Working with Missing Values

Ed Heaton, Westat, Rockville, MD
Michelle Zhuang, Westat, Rockville, MD

ABSTRACT

SAS® 9.1 provides a call routine – **`Call missing()`** – that sets values to missing for a collection of variables. We will look at some methods to determine if those values are missing. SAS 9.1 also provides some new concatenation functions. We will use these to test collections of values to see if they are all missing.

SETTING TO MISSING

`Call missing()` will set values of the variables in its arguments to missing whether the variables are character or numeric. Programmers do not have to concern themselves with the data types.
Suppose, for example, we have the following variables.

```
                 Variables in Creation Order

                 #    Variable    Type    Len

                 1    a           Char      8
                 2    b           Char      8
                 3    c           Char      8
                 4    d           Num       8
                 5    e           Num       8
                 6    q1          Char      2
                 7    q2          Char      2
                 8    q3          Char      2
                 9    q4          Char      2
                10    q5          Char      2
                11    q6          Char      2
                12    q7          Char      2
                13    x1          Num       8
                14    x2          Num       8
                15    x3          Num       8
                16    x4          Num       8
                17    x5          Num       8
                18    y           Char      1
                19    z           Char      1
```

We can set **a**, **b**, **c**, **d**, and **e** to missing with comma delimited arguments.

```
   Call missing( a , b , c , d , e ) ;
```

If our list gets quite long, we might want to use the nice variable-list styles provided by SAS.

```
   Call missing( of a--e x1-x5 q: ) ;
```

The key word – **`of`** – tells SAS that SAS variable lists follow. Otherwise, you must use a comma-delimited list in the style introduced with **`Proc sql`**. We like the **`of`** keyword. You can even use an array name with the **`of`** keyword.

```
   Array foo[*] a--c ;
   Call missing( of foo[*] ) ;
```

TESTING FOR MISSING

TESTING ONE VARIABLE

SAS provides several ways to test if a single value is missing. We can, of course, simply compare the value with a missing value constant.

```
   Where ( ( c eq ' ' ) and ( d eq . ) ) ;
```

To use this method, we need to know whether each variable is character or numeric. We can eliminate that concern with the **`missing()`** function.

```
Where ( missing(c) and missing(d) ) ;
```

The **`missing()`** function works perfectly for character variables. Problems might arise with numeric variables because all missing numeric values are treated the same. We can't use the **`missing()`** function to return *true* only when the values are, e.g., **`.M`** or **`.S`**.

SAS borrowed the **`is null`** phrase from ANSI SQL.

```
Proc print ;  Where y is null ;  Run ;
```

This works for both character and numeric variables. However, the **`is null`** phrase is not valid in an **`If`** statement.

It might be worth noting that the **`is null`** phrase will return *false* if a character value contains the null character! The following **`Data`** step returns a table with one row.

```
Data _data_( where=( x is not null ) ) ;
    Length x $1 ;
    x = byte(0) ;
Run ;
```

To make things a little less confusing, SAS allows us to use the **`is missing`** phrase in **`Proc sql`**, in the **`Where`** statement, and in the **`where=()`** dataset option. It works just like **`is null`**, but doesn't imply that the value is the null character.

```
Select * from &sysLast where c is not missing ;
```

TESTING MULTIPLE VARIABLES

Sometimes we want to test to determine if all values from a list of variables are missing. If they are all numeric, we can sum them with the **`sum()`** function and then test if the returned value is missing.

```
If missing( sum( of x1-x5 ) ) then do ;
```

If they are all character, we can concatenate them and test if the returned value is missing.

```
If missing( catS( of q: ) ) then do ;
```

However, we would really like to not worry about whether they are character or numeric. Consider the following.

```
If (
    missing(  sum( of a-numeric-z   ) )
  & missing( catS( of a-character-z ) )
) then do ;
```

This would work for a set of variables that could otherwise be specified as **`a--z`**, however, it wouldn't work so well for variables that could only be specified as a numbered range list (e.g., **`x1-x5`**) or a name prefix list (e.g., **`q:`**).

Now, an interesting feature of the new concatenation functions – {**`cat()`**, **`catT()`**, **`catS()`**, **`catX()`**} – is that they convert numeric values to character strings with the **`best.`** format and then remove leading and trailing blanks from those values – automatically – with no notes in the log. We can use this. However, we must first tell SAS to display a missing value as a blank.

```
Options missing=' ' ;
Data _null_ ;
 /* Create and initialize some character variables. */
    Array abc[*] $8 a b c ( "a" "bee" "cee" ) ;
 /* Create and initialize some numeric variables. */
    Array de[*] d e ( 4 5 ) ;
 /* Create and initialize some more character variables. */
    Array q[7] $2 ( '1' '2' '3' '4' '5' '6' '7' ) ;
 /* Create and initialize some more numeric variables. */
    Array x[5] ( 1 2 3 4 5 ) ;
 /* Create and initialize even more character variables. */
    Array yz[2] $1 y z ( 'y' 'z' ) ;
 /* Now, set a few of these to missing and test. */
    Call missing( b , d , q1 , x2 ) ;
    If missing( catS( of a--e x[*] ) )
        then put "They're all missing." ;
        Else put "Some are not missing." ;
 /* Now set all of the tested variables to missing and retest. */
    Call missing( of a--e x[*] ) ;
    If missing( catS( of a--e x[*] ) )
        then put "Now they're all missing." ;
        Else put "Some are still not missing." ;
Run ;
```

The log will show the following two lines.

```
   Some are not missing.
   Now they're all missing.
```

A few notes are in order.

- Without the **`missing=' '`** option, a numeric missing value will be converted to **`'.'`** and that is not a character missing value.
- We used the **`catS()`** function because it strips leading and trailing blanks off the values before it concatenates them. This might save memory.
- This process will not treat special numeric missing values such as **`.A`**, **`.Z`**, and **`._`** as missing in the list of variables.
- This technique is applicable only in **`If`** statements. **`Proc sql`** syntax, the **`Where`** statement, and the **`where=()`** dataset option do not allow us to specify a SAS variable list by prefacing it with the **`of`** keyword.

CONCLUSION

The combination of the **`missing=' '`** system option and the **`missing( catS( of ... ) )`** construct in an **`If`** statement can make the task of testing a range of variables simpler to code and more robust. The code will also be easier to maintain. Of course, this requires SAS® 9 or later.

CONTACT INFORMATION

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

Ed Heaton
Westat
1650 Research Boulevard
Rockville, MD 20850-1395
Voice: (301) 610-4818
Fax: (301) 294-3879
Email: EdHeaton@Westat.com

Michelle Zhuang
Westat
1650 Research Boulevard
Rockville, MD 20850-1395
Voice: (301) 294-2002
Fax: (301) 294-3879
Email: MichelleZhuang@Westat.com