Paper CC05

Missing Values in SAS
Magnus Mengelbier, Limelogic Ltd, London, United Kingdom

ABSTRACT
Missing values within Life Sciences can be an interesting exercise, not just how they are used in programming and analysis but also how they are used as a tool to code and store data. Whether the value is not applicable, below a level of quantification or just simply missing, is sometimes not trivial. SAS treats missing values as a very valid value and provides powerful functions that enable great control over their processing.

INTRODUCTION
A missing value in SAS is a valid value or a special case, which is dependent on SAS functions, statements or procedures. This makes missing values important to manage consistently in order to avoid unexpected results with little or no indication and very limited information.

The ability to identify or check for missing values is in most cases more important than to determine the reason for or the actual method of assigning missing as a value. SAS provides a set of simple functions and statements that allow assigning missing values and checking if a variable value is missing or determine the number of variables with missing or non-missing values.

The functions and methods are discussed with DATA step examples, but PROC SQL is also applicable in all but a few cases. Most functions are also available in SAS macro code with slightly different behaviour, which is beyond the scope for this paper.

MISSING VALUES
A missing value in SAS is a valid value, just like the number 7 or this string of "characters", or it is treated as a special case, which is entirely dependent on what SAS function, statement or procedure is executing at the time. This makes missing values a potential pitfall that can generate unexpected results with no or very limited indication of what actually happened.

The missing value is not consistent in SAS as it differs between numeric and character variables. For a number, a missing value is usually denoted by a single period ("."). A missing character value is an empty string. Converting a missing number to character with the PUT() function does not produce an empty string as would be logical – missing converted to missing – but rather the resulting character value contains a single period, which of course is not a missing character value.

```sas
27   data _null_;  
28   29      length char $ 10 ;  
30  31      num = . ;  
32      char = put( num, 8.-L);  
33  34      put num = / char = ;  
35  36      if missing(num) then put "num is missing";  
37      if missing(char) then put "char is missing";  
38   run;  
num=.  
char=.  
num is missing
```
The above example indicates some of the interesting facets of missing values in SAS. The MISSING() function will be discussed a bit further along in this paper, but for now it is simply a test if SAS considers the value of char missing, which the example shows that it is not.

The period within the character string can be circumvented by assigning a different default character to denote a missing numeric value with the missing system option. If we set the missing option to a space, then the character value suddenly becomes missing – missing number converted to a missing character value.

```sas
44 option missing = '';  
45 46 data _null_;  
47 48 length char $ 10 ;  
49 50 num = . ;  
51 char = put( num, 8.-L);  
52 53 put num = / char = ;  
54 55 if missing(num) then put "num is missing";  
56 if missing(char) then put "char is missing";  
57 run;
```

In the above example, a very common programming approach to test for missing fails to detect the missing value, if we assume that the MISSING() function has correctly determined that it is missing. One perspective to consider is if
the BLQ value is actually missing or just noted as small enough that by all intends and purposes we cannot accurately determine its value – or it is simply small enough to be considered missing. That discussion is dependent on your statistical analysis and is far beyond the scope of this paper, but we are provided the tools to judge on an analysis basis since SAS treats missing values as any valid value.

Missing codes however are not new to SAS. The missing codes .A, .C, .D, and .M are still used within Life Science data but maybe not as extensively as a decade ago.

ASSIGNING A MISSING VALUE
There are several different approaches and possible conventions to assign a missing value to a numeric or character variable. By default, variables uninitiated, e.g. no values previously assigned, are assigned a missing value.

```
81   data _null_;
82     num_3 = w;
83     put w = / num_1 = / num_2 = / num_3 = ;
84   run;
```

```
NOTE: Variable w is uninitialized.
NOTE: Variable num_1 is uninitialized.
NOTE: Variable num_2 is uninitialized.
```

```
num_1=.
num_2=.
num_3=.
```

Note that the variable `num_3` is missing from the list of SAS Notes on uninitialized because `num_3` is assigned the value missing from the variable `w`, which in turn is assigned the default missing value. Both `w` and `num_3` as all variables in the example are numeric since we have not specified otherwise.

If we consider the second DATA step below, we also see that assigning a value for the missing system option does not make that value, character ‘w’ in our case, indicate a missing value, i.e. the SAS Note that variable `w` is uninitialized. This role is still reserved for the period or the missing code and it is no coincidence that the missing code starts with a period. Also, consider the `num_1` variable below, which is assigned a missing value (period notation) in the first DATA step and subsequently uses the character specified with the `missing` system option (‘w’) when we output the value to the log in the second DATA step using the `PUT` statement. The missing value is an internal SAS data construct with the missing option a general presentation “format” restricted to one character in length.

```
92   data work.missing ;
93      num_1 = .;
94      num_2 = .B;
95   run;
```

```
NOTE: The data set WORK.MISSING has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time       0.00 seconds
      cpu time        0.00 seconds
```

```
96   option missing = 'w';
97   data _null_;
98      set work.missing ;
99      put num_1 = / num_2 = ;
100     num_3 = w;
```

```
NOTE: The data set WORK.MISSING has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time       0.00 seconds
      cpu time        0.00 seconds
```
105  run;

NOTE: Variable w is uninitialized.
num_1=w
num_2=B
NOTE: There were 1 observations read from the data set WORK.MISSING.
NOTE: DATA statement used {Total process time}:
   real time           0.01 seconds
   cpu time            0.00 seconds

106 107  option missing = '.';  *  reset the missing to be a period again ;

SAS provides the statement CALL MISSING() to explicitly initialise or set a variable value to be missing. Note that
the CALL MISSING() statement can of course be used within a conditional if-statement and block.

113  data _null_;
114 115    num_3 = w;
116 117    put num_1 = / num_2 = / num_3 = ;
118 119  run;

NOTE: Variable w is uninitialized.
NOTE: Variable num_1 is uninitialized.
NOTE: Variable num_2 is uninitialized.
num_1=.
num_2=.
um_3=.
NOTE: DATA statement used {Total process time}:
   real time           0.00 seconds
   cpu time            0.00 seconds

120
121
122  data _null_;
123 124    call missing( num_1, num_2, w );
125 126    num_3 = w;
127 128    put num_1 = / num_2 = / num_3 = ;
129 129  run;

num_1=.
num_2=.
um_3=.
NOTE: DATA statement used {Total process time}:
   real time           0.05 seconds
   cpu time            0.01 seconds

The CALL MISSING() statement is a convenient convention to require that all variables are explicitly initialised as
missing and the SAS Note “Variable … is uninitialized.” can become a key indicator of data issues or a case not
handled properly within a program.
TESTING FOR MISSING VALUES
There are several different approaches to test or check for a missing value, applicable to both the DATA step and PROC SQL statements. The classic approach that is still very common today is to check for the ‘period’.

```sas
139 data _null_;  
140   set work.missing ;  
141  
142   put num_1 = / num_2 = ;  
143  
144   if ( num_1 = . ) then put "num_1 is missing";  
145   if ( num_2 = . ) then put "num_2 is missing";  
146  
147 run;
num_1=.
num_2=B
num_1 is missing
NOTE: There were 1 observations read from the data set WORK.MISSING.
NOTE: DATA statement used (Total process time):
   real time           0.00 seconds
   cpu time            0.00 seconds

148  
149  option missing = 'w';  
150  
151 data _null_;  
152   set work.missing ;  
153  
154   put num_1 = / num_2 = ;  
155  
156   if ( num_1 = w ) then put "num_1 is missing";  
157  
158 run;
NOTE: Variable w is uninitialized.
num_1=w
num_2=B
num_1 is missing
NOTE: There were 1 observations read from the data set WORK.MISSING.
NOTE: DATA statement used (Total process time):
   real time           0.00 seconds
   cpu time            0.01 seconds

160  
161 option missing = '.';  
     * reset the missing to be a period again ;
```

If you consider the first DATA step, the missing value of `num_1` is correctly identified. Note that testing for a missing value in `num_2` fails. If you recall, a missing value and a missing code, e.g. value .B assigned to `num_2`, are both valid values, which should not be equal. Therefore, our check has to include missing codes in order to identify them as missing values. Particular attention should also be shared with the value specified for the missing option (‘w’) as it is still interpreted as a variable and the string ‘w’ (not shown) is just a valid string of characters, which is consistent with previous examples, and cannot be used when checking for a missing value.

SAS provides several functions to test for missing values, of which we will focus on MISSING(), CMISS() and NMISS() functions. The latter of the three functions is reserved for numeric variables.
The **MISSING()** function will test for one variable at a time regardless if it is numeric or character.

```sas
data _null_
set work.missing;
put num_1 = / num_2 = ;
if missing( num_1 ) then put "num_1 is missing"
if missing( num_2 ) then put "num_2 is missing"
run;
```

num_1=.
num_2=B
num_1 is missing
num_2 is missing

NOTE: There were 1 observations read from the data set WORK.MISSING.
NOTE: DATA statement used (Total process time):
   real time           0.00 seconds
   cpu time            0.00 seconds

The **MISSING()** function also identifies missing codes as missing values, which greatly simplifies checking for any missing value.

The **NMISS()** function will return the number of missing values in the specified list of numeric variables, which is quite beneficial if you wish to ensure that at least one variable in the list is not missing. The **NMISS()** function will convert any character values to numeric before assessing if the argument value is missing.

The **CMISS()** function introduced in SAS 9.2 is similar to the **NMISS()** function that it counts the number arguments that are missing, but for both character and numeric variables without requiring character values to be converted to numeric.

Similar to the **MISSING()** function, the **NMISS()** and **CMISS()** functions identify missing codes as a missing value.

```sas
data _null_
set work.missing;
put num_1 = / num_2 = ;
count_missing = nmiss( num_1, num_2 );
put count_missing =
if ( nmiss( num_1, num_2 ) = 2 ) then put "num_1 and num_2 are missing"
run;
```

count_missing=2
num_1 and num_2 are missing

NOTE: There were 1 observations read from the data set WORK.MISSING.
NOTE: DATA statement used (Total process time):
   real time           0.01 seconds
   cpu time            0.00 seconds

Of course the condition **NMISS(…) = 0** is applicable if we wish to determine that all numeric variables have non-missing values. We could use the **N()** function as well given that it returns the number of non-missing values in a list of numeric variables.

There is a simple trick to check if multiple character variables are all missing prior to SAS 9.2 and the **CMISS()** function by emulating the **CMISS()** and **NMISS()** functions for character variables. Simply concatenate the character variables that are all missing and the result should be a single character variable with a missing value. We can then use the **MISSING()** function, added to SAS in version 7, to check for missing value as in the example below.
The MISSING(), CMISS(), NMISS() and N() functions provide a simple approach to check for missing values and giving the potential to avoid large if-statements when you need to check for missing values in several values at the same time.

‘OF’ MISSING VALUES
There are cases, whether good or bad programming practice, where you do end up with a rather cumbersome variables list, for example char1, char2, …, char100, and hard-coding each variable or using arrays and DO-loops are not exercises you look forward to.

The OF operator for variable lists is supported by CALL MISSING(), CMISS(), NMISS() and N() functions as well as the CATS() function, if you want to check for missing of several character variables with the trick mentioned above.

```sas
224  data _null_;
225
226    length char_1 - char_100 $ 10 ;
227    call missing( of char_1 - char_100, of num_1 - num_100 );
228
229    if missing( cats( of char_1 - char_100 )) and
230      ( n( of num_1 - num_100 ) = 0 )         then
231          put "all of them are missing";
232  run;
```

NOTE: DATA statement used (Total process time):
real time           0.00 seconds
cpu time            0.00 seconds

CONCLUSION
Missing values in SAS are valid values that may or may not represent special cases, which is dependent on the data source, SAS functions, statements or procedures and on your analysis. This makes missing values important to manage consistently in order to avoid unexpected results with little or no indication and very limited information.

The CALL MISSING statement provides an efficient and explicit method to initialize variables and assign a missing value. The functions MISSING, CMISS, NMISS and N provide the ability to identify, check for or simply count the number of missing values

The statements and function available in SAS provides simple, efficient and ample means to ensure that missing values are managed appropriately and any uninitialized variables become a key indicator of data issues.

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

Magnus Mengelbier
Limelogic Ltd
London, United Kingdom

E-mail: papers@limelogic.com
Web: www.limelogic.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.

Limelogic is a trademark of LimeLogic Limited, United Kingdom.

Other brand and product names are trademarks of their respective companies.
APPENDIX. SAS PROGRAM EXAMPLES

AccessType: Restricted

/*  --------------------------------------------------------------------
Limelogic Limited
London, United Kingdom

Program       phuse2011-cc05-missing_values.sas
Author        Magnus Mengelbier
Date          03mar2011
Version       1.0

--------------------------------------------------------------------
Description

--------------------------------------------------------------------
History
31mar2011   Version   1.0                          Magnus Mengelbier

-------------------------------------------------------------------- */

/*
Missing values
*/

data _null_ ;
  length char $ 10 ;
  num = . ;
  char = put( num, 8.-L);
  put num = / char = ;
  if missing(num) then put "num is missing";
  if missing(char) then put "char is missing";
run;

/* Missing system option ;
option missing = ' ';

data _null_ ;
  length char $ 10 ;
  num = . ;
  char = put( num, 8.-L);
  put num = / char = ;
  if missing(num) then put "num is missing";
  if missing(char) then put "char is missing";
run;

option missing = '.'; * reset the missing to be a period again ;
* Missing codes *

```sas
* data _null_;
   num = .B;
   if ( num = . ) then put "num is missing";
   if missing( num ) then put "missing() thinks num is missing";
run;
```

/*
Assigning a missing value
*/

```sas
* data _null_;
   num_3 = w;
   put w = / num_1 = / num_2 = / num_3 = ;
run;
```

* missing system option values and assigning missing values;

```sas
* data work.missing;
   num_1 = .;
   num_2 = .B;
run;
```

```sas
option missing = 'w';
```

```sas
* data _null_;
   set work.missing;
   put num_1 = / num_2 = ;
   num_3 = w;
run;
```

```sas
option missing = '.'; * reset the missing to be a period again ;
```

* call missing() statement *

```sas
* data _null_;
   num_3 = w;
   put num_1 = / num_2 = / num_3 = ;
run;
```

```sas
* data _null_;
   call missing( num_1, num_2, w );
   num_3 = w;
   put num_1 = / num_2 = / num_3 = ;
run;
```
/*
   Testing for missing values
*/
data _null_; set work.missing;
   put num_1 = / num_2 = ;
   if ( num_1 = . ) then put "num_1 is missing";
   if ( num_2 = . ) then put "num_2 is missing";
run;

option missing = 'w';
data _null_; set work.missing;
   put num_1 = / num_2 = ;
   if ( num_1 = w ) then put "num_1 is missing";
run;

option missing = '.'; * reset the missing to be a period again ;
data _null_;* the missing() function ;
   set work.missing;
   put num_1 = / num_2 = ;
   if missing( num_1 ) then put "num_1 is missing";
   if missing( num_2 ) then put "num_2 is missing";
run;

* the nmiss() function for number of missing ;
data _null_;* the n() function for number of non-missing ;
   set work.missing;
   put num_1 = / num_2 = ;
   count_missing = nmiss( num_1, num_2 );
   put count_missing = ;
   if ( nmiss( num_1, num_2 ) = 2 ) then put "num_1 and num_2 are missing";
run;

* the n() function for number of non-missing;
data _null_; set work.missing;
   put num_1 = / num_2 = ;
   count_nonmissing = n( num_1, num_2 );
   put count_nonmissing = ;
run;
* check for missing values in multiple character variables;

data _null_;
length char_1 - char_100 $ 10 ;
call missing( of char_1 - char_100, of num_1 - num_100 );
if missing( cats( of char_1 - char_100 ) ) and ( n( of num_1 - num_100 ) = 0 ) then
   put "all of them are missing";
run;

/*
   OF Missing values
*/

data _null_;
length char_1 - char_100 $ 10 ;
call missing( of char_1 - char_100, of num_1 - num_100 );
if missing( cats( of char_1 - char_100 ) ) and
   ( n( of num_1 - num_100 ) = 0 ) then
   put "all of them are missing";
run;