Alas, data often come much messier than this. The inescapability of missing values in data demands a concise, user-independent, application-independent, and automated method for making datasets ready for SAS.

So, imagine that **at least one** of the following is true for you and your dataset:

1. You do not know which variables contain missing values
2. The number of variables with missing values is not small
3. The missing value code will change the intended types of each variable when the data file is read into SAS

Data can come from many different sources, including many which do not consider the SAS preference for periods (numeric) and blanks (character) as missing value representations. In what follows, we propose a method for recoding missing values, to those preferred by SAS, in the presence of any combination of the above three issues. To maximize ease-of-use, a macro version of the method is provided which takes a single argument/parameter: the missing value code in your original data.

SPECIAL MISSING VALUES IN SAS

SAS can actually already accommodate missing values coded as something other than a period or a blank! Foley (2005) provides a wonderful review of these capabilities, and so we only briefly cover them here [1].

Numeric variables can store 28 different missing values. With the use of the MISSING statement, we can tell SAS to identify the letters A-Z or an underscore as alternative missing numeric values. To reference these in a SAS expression or assignment statement, you must begin the value with a period, followed by the letter or underscore [4]. SAS will, of course, also recognize the blank as a numeric missing value and convert it to a dot on all inputs except LIST inputs.

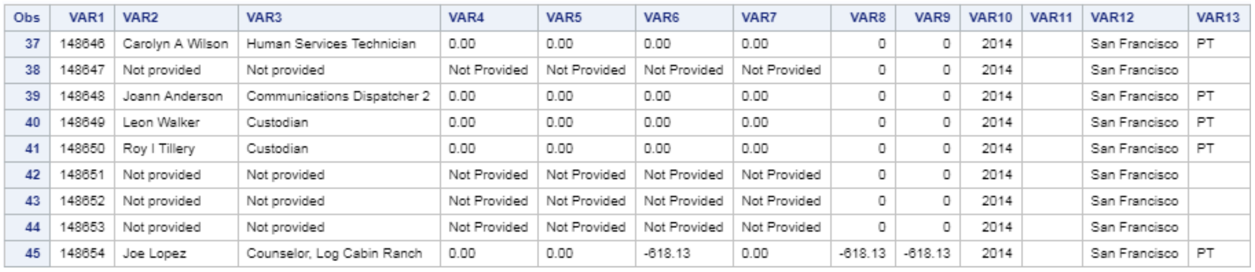
SAS officially recognizes only one missing character value: the blank. Unofficially, SAS almost always accepts the null character (ASCII code 0) and the special blank character (ASCII code 255) as missing as well.

While these accommodations could help address issue (3) above, the usefulness of them is limited to situations where missing value codes consist of a single character. Our work and proposed method were motivated by non-numeric missing value codes like "NA", "BLQ", "<xx", "Not Provided", etc.

AN AUTOMATED WAY TO RECODE GENERAL MISSING VALUES

The biggest missing value "thorn in our side" consists of a coding that changes the intended types of variables in our dataset. Even if the number of variables with missing values is small and we know which ones they are, the recoding involves manually creating new variables of the appropriate types and translating values over from the original variables. Not only does this process lack reproducibility, but it can cost the programmer more time than it should.

So, let us begin with a minimum working example. Consider the San Francisco salaries dataset found on kaggle.com [2]. This dataset contains the names, job title, compensation, and a few other variables for San Francisco city employees on an annual basis from 2011 to 2014. Figure 1 gives a screenshot of a snippet of the dataset.



Obs	VAR1	VAR2	VAR3	VAR4	VAR5	VAR6	VAR7	VAR8	VAR9	VAR10	VAR11	VAR12	VAR13
37	148646	Carolyn A Wilson	Human Services Technician	0.00	0.00	0.00	0.00	0	0	2014		San Francisco	PT
38	148647	Not provided	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0	0	2014		San Francisco	
39	148648	Joann Anderson	Communications Dispatcher 2	0.00	0.00	0.00	0.00	0	0	2014		San Francisco	PT
40	148649	Leon Walker	Custodian	0.00	0.00	0.00	0.00	0	0	2014		San Francisco	PT
41	148650	Roy I Tillery	Custodian	0.00	0.00	0.00	0.00	0	0	2014		San Francisco	PT
42	148651	Not provided	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0	0	2014		San Francisco	
43	148652	Not provided	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0	0	2014		San Francisco	
44	148653	Not provided	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0	0	2014		San Francisco	
45	148654	Joe Lopez	Counselor, Log Cabin Ranch	0.00	0.00	-618.13	0.00	-618.13	-618.13	2014		San Francisco	PT

Figure 1. Snippet of San Francisco salaries dataset.

As you can see via Figure 1, there appear to be multiple missing value codes with blanks in VAR11 and VAR13; "Not Provided" in VAR4-VAR7; and "Not provided" in VAR2 and VAR3. For our purposes, we will focus on the "Not Provided" code in VAR4-VAR7. This character missing value code forces SAS to treat these numeric variables as character variables. Again, it would be simple enough to write a custom DATA step to fix this problem but it would involve the following:

i) Manually creating new, numeric variables to store the converted values in.

ii) Using the input, or some other, function to manually convert these (VAR4-VAR7) four variables' character values.

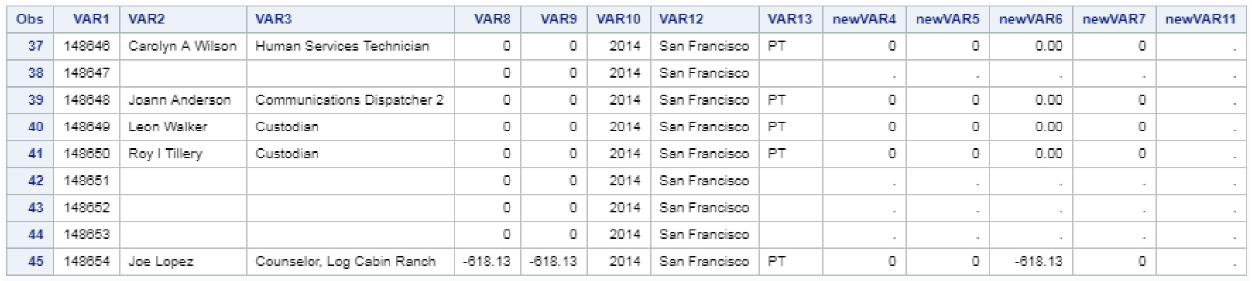
The term "manually" here refers to the fact that VAR4 through VAR7 (and the corresponding new variables) would need to be named explicitly since these are the problematic variables of interest. We propose a multi-step, data-driven approach to solving this issue:

1) Collect and store all of the variable names and some metadata about them in macro variables. Store the missing value code ("Not Provided" above) in a macro variable.

2) Recode all of the missing ("Not Provided") values, across all variables to the corresponding blank or period that SAS requires.

3) Drop variables that were the wrong type originally (i.e. character variables that were only character because of the "Not Provided" missing value code).

Figure 2 demonstrates the result after applying the above steps to the subset in Figure 1.



Obs	VAR1	VAR2	VAR3	VAR8	VAR9	VAR10	VAR12	VAR13	newVAR4	newVAR5	newVAR6	newVAR7	newVAR11
37	148646	Carolyn A Wilson	Human Services Technician	0	0	2014	San Francisco	PT	0	0	0.00	0	.
38	148647			0	0	2014	San Francisco		.	.	.	.	.
39	148648	Joann Anderson	Communications Dispatcher 2	0	0	2014	San Francisco	PT	0	0	0.00	0	.
40	148649	Leon Walker	Custodian	0	0	2014	San Francisco	PT	0	0	0.00	0	.
41	148650	Roy I Tillery	Custodian	0	0	2014	San Francisco	PT	0	0	0.00	0	.
42	148651			0	0	2014	San Francisco		.	.	.	.	.
43	148652			0	0	2014	San Francisco		.	.	.	.	.
44	148653			0	0	2014	San Francisco		.	.	.	.	.
45	148654	Joe Lopez	Counselor, Log Cabin Ranch	-618.13	-618.13	2014	San Francisco	PT	0	0	-618.13	0	.

Figure 2. Snippet of San Francisco salaries dataset with missing values correctly recoded into appropriate new values.

Notice in Figure 2 that the repaired numeric variables include "new" at the beginning of their variable names. This is for demonstration clarity. In the full macro, all of the original variable names are preserved. The macro is called **missfix** and takes three parameters: dsetin (input dataset), dsetout (output dataset), and missval (missing value code/string).   For details and implementation, we refer readers to the macro code found in Appendix 1, which is well-commented.

There is one thing to note when considering implementation with your own data. Your data may contain multiple missing value codes similar to our toy example ("Not Provided" and "Not provided"). While you should feel free to revise the macro to suit your needs, our suggestion here would be to just run the macro multiple times – once for each missing value code present in the data.

CONCLUSION

The abundance of missing data and the importance of addressing it in any kind of work with data cannot be overstated! While an extremely powerful tool, SAS requires special treatment and coding of missing values. If your data contain missing value codes that cause SAS to misidentify variable types, then the data cleaning portion of your work will grow. We provide a data-driven macro for re-coding missing values and restoring variables to their proper types. This tool completely removes this step from the data cleaning process, and does so in a reproducible way by not requiring users to hard code variable conversions.

REFERENCES

1. Foley, Malachy J. 2005. "MISSING VALUES: Everything You Ever Wanted to Know." *Proceedings of the Southeast SAS User Group Conference*. Available at https://analytics.ncsu.edu/sesug/2005/TU06_05.PDF

2. Kaggle. 2018. Accessed May 1, 2018. "SF Salaries." https://www.kaggle.com/kaggle/sf-salaries

3. SAS® 9.2 Language Reference: Concepts, Second Edition. 2018. "Working with Missing Values." Accessed May 1, 2018. https://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a002316433.htm

4. SAS® 9.2 Language Reference: Concepts, Second Edition. 2018. "Creating Special Missing Values." Accessed May 1, 2018.

https://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000992455.htm

CONTACT INFORMATION

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

Hunter Glanz
Statistics Department, California Polytechnic State University
1 Grand Avenue
San Luis Obispo, CA 93407
805-756-2792
hglanz@calpoly.edu

APPENDIX 1: FIXMISS MACRO CODE

```
**********************************************************************;
* MISSFIX Macro                                                      *;
*                                                                    *;
* Purpose: To create numeric variables from character variables so   *;
*          determined during import from an external file due to the *;
*          file due to the presence of non-numeric missing values.   *;
*                                                                    *;
* Authors: Hunter Glanz and Josh Horstman                            *;
* Date   : August 12, 2018                                           *;
*                                                                    *;
**********************************************************************;

%macro missfix(
       dsetin  = /* Name of input dataset */
      ,dsetout = /* Name of output dataset */
      ,missval = /* String to treat as missing (case insensitive) */
      );

      /* Determine whether DSETIN parameter includes a library name. */
      %IF %INDEX(&dsetin,.) %THEN %DO;
            %LET mylib = %SCAN(&dsetin,1,.);
            %LET myds  = %SCAN(&dsetin,2,.);
      %END;
      %ELSE %DO;
            %LET mylib = WORK;
            %LET myds  = &dsetin;
      %END;

      /* Get count and list of character variables. */
      proc sql noprint;
            select count(*),
                     name,
                     "new"||name into :varcnt trimmed,
                     :varlst separated by ' ', :newvarlst separated by ' '
                  from dictionary.columns
                  where memname="%UPCASE(&myds)"
                    and libname="%UPCASE(&mylib)" and type="char";
      quit;

      /* Create a new temporary dataset by attempting to convert character
         variables to numeric after removing missing value flags. */
      data _tmpfixed;
            set &dsetin end=eof;
            length dropvarlst $32767 renamelst $32767;

            /* Vars array will keep variables which were originally
               character.  Newvars array will contain numeric variables that
               were character only due to non-numeric missing value flags. */
            array vars    (&varcnt) $ &varlst;
            array newvars (&varcnt) &newvarlst;

            /* dropflag will track which variable should be dropped (1 = drop
               the original character var, 2 = drop the new numeric var) */
            array dropflag(&varcnt);
            retain dropflag:;
```

```
          /* Loop through character variables to identify numeric values
             and set flag accordingly. However, once decision has been made
             to drop new numeric variable and keep original character
             variable, subsequent records cannot alter this decision. */
          do i = 1 to dim(dropflag);
                num = input(vars(i),?? best32.);
                IsNum = (not missing(num)) or
                  strip(vars[i]) in ('','.') or
                  (.A le num le .Z) or (num eq ._);
                if upcase(vars(i)) = "%UPCASE(&missval)" then
                      call missing(vars[i],newvars[i]);
                else do;
                      dropflag[i] = max(ifn(IsNum,1,2),dropflag[i]);
                      newvars[i] = num;
                end;
          end;

          /* After all records are processed, create macro variables
             containing names of variables to be dropped and renamed. */
          if eof then do;
                do i = 1 to dim(dropflag);
                      if not missing(dropflag[i]) then
                            dropvarlst = catx(' ',
                                    dropvarlst,
                                    choosec(dropflag[i],
                                            vname(vars[i]),
                                            vname(newvars[i])));
                      if dropflag[i]=1 then
                            renamelst = catx(' ',
                                   renamelst,
                                   catx('=',
                                        vname(newvars[i]),
                                        vname(vars[i])));
                end;
                call symputx('dropvarlst', dropvarlst);
                call symputx('renamelst',  renamelst);
          end;

          drop dropvarlst renamelst dropflag: i num IsNum;
    run;

    /* Create the output data set by dropping and renaming
       variables as determined in the previous step. */
    data &dsetout;
          set _tmpfixed;
          %IF %bquote(&dropvarlst) ne %THEN drop &dropvarlst;;
          %IF %bquote(&renamelst) ne %THEN rename &renamelst;;
    run;

    /* Clean up by deleting the temporary data set. */
    proc delete data=_tmpfixed; run;

%mend missfix;
```