

Paper 025-31

MISSING! - Understanding and Making the Most of Missing Data

Suzanne M. Humphreys, PRA International, Victoria, BC (Canada)

ABSTRACT

Missing values can have a surprising impact on the way that data is dealt with. They can be represented and referenced by SAS® in a number of ways, there are various functions linked to missing values and procedures will have specific ways of handling them too. It can be a bit of a minefield but hopefully this paper will show some useful ways of dealing with missing data and how by using certain techniques you can turn this to your advantage. Topics will include the MISSING function, the NMISS function, special missing characters, different ways to identify missing data and MERGE vs UPDATE. The focus will be missing data in SAS data sets, rather than reading in raw data including missing values. Intended audience – beginner to intermediate, not limited to any particular operating system, based on SAS 8.2.

KEYWORDS: MISSING, NMISS, MERGE, UPDATE, MODIFY, survey data, questionnaire data, special missing characters

INTRODUCTION

Regardless of what sector of business you work in, if you deal with data then you will have missing data. It is what happens in the real world and sometimes it is a nuisance. SAS represents missing data in a number of ways. Usually the basic rule is that character values are represented by a blank (' ') or a null ("") and numeric values are represented by a single period (.).

Example 1: how SAS represents missing data in character and numeric variables

```
data miss1;
  input charmiss $ 1 nummiss 3;
cards;
A 1          1      A      1
B           2      B      .
3           3      3      3
D 4          4      D      4
;
```

There are also special characters that can be used to represent missing numeric data. Say for example, rather than just having a missing value you want to show why that value is missing (particularly useful with survey data) - not done, absent or refused? You can use special characters A - Z or _ (underscore).

Example 2: SAS special missing characters for numeric data

```
data miss2;
  input charmiss $ 1 nummiss 3-4;
  missing n a r _;
cards;
A -1          1      A      -1
A .           2      A      .
B r           3      B      R
3            4      3      3
D 0           5      D      0
a            6      A      A
E 6           7      E      6
F n           8      F      N
G _           9      G      _
;
```

In fact these special characters are all preceded by a period (i.e. .A) and this is how they should be referenced with SAS code, although you cannot see it in the print out. It can be quite advantageous to be able to differentiate

between types of missing data, especially if there are a large number. Missing data can be summarized by these characters or formats can be applied when listing them to label their true meaning.

Example 3: formats and special missing characters

	Obs	charmiss	nummiss
proc format;			
value spec .='Missing'			
._='Illegible'	1	A	-1
.R='Refused'	2	A	Missing
.N='Not Done'	3	B	Refused
.A='Absent';	4		3
run;	5	D	0
	6		Absent
proc print data=miss2;	7	E	6
var charmiss nummiss;	8	F	Not Done
format nummiss spec.;	9	G	Illegible
run;			

REFERENCING MISSING VALUES

Now we have seen the basics of how missing values can be represented in SAS, how can we reference them? There are a number of ways of picking out missing data. Firstly let us sort the data set by the variable nummiss. SAS orders numeric values (smallest to largest): _ (underscore), . (period), .A - .Z, negative numbers, 0, positive numbers.

Example 4: note how SAS orders the numeric values

	Obs	charmiss	nummiss
proc sort data=miss2 out=miss3;			
by nummiss;			
run;	1	G	_
	2	A	.
	3		A
	4	F	N
	5	B	R
	6	A	-1
	7	D	0
	8		3
	9	E	6

Each of the following steps will show different ways of referencing the missing values, check the print out of each data set to see the functionality of each method.

Example 5: where nummiss is not equal to period

	Obs	charmiss	nummiss
proc sort data=miss2(where=(nummiss			
ne .))			
out=miss3;			
by nummiss;			
run;	1	G	_
	2		A
	3	F	N
	4	B	R
	5	A	-1
	6	D	0
	7		3
	8	E	6

Example 6: where nummiss is not equal to period, ._, .R, .A and .N

```
proc sort data=miss2(where=(nummiss
                           not in (. _ .r .a .n)))
  out=miss3;
by nummiss;
run;
```

Obs	charmiss	nummiss
1	A	-1
2	D	0
3		3
4	E	6

Example 7: where nummiss is less than or equal to .Z

```
proc sort data=miss2(where=(nummiss
                           le .z))
  out=miss3;
by nummiss;
run;
```

Obs	charmiss	nummiss
1	G	-
2	A	.
3		A
4	F	N
5	B	R

Example 8: where nummiss is greater than .Z

```
proc sort data=miss2(where=(nummiss
                           gt .z))
  out=miss3;
by nummiss;
run;
```

Obs	charmiss	nummiss
1	A	-1
2	D	0
3		3
4	E	6

Example 9: where charmiss is not equal to blank

```
proc sort data=miss2(where=(charmiss
                           ne ''))
  out=miss3;
by nummiss;
run;
```

Obs	charmiss	nummiss
1	G	-
2	A	.
3	F	N
4	B	R
5	A	-1
6	D	0
7	E	6

If special missing characters are amongst your numeric data it is important to note that WHERE varname ne . will not exclude the special characters and WHERE varname gt .z must be used instead.

WHERE varname is not missing or WHERE varname is not null can be used for both character and numeric data; note that all missing data including special characters will be removed from the data set.

Example 10: where nummiss is not missing

```
proc sort data=miss2(where=(nummiss
                           is not missing))
  out=miss3;
by nummiss;
run;
```

Obs	charmiss	nummiss
1	A	-1
2	D	0
3		3
4	E	6

Example 11: where nummiss is not null

```

proc sort data=miss2(where=(nummiss
                          is not null))
    out=miss3;
by nummiss;
run;

```

Obs	charmmiss	nummiss
1	A	-1
2	D	0
3		3
4	E	6

MISSING AND NMISS FUNCTIONS

The MISSING function can be used with either character or numeric variables. This produces a numeric result (0 or 1) if the data point is present or missing. Special missing characters are treated as missing.

Example 12: using the missing function

```

proc sort data=miss2(where=(missing
                          (nummiss)))
    out=miss3;
by nummiss;
run;

```

Obs	charmmiss	nummiss
1	G	-
2	A	.
3		A
4	F	N
5	B	R

MISSING(varname) is the same as MISSING(varname)=1. MISSING(varname)=0 specifies when the data point is present. As the MISSING function can be used with either numeric or character variables it can be useful within a macro, in which case both numeric and character variables can be run through the same code.

The NMISS function has similar functionality to the MISSING function and returns the number of missing values within a list of variables defined. The differences between MISSING and NMISS are: only numeric variables can be used with NMISS and more than one variable can be specified. A numeric result is produced, its value depending on the number of missing data points. This function can be useful with survey data as the following example shows.

Example 13: TEST data set, showing a number of score variables, containing some missing values

```

data test;
    n=_n_;
    input score1 - score4;
cards;
    1 . . 3
    . 3 2 2
    2 2 2 2
    1 . . .
    . . . .
    3 3 2 1
    4 . 1 1
;

```

Obs	n	score1	score2	score3	score4
1	1	1	.	.	3
2	2	.	3	2	2
3	3	2	2	2	2
4	4	1	.	.	.
5	5
6	6	3	3	2	1
7	7	4	.	1	1

NMISS can be used to create a new variable, holding the value for the number of missing data points within the variables specified (for each observation). If your variables are named sequentially then you can use the 'of' syntax thereby not having to specify each variable in turn.

Example 13 (cont.): using the NMISS function

```
data countmiss;
set test;
miss_c=nmiss(of score1-score4); /*same as nmiss(score1,score2,score3,score4)*/
run;
```

Obs	n	score1	score2	score3	score4	miss_c
1	1	1	.	.	3	2
2	2	.	3	2	2	1
3	3	2	2	2	2	0
4	4	1	.	.	.	3
5	5	4
6	6	3	3	2	1	0
7	7	4	.	1	1	1

The new variable created (miss_c) can then be used to perform different actions on the data depending on their value (i.e. how many scores are missing).

```
data test2;
set countmiss;
by n;
if miss_c=0 then total=sum(of score1-score4);
else if miss_c lt 4 then flag=1;
else if miss_c=4 then flag=2;
run;
```

Obs	n	score1	score2	score3	score4	miss_c	total	flag
1	1	1	.	.	3	2	.	1
2	2	.	3	2	2	1	.	1
3	3	2	2	2	2	0	8	.
4	4	1	.	.	.	3	.	1
5	5	4	.	2
6	6	3	3	2	1	0	9	.
7	7	4	.	1	1	1	.	1

or simply, NMISS can be used in a WHERE clause to restrict observations by a number of missing data:

```
proc sort data=test(where=(nmiss(score1,score2,score3,score4) lt 2)) out=test1;
by score1;
run;
```

Obs	n	score1	score2	score3	score4
1	2	.	3	2	2
2	3	2	2	2	2
3	6	3	3	2	1
4	7	4	.	1	1

What if you want to see how many values are missing in a string of character variables? The MISSING function can be combined with the SUM function to act like the NMISS function, here is an example data set called TEST.

n	score1	score2	score3	score4
1	improvement			no change
2		worsening	worsening	no change
3	no change	improvement	no change	improvement
4	worsening			

Sum the number of missing values for each variable to mimic the NMISS function.

```
data countmiss;
  set test;
  by n;
  miss_c=sum(missing(score1),missing(score2),missing(score3),missing(score4));
run;
```

n	score1	score2	score3	score4	miss_c
1	improvement			no change	2
2		worsening	worsening	no change	1
3	no change	improvement	no change	improvement	0
4	worsening				3

If there are many sequentially named variables (for example, 50 score variables) a %DO loop can be used to restrict the amount of code required.

```
%macro dummy;
data countmiss;
  set test;
  by n;
  miss_c=sum(missing(score1),%do i=2 %to 49; missing(score&i),%end; missing(score50));
run;
%mend dummy;
%dummy;
```

HOW UPDATE AND MERGE DEAL WITH MISSING VALUES

UPDATE and MERGE are both ways of joining together two data sets. The official definitions of both are given below:

UPDATE - updates a master file by applying transactions

MERGE - joins observations from two or more SAS data sets into single observations ^[1]

MERGE combines observations from two or more SAS data sets. UPDATE combines observations from exactly two SAS data sets. UPDATE changes or updates the values of selected observations in a master data set and may also add observations.

An interesting difference between the two is the way, by default, that they deal with missing values. With UPDATE, any missing values in the overwriting data set will be ignored, whereas with MERGE any missing values in the overwriting data set (or transaction data set) will replace values from the base data set (or master data set).

The easiest way to see this difference is to look at an example based on common data found in the pharmaceutical industry; below there are two data sets MISSDT1 and MISSDT2. MISSDT1 holds the visit date for each of the visits listed, 1-5 (renamed to labdate). MISSDT2 holds lab sample dates for each visit listed, 1-5, some of which are missing.

Data set MISSDT1			Data set MISSDT2		
Obs	visit	labdate	Obs	visit	labdate
1	1	01JAN2006	1	1	01FEB2006
2	2	02JAN2006	2	2	.
3	3	03JAN2006	3	3	.
4	4	04JAN2006	4	4	04FEB2006
5	5	05JAN2006	5	5	05FEB2006

The aim is to keep labdate from data set MISSDT2 when it is present and when it is missing to replace it with the visit date (labdate) from MISSDT1. If the two data sets are merged together, the missing dates in MISSDT2 overwrite the

dates from MISSDT1:

```

data merged;
merge missdt1
      missdt2;
by visit;
run;

```

Obs	visit	labdate
1	1	01FEB2006
2	2	.
3	3	.
4	4	04FEB2006
5	5	05FEB2006

If UPDATE is used, the missing values in MISSDT2 do not overwrite the values from MISSDT1. Therefore, in the resulting data set we are keeping lab dates where populated and replacing missing dates with visit dates.

```

data updated;
update missdt1
      missdt2;
by visit;
run;

```

Obs	visit	labdate
1	1	01FEB2006
2	2	02JAN2006
3	3	03JAN2006
4	4	04FEB2006
5	5	05FEB2006

The default command that forces this behavior by UPDATE is the option UPDATEMODE=NOMISSINGCHECK (which is specified after the transaction or second data set), see the example syntax below:

```

UPDATEmaster-data-set<(data-set-options)> transaction-data-set<(data-set-options)>
<END=variable>
<UPDATEMODE= MISSINGCHECK|NOMISSINGCHECK>;
BY by-variable; [1]

```

It is important to note that special missing characters are the exception and will replace values in the master data set even when MISSINGCHECK (the default) is in effect. If some missing values need to be replaced and not others, some missing values can be set to special characters and others to periods in the transaction data set. In that case the special characters will not be replaced by values in the master or base data set, but the periods will, as in the following example:

Missing lab dates should be replaced by visit dates except at the baseline visit (visit 2).

Data set MISSDT1			Data set MISSDT2		
Obs	visit	labdate	Obs	visit	labdate
1	1	01JAN2006	1	1	01FEB2006
2	2	02JAN2006	2	2	Z
3	3	03JAN2006	3	3	.
4	4	04JAN2006	4	4	04FEB2006
5	5	05JAN2006	5	5	05FEB2006

```

data updated;
update missdt1
      missdt2;
by visit;
run;

```

Obs	visit	labdate
1	1	01FEB2006
2	2	Z
3	3	03JAN2006
4	4	04FEB2006
5	5	05FEB2006

The special character Z has not been replaced by the overwriting data set.

Usefully, if you want to represent that special character as a period (normal missing value) in the output data set you can use a single underscore as the special character, this can be used for numeric and character variables (when it will produce a blank).

Data set MISSDT1			Data set MISSDT2		
Obs	visit	labdate	Obs	visit	labdate
1	1	01JAN2006	1	1	01FEB2006
2	2	02JAN2006	2	2	—
3	3	03JAN2006	3	3	.
4	4	04JAN2006	4	4	04FEB2006
5	5	05JAN2006	5	5	05FEB2006


```

data updated;
  update missdt1
        missdt2;
  by visit;
run;

```

Obs	visit	labdate
1	1	01FEB2006
2	2	.
3	3	03JAN2006
4	4	04FEB2006
5	5	05FEB2006

The special underscore character has been represented as a period in the updated data set, the normal missing values have been replaced by the overwriting data set.

MODIFY

As a small aside it is interesting to note the behavior of MODIFY with missing values.

Like UPDATE, MODIFY combines observations from two SAS data sets by changing or updating values of selected observations in a master data set. The official definition is given below:

MODIFY - replaces, deletes, and appends observations in an existing SAS data set in place; does not create an additional copy^[1]

The option <UPDATEMODE=MISSINGCHECK|NOMISSINGCHECK> is specified in exactly the same way as for UPDATE and shows the same behavior. MISSINGCHECK is also the default value.

CONCLUSION

Missing data can be turned into useful tools if enough is understood about how SAS represents and deals with them. It is important to know your data, have special missing characters been used, are certain values expected to be missing? If robust programming techniques are not used then some special missing characters could be incorrectly included in DATA step processing and data misrepresented. Alternatively, reports can be clearer and provide more information if it is known that special missing characters provide extra meaning. If data is not expected to be missing use PUT statements to the log to alert you to any missing values, if certain data is expected to be missing make sure you have programmed to take account of this, think how the MISSING and NMISS functions can help you to process your data efficiently. Make sure you know how the common procedures you use deal with missing values by default and if there are options you can change to process them differently. If missing values are set up advantageously and the correct syntax is utilized, common pitfalls can be avoided.

REFERENCES

^[1] SAS OnlineDoc® documentation for SAS version 8.2.

Mathews, JoAnn. "Avoiding Mayhem in the New Millennium: Working with Missing Data" *Proceedings of the Twenty-Fifth Annual SAS User Group International Conference*, Indianapolis, IN. March 2000. <<http://www2.sas.com/proceedings/sugi25/25/btu/25p056.pdf>>

Phillips, Stacey D. and Klein, Gary. 2005. "Oh No, a Zero Row: 5 Ways to Summarize Absolutely Nothing" *Proceedings of the the Pharmaceutical SAS User Group Conference*, Phoenix, AZ. May 2005. <<http://www.lexjansen.com/pharmasug/2005/coderscorner/cc22.pdf>>

High, Robin. "How Statistical Programs Such as SAS Work with Missing Data" *Computing News, User Services and Network Applications Group, University of Oregon Computing Center*, Winter 2004. <<http://cc.uoregon.edu/cnews/winter2004/missingdata.html>>

Research Consulting, Information Technology Services at The University of Texas at Austin. "SAS FAQ #33: SAS missing values".

<<http://www.utexas.edu/its/rc/answers/sas/sas33.html>>

Dickman, Paul. "Missing values in SAS" SAS notes.

<<http://www.pauldickman.com/teaching/sas/missing.php>>

University of California Academic Technology Services. "Missing data in SAS" SAS Learning Module.

<<http://www.ats.ucla.edu/stat/sas/modules/missing.htm>>

Spicer, Jeanne Marie. The Pennsylvania State University Population Research Institute. "SAS Missing Values".

<<http://www.pop.psu.edu/help/cacpri/base/missing.htm>>

ACKNOWLEDGMENTS

The author wishes to acknowledge John Bridgman and Andi Duffes, whose original piece of programming advice prompted the idea for this paper. Also, thanks to colleagues Tim Williams and Stephen Hunt who gave their time to review and give valuable feedback.

CONTACT INFORMATION

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

Suzanne Humphreys
PRA International
300-730 View Street
Victoria, BC, Canada V8W 3Y7
Work Phone: 250-483-4404
Fax: 250-483-4588
E-mail: HumphreysSuzanne@PRAIntl.com
Web: www.prainternational.com

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.

Other brand and product names are trademarks of their respective companies.