Paper 1440-2014

What You're Missing About Missing Values

Christopher J. Bost, MDRC, New York, NY

ABSTRACT

Do you know everything you need to know about missing values? Do you know how to assign a missing value to multiple variables with one statement? Can you display missing values as something other than . or blank? How many types of missing numeric values are there? This paper reviews techniques for assigning, displaying, referencing, and summarizing missing values for numeric variables and character variables.

INTRODUCTION

Missing values are common. Most programmers know the essentials, but there are multiple statements, options, and functions for working with missing numeric values and missing character values. This paper includes basic and advanced techniques that can be used by beginning programmers and experienced programmers.

SAMPLE DATA

SAS® data set RETIRE stores information about retirement savings. It contains 10 observations and 10 variables. It has some missing numeric values and some missing character values. It also has some data entry errors.

Obs	id	jobstat	salary	contrib	ira	ira_amt	roth	roth_amt	deferred	def_amt
1	101	F	50000	1	Y	5500	Y	0	Y	2000
2	102	U	999999	7		.		.		.
3	103	P	24000	0	N	.	Y	2000	N	0
4	104	F	41000	1	Y	4000	N	0	N	0
5	105	F	.	8		.	Y	.		.
6	106	F	82500	.	N	0	Y	6500	Y	10000
7	107	P	13000	1	Y	1500	N	0	N	0
8	108		30000	9		.		.		.
9	109	F	104000	1	Y	4000	Y	1500	N	.
10	110	F	72000	.		.		.	Y	3000

Output 1: Print of data set RETIRE

#	Variable	Type	Len	Label
1	id	Num	8	Unique identifier
2	jobstat	Char	1	Working: F=Full-time, P=Part-time, U=Unemployed
3	salary	Num	8	Current salary: $
4	contrib	Num	8	Contributing: 0=No, 1=Yes, 7=N/A, 8=DK, 9=Refused
5	ira	Char	1	IRA: Y=Yes, N=No
6	ira_amt	Num	8	IRA: $
7	roth	Char	1	Roth IRA: Y=Yes, N=No
8	roth_amt	Num	8	Roth IRA: $
9	deferred	Char	1	Deferred: Y=Yes, N=No
10	def_amt	Num	8	Deferred: $

Output 2: Contents of data set RETIRE

ASSIGNING MISSING VALUES

Assign a missing numeric value as a period. For example, assign a missing value to numeric variable X:

```
x=.;
```

Assign a missing character value as a single blank. For example, assign a missing value to character variable Y:

```
y=' ';
```

A missing character value must be enclosed in single quotes or double quotes.

Note that SAS determines the length of a variable the first time it is encountered. The default length of 8 bytes for numeric variables is usually fine. Use a LENGTH statement as needed to assign the length of a character variable. For example:

```
length z $ 10;
```

The above statement creates a character variable named Z that is 10 bytes long. It is initialized to a missing value.

In the RETIRE data set, someone who is unemployed should have a missing value for SALARY. Assign a missing numeric value as follows:

```
if jobstat='U' then salary=.;
```

Someone who has a missing value for IRA_AMT should have a missing value for the IRA variable. Assign a missing character value as follows:

```
if ira_amt=. then ira=' ';
```

ADVANCED

Sometimes more than one variable should be assigned a missing value. In the RETIRE data set, if someone does not currently contribute to any retirement plan, all of the retirement plan variables and all of the retirement amount variables should not have values.

Missing values can be assigned with assignment statements. For example:

```
if contrib ne 1 then do;
  ira=' ';
  ira_amt=.;
  roth=' ';
  roth_amt=.;
  deferred=' ';
  def_amt=.;
end;
```

If the value of CONTRIB does not equal 1, this code assigns missing character values to IRA, ROTH, and DEFFERED, and assigns missing numeric values to IRA_AMT, ROTH_AMT, and DEF_AMT. Six assignment statements are required.

Missing values can also be assigned using arrays. Note that numeric variables and character variables must be defined in separate arrays. For example:

```
array pln{3} ira roth deferred;
array amt{3} ira_amt roth_amt def_amt;
if contrib ne 1 then do i=1 to 3;
  pln{i}=' ';
  amt{i}=.;
end;
```

If the value of CONTRIB does not equal 1, this code assigns a missing character value to each variable in array PLN and a missing numeric value to each variable in array AMT. Two ARRAY statements and two assignment statements are required.

The above two approaches require a fair amount of code. It is possible to use a single CALL MISSING statement to assign missing values to multiple variables. For example:

```
if contrib ne 1 then call missing(ira,ira_amt,roth,roth_amt,deferred,def_amt);
```

CALL MISSING works on numeric variables and/or character variables. If the value of CONTRIB does not equal 1, this code assigns missing character values to IRA, ROTH, and DEFERRED, and assigns missing numeric values to IRA_AMT, ROTH_AMT, and DEF_AMT.

CALL MISSING arguments can also be array references and variable lists. For example:

```
if contrib ne 1 then call missing(of pln{*},of amt{*});

if contrib ne 1 then call missing(of ira--def_amt);
```

The arguments to CALL MISSING in the first statement reference all variables in array PLN and all variables in array AMT. The argument to CALL MISSING in the second statement references all variables from IRA through DEF_AMT based on their positional order in the data set.

Note that both of the array references and the positional list of variables must be preceded by OF when used as arguments to functions.

All of the techniques in this section achieve the same results, but the last statement using CALL MISSING and a positional list of variables is the shortest and simplest approach.

SPECIAL MISSING VALUES

There are many reasons a variable value might be missing. For example, an answer to a survey question might be missing because it did not apply to the respondent, because the respondent did not know the answer, or because the respondent refused to answer. In addition, an answer might be missing for an unknown reason.

In the RETIRE data set, the variable CONTRIB equals 1 if the person currently contributes to any retirement plan and 0 if the person does not. CONTRIB equals 7 if the person is not eligible to contribute, 8 if the person did not know the answer, and 9 if the person refused to answer. CONTRIB is also simply missing (.) for two observations.

The values 7, 8, and 9 are not “yes or no” responses. These values might be recoded to missing. For example:

```
if contrib in(7,8,9) then contrib=.;
```

The above IF-THEN statement replaces 7, 8, and 9 with a missing value, but it lumps all three values together. “Not Applicable,” “Don’t Know,” and “Refused” are qualitatively distinct missing values. It is possible to assign special missing values to numeric variables to distinguish different types of missing data. For example:

```
     if contrib=7 then contrib2=.A; /* Not Applicable */
else if contrib=8 then contrib2=.K; /* Don't Know */
else if contrib=9 then contrib2=.R; /* Refused */
else contrib2=contrib;
```

Special missing values can be .A, .B, .C through .X, .Y, .Z as well as ._ (“dot underscore”). The letter in the special missing value may be uppercase or lowercase. For example, .A and .a are equivalent.

DISPLAYING MISSING VALUES

SAS displays missing values differently depending on the variable type. Missing numeric values are displayed as a period. Missing character values are displayed as a blank space. Special missing values are displayed as the upper case letter with no leading period. For example:

Obs	jobstat	contrib	contrib2
1	F	1	1
2	U	7	A
3	P	0	0
4	F	1	1
5	F	8	K
6	F	.	.
7	P	1	1
8		9	R
9	F	1	1
10	F	.	.

Output 3: Special missing values

Printed values of CONTRIB2 are 1 and 0 (non-missing values) and ., A, K, and R (missing numeric values and special missing values). The special missing values are still stored as .A, .K, and .R in the SAS data set.

Use the MISSING option to control how ordinary (not special) missing values for numeric variables are displayed. A period is the default. You can specify an alternate character. For example:

```
options missing=*;
```

The value after the equals sign can be any single character. The above OPTIONS statement tells SAS to print ordinary missing values as an asterisk (*). Note that special missing values are still printed as A, K, and R. For example:

```
Obs    contrib    contrib2

  1       1           1
  2       7           A
  3       0           0
  4       1           1
  5       8           K
  6       *           *
  7       1           1
  8       9           R
  9       1           1
 10       *           *
```

Output 4: MISSING= option

ADVANCED

Missing numeric values and missing character values can be formatted. For example:

```
proc format;
value $jobstat2f ' '='Missing'
                 'F'='Full-time'
                 'P'='Part-time'
                 'U'='Unemployed';
value contrib3f   0='No'
                  1='Yes'
                 . ='Missing'
                 .A='Not Applicable'
                 .K="Don't Know"
                 .R='Refused';
run;
```

Use a FORMAT statement to assign formats to the respective numeric variable(s) and/or character variable(s) and display missing values with the specified value labels. For example:

```
format jobstat2 $jobstat2f. contrib3 contrib3f.;
```

Variables JOBSTAT2 and CONTRIB3 are copies for illustrative purposes. PROC PRINT displays the following:

```
Obs    jobstat     jobstat2     contrib    contrib2    contrib3

  1       F       Full-time        1           1       Yes
  2       U       Unemployed       7           A       Not Applicable
  3       P       Part-time        0           0       No
  4       F       Full-time        1           1       Yes
  5       F       Full-time        8           K       Don't Know
  6       F       Full-time        .           .       Missing
  7       P       Part-time        1           1       Yes
  8               Missing          9           R       Refused
  9       F       Full-time        1           1       Yes
 10       F       Full-time        .           .       Missing
```

Output 5: Formatted missing values

The missing character value for JOBSTAT2 is labeled “Missing.” The missing numeric values for CONTRIB3 are labeled “Missing” and the special missing values (.A, .K, and .R) are labeled “Not Applicable,” “Don’t Know,” and “Refused,” respectively.

REFERENCING MISSING VALUES

Missing numeric values and missing character values can be referenced in expressions.

NUMERIC VARIABLES

Check if a numeric variable is missing by comparing it to a period. For example:

```
if contrib=.;
```

This statement evaluates whether the numeric variable CONTRIB is missing. (Always reference missing numeric values this way even if MISSING= is used in an OPTIONS statement. It only tells SAS how to print missing values. The internal value is still missing.)

Check if a numeric variable equals a particular special missing value by specifying . and the respective letter or an underscore. For example:

```
if contrib2=.R;
```

This statement evaluates whether CONTRIB2 equals .R. The letter can be specified in uppercase or lowercase.

ADVANCED

Check if a numeric variable is not missing and is not a special missing value. For example:

```
if contrib2 > .Z;
```

Note that the sort order of numeric values from lowest to highest is:

._
.
.A
.B
.C
[continues]
.X
.Y
.Z
negative numbers
0
positive numbers

Table 1: Sort order of numeric values

CHARACTER VARIABLES

Check if a character variable is missing by comparing it to a blank space. For example:

```
if jobstat=' ';
```

Enclose a single space in quotes (single or double) regardless of the variable length.

FUNCTIONS

Use the MISSING function to check if a variable is missing. For example:

```
if missing(jobstat);

if missing(salary);
```

These statements check if the specified variable is missing. The variable can be character or numeric. If a character value is a blank, the MISSING function returns a 1. If a numeric value is missing (including special missing values), the MISSING function returns a 1. If a character value or a numeric value is not missing, the MISSING function returns a 0.

ADVANCED

Programmers frequently need to perform an action when the value of a particular variable is not missing. A common technique is the following subsetting IF statement:

```
if contrib;
```

Programmers often assume that an observation with a non-missing value of CONTRIB meets the above condition. That is not correct. The above statement evaluates whether the value of CONTRIB is true. A true numeric value is any value that is not missing (including special missing values) and not equal to 0. The above statement is actually the equivalent of:

```
if contrib > .Z and contrib ne 0;
```

IF *variable;* can be problematic when it references a character variable. SAS first attempts to convert the value of the variable from character to numeric and notes this in the Log. SAS then checks if the value is true as described above. If the original value is not a number, SAS notes “invalid numeric data” in the Log.

Use the MISSING function to check if a variable is not missing. For example:

```
if not missing(contrib);

if missing(contrib)=0;
```

Both statements produce the same results (i.e., subset observations where CONTRIB is not a missing value or a special missing value). The argument to the MISSING function can be a numeric variable or a character variable.

SUMMARIZING MISSING VALUES

Missing values can be summarized within observations and across observations. The techniques are different for numeric variables and character variables.

WITHIN OBSERVATIONS: NUMERIC VARIABLES

Use the NMISS function to count the number of missing numeric arguments. For example:

```
badcount=nmiss(ira_amt,roth_amt,def_amt);
```

BADCOUNT stores the number of arguments that are missing values or special missing values.

Conversely, use the N function to count the number of non-missing numeric arguments. For example:

```
goodcount=n(ira_amt,roth_amt,def_amt);
```

GOODCOUNT stores the number of arguments that are not missing values or special missing values.

Variables can be printed with PROC PRINT for inspection:

Obs	ira_amt	roth_amt	def_amt	badcount	goodcount
1	5500	0	2000	0	3
2	.	.	.	3	0
3	.	2000	0	1	2
4	4000	0	0	0	3
5	.	.	.	3	0
6	0	6500	10000	0	3
7	1500	0	0	0	3
8	.	.	.	3	0
9	4000	1500	.	1	2
10	.	.	3000	2	1

Output 6: NMISS and N functions

WITHIN OBSERVATIONS: CHARACTER VARIABLES

Use the CMISS function to count the number of missing arguments. (Arguments can be character and/or numeric.) For example:

```
badcount2=cmiss(ira,roth,deferred);
```

As of SAS 9.3, there is no function that counts the number of non-missing character arguments. It is possible, however, to do this with nested functions. For example:

```
goodcount2=countw(catx('*',ira,roth,deferred));
```

The CATX function concatenates all of the arguments separated by an asterisk (*).The COUNTW function counts the number of “words” separated by delimiters. The asterisk is one of the default delimiters. If an asterisk might be included in the value of an argument, specify another character as the first argument to the CATX function and the second argument to the COUNTW function. For example:

```
goodcount3=countw(catx('\',ira,roth,deferred),'\');
```

The CATX function concatenates all of the arguments separated by a backslash (\). The COUNTW function counts the number of “words” separated by a backslash (only). Specifying a delimiter overrides the default delimiters for COUNTW.

Variables can be printed with PROC PRINT for inspection:

Obs	ira	roth	deferred	badcount2	goodcount2	goodcount3
1	Y	Y	Y	0	3	3
2				3	0	0
3	N	Y	N	0	3	3
4	Y	N	N	0	3	3
5		Y		2	1	1
6	N	Y	Y	0	3	3
7	Y	N	N	0	3	3
8				3	0	0
9	Y	Y	N	0	3	3
10			Y	2	1	1

Output 7: CMISS and COUNTW functions

ACROSS OBSERVATIONS: NUMERIC VARIABLES

Use PROC MEANS and the NMISS statistic-keyword to count the number of missing values and special missing values across observations. For example:

```
proc means data=retire nmiss n;
run;
```

PROC MEANS produces the following output:

The MEANS Procedure

Variable	Label	N Miss	N
id	Unique identifier	0	10
salary	Current salary: $	1	9
contrib	Contributing: 0=No, 1=Yes, 7=N/A, 8=DK, 9=Refused	2	8
ira_amt	IRA: $	5	5
roth_amt	Roth IRA: $	4	6
def_amt	Deferred: $	4	6

Output 8: PROC MEANS with NMISS and N

Statistics are calculated for all numeric variables by default. Use the VAR statement to specify variables (optional).

The PROC MEANS step above also specifies the N statistic-keyword to display the number of non-missing values. It is useful to know the number of missing values and the number of non-missing values.

ACROSS OBSERVATIONS: CHARACTER VARIABLES

Use PROC FORMAT and PROC FREQ to summarize the number of missing values and non-missing values across observations. For example:

```
proc format;
value $charf ' '='Missing'
             OTHER='Not missing';
run;

proc freq data=retire;
tables _character_/missing;
format _character_ $charf.;
run;
```

The PROC FORMAT step creates a format named $CHARF. that assigns the label “Missing” to missing character values. The OTHER= keyword assigns the label “Not missing” to all other values.

The TABLES statement specifies a one-way frequency table for each character variable.

PROC FREQ excludes missing values by default. On the TABLES statement, use the MISSING option after a slash to include missing values in the tabulations.

The FORMAT statement assigns the $CHARF. format to all character variables.

PROC FREQ produces the following output:

```
The FREQ Procedure

        Working: F=Full-time, P=Part-time, U=Unemployed

                                          Cumulative    Cumulative
jobstat        Frequency     Percent     Frequency      Percent
-----------------------------------------------------------------
Missing                1       10.00             1        10.00
Not missing            9       90.00            10       100.00


                      IRA: Y=Yes, N=No

                                          Cumulative    Cumulative
ira            Frequency     Percent     Frequency      Percent
-----------------------------------------------------------------
Missing                4       40.00             4        40.00
Not missing            6       60.00            10       100.00


                   Roth IRA: Y=Yes, N=No

                                          Cumulative    Cumulative
roth           Frequency     Percent     Frequency      Percent
-----------------------------------------------------------------
Missing                3       30.00             3        30.00
Not missing            7       70.00            10       100.00


                   Deferred: Y=Yes, N=No

                                          Cumulative    Cumulative
deferred       Frequency     Percent     Frequency      Percent
-----------------------------------------------------------------
Missing                3       30.00             3        30.00
Not missing            7       70.00            10       100.00
```

Output 9: Summarizing missing character values

ADVANCED

Use PROC FORMAT and PROC FREQ to summarize the number of missing values and non-missing values across observations for all variables, whether character or numeric. For example:

```
proc format;
value $charf ' '='Missing'
             OTHER='Not missing';
value numf LOW-HIGH='Not missing';
run;

proc freq data=retire;
tables _all_/missing nocum;
format _character_ $charf.
       _numeric_ numf.;
run;
```

The PROC FORMAT step creates two formats. The $CHARF format again assigns the label “Missing” to missing character values and the label “Not missing” to all other values.

The NUMF format assigns the label “Not missing” to all non-missing values, from the LOWest to the HIGHest. All other values (i.e., missing values and special missing values) are not formatted and will be displayed as-is.

The PROC FREQ step specifies a one-way frequency table for each variable, whether character or numeric.

The MISSING option tells SAS to include missing values (and special missing values for numeric variables) in the tabulations.

The FORMAT statement assigns the $CHARF. format to all character variables and the NUMF. format to all numeric variables.

Note that the NOCUM option was used on the TABLES statement to reduce output (i.e., eliminate cumulative statistics) and fit the results on this page.

Frequencies below include all of the original variables in SAS data set RETIRE plus the CONTRIB2 variable.

Unique identifier

id	Frequency	Percent
Not missing	10	100.00

Working: F=Full-time, P=Part-time …

jobstat	Frequency	Percent
Missing	1	10.00
Not missing	9	90.00

Current salary: $

salary	Frequency	Percent
.	1	10.00
Not missing	9	90.00

Contributing: 0=No, 1=Yes, 7=N/A, …

contrib	Frequency	Percent
.	2	20.00
Not missing	8	80.00

IRA: Y=Yes, N=No

ira	Frequency	Percent
Missing	4	40.00
Not missing	6	60.00

IRA: $

ira_amt	Frequency	Percent
.	5	50.00
Not missing	5	50.00

Roth IRA: Y=Yes, N=No

roth	Frequency	Percent
Missing	3	30.00
Not missing	7	70.00

Roth IRA: $

roth_amt	Frequency	Percent
.	4	40.00
Not missing	6	60.00

Deferred: Y=Yes, N=No

deferred	Frequency	Percent
Missing	3	30.00
Not missing	7	70.00

Deferred: $

def_amt	Frequency	Percent
.	4	40.00
Not missing	6	60.00

Contributing 2: 0=No, 1=Yes, 7=.A,…

contrib2	Frequency	Percent
.	2	20.00
A	1	10.00
K	1	10.00
R	1	10.00
Not missing	5	50.00

Output 10: Summarizing missing numeric and character values

CONCLUSION

There is more to missing values than most people realize. Programmers need to understand how SAS handles missing numeric values and missing character values in order to write accurate and efficient code. Keep these techniques in your SAS “toolkit” to assign, display, reference, and summarize missing values.

CONTACT INFORMATION

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

Christopher J. Bost
MDRC
16 East 34th Street
New York, NY 10016
(212) 340-8613
christopher.bost@mdrc.org
chrisbost@gmail.com

What You're Missing About **Missing Values**

Assigning

- Assign a missing numeric value as a period:

```
x=.;
```

- Assign a missing character value as a blank:

```
y=' ';
```

A missing character value must be enclosed in single quotes or double quotes. (Use a preceding LENGTH statement as needed to set the variable length.)

- Use CALL MISSING to assign a missing value to one or more variables:

```
if z=1 then call missing(a,b,c,d,e);
```

Arguments to CALL MISSING can be numeric variables and/or character variables.

Arguments to CALL MISSING can also be (previously defined) arrays and/or variable lists:

```
if z=1 then call missing(of arrayname{*});
if z=1 then call missing(of a--e);
```

Array references and variable lists must be preceded by OF when used as function arguments.

- Assign special missing values .A, .B, .C through .X, .Y, .Z as well as ._ ("dot underscore") to distinguish missing values:

```
if income=9 then income=.R;
```

The letter in the special missing value can be uppercase or lowercase. For example, .R and .r are equivalent.

The sort order of numeric values from lowest to highest is:

._	.	.A	.B	.C	(con't)	.X	.Y	.Z	negative	0	positive

Displaying

- Use the MISSING= option to specify a character other than the period to represent a missing numeric value:

```
options missing=*;
```

- Use PROC FORMAT to assign labels to missing character values and/or missing numeric values, including special missing values:

```
proc format;
value $charfmt ' ' = 'Missing';
value numfmt    .  = 'Missing'
               .R = 'Refused';
run;
```

Referencing

- Reference missing values in expressions:

```
if x=.;     Checks if variable is a numeric missing value.
if y=' ';   Checks if variable is a character missing value.

if income=.R;   Checks if variable equals specified special missing value.
if income > .Z; Checks if variable is not missing and not special missing.

if missing(income); Checks if variable is missing or special missing.

if not missing(income); Checks if variable is not missing and not
if missing(income)=0;   special missing. (Equivalent statements.)

if income;                          Subsetting IF statement.
if income > .Z and income ne 0;     Equivalent statement.
```

Summarizing

- Use NMISS to summarize missing numeric values within each observation:

```
badcount1=nmiss(a,b,c);
```

- Use CMISS to summarize missing character values (and/or numeric values) within each observation:

```
badcount2=cmiss(x,y,z);
```

- Use PROC MEANS to summarize missing numeric values across observations:

```
proc means data=dsname nmiss;
var variable(s);
run;
```

- Use PROC FREQ to summarize missing character values and missing numeric values across observations:

```
proc format;
value $charf ' '='Missing'
             OTHER='Not missing';
value numf LOW-HIGH='Not missing';
run;

proc freq data=dsname;
tables _all_ /missing;
format _character_ $charf.
       _numeric_ numf.;
run;
```

Use the MISSING option after a slash (/) on the TABLES statement to include missing values (and special missing values for numeric variables) in the tabulations.

Non-missing values for each character variable and each numeric variable are labeled "Not missing" and counted as a single category.