

Paper CC-191**Using CALL VNAME to Populate Missing Data from a
Default Values Lookup Dataset**

Raghav Adimulam, Westat, Rockville, MD

ABSTRACT

When SAS[®] is used to read ASCII data files, there is sometimes a need to recode missing data or to alter already existing data values. Often, the values to be substituted are in an Excel spreadsheet, which can be imported to SAS as a lookup table. This paper walks through the creation of a program that populates default values in a column when that column's information is empty in the input file. The relevant functions and routines used in this process are CALL VNAME, CMISS, Arrays, and DO loops. The goal is to output specific data patterns from the lookup datasets into the variables with missing values.

INTRODUCTION

When there are missing values present in an ASCII data file and there is a need to populate those missing values with specific default values, there is an easy way to accomplish this task. This paper walks through the creation of a program that populates the default values in data when that data is missing in the input file. The program doesn't overwrite any existing data. A CALL VNAME Routine is used to assign the Default Values Table's variable names as the values of a specified variable. The call routine is looped through a character array of variables present in the Default Values Table. As each variable is looped through a DO loop until the dimension of the Array is reached, a dynamic field is used to temporarily store the value of the column name, and another dynamic field is used to store the value of the variable. The CMISS function serves as a counter; representing the number of missing arguments in the range of the array. The goal of the program is to output specific data patterns from the datasets for the variables with missing values. The ultimate destination file is an ASCII file with all the populated default values from the lookup table.

READ ASCII DATA FILE INTO SAS

To illustrate the program, a sample ASCII data file is taken as shown below. This *column-delimited* ASCII data file is read into SAS using a data step.

The sample input file looks like this:

```
103 DOE^JOHN^A          19600101 MD22          4 NEW PLACE ROCKVILLE^MD^20850|||||||
103 SMITH^JANE^E        19501231 VA4455          56 SENECA ST^^VIENNA^VA^20210|||||||
103 JAMES^ROBERT^W      19400101 DC          3RD ST NW^^WASHINGTON^DC^20011|||||||
```

```
data Subjects;
  infile "C:\Data\Input\SubjectsFile.txt"
    filename=FNAME
    lrecl=100 missover;
  input
    RECORD $ 1 - 100
    Type $ 1 - 3
    Name $ 5 - 34
    BirthDate $ 36 - 43
    State $ 45 - 46
    Tst1 $ 47 - 48
    Tst2 $ 49 - 50
    Tst3 $ 51 - 52
    Tst4 $ 53 - 54
    Tst5 $ 55 - 56
    Tst6 $ 57 - 58
    Address $ 59 - 100
    ;
  Var_miss = cmiss(of tst1 -- tst6);
  N + 1;
  Lastname = scan(name,1);
  Firstname = scan(name,2);
  record_id = cats("%sysfunc(date()),date9.")', '|', "%sysfunc(time()),time.")
    , '|', put (n,4.));
run;
```

A unique Record ID containing the timestamp is assigned to each observation during this process. The CMISS function computes a variable containing the number of missing values.

In addition to reading the whole record into a variable, the program creates columns for the version type, name, date of birth, State, Test scores, the output file name and also reads in other columns for the process of recoding to default values. Three observations from dataset Subject are shown in the table below.

VIEWTABLE: Work.Subjects														
	Type	Name	BirthDate	State	Tst1	Tst2	Tst3	Tst4	Tst5	Tst6	Address	Var_miss	record_id	n
1	103	DOE^JOHN^A	19600101	MD	22						4 NEW PLACE ROCKVILLE^MD^20850	5	02MAY2016 13:58:15 0	
2	103	SMITH^JANE^E	19501231	VA	44	55					56 SENECA ST^^VIENNA^VA^20210	4	02MAY2016 13:58:15 1	
3	103	JAMES^ROBERT^W	19400101	DC							3RD ST NW^^WASHINGTON^DC^20011	6	02MAY2016 13:58:15 2	

OUTPUT 1: SAMPLE OF THE SUBJECTS DATA

LOAD THE LOOKUP FILE INTO SAS AND MERGE IT WITH SUBJECTS DATA

The Import procedure is used to read in the default values present in an excel file into a SAS dataset called *Default Values*.

```
PROC IMPORT DATAFILE="C:\Data\Input\defaults_v01.xls"
  OUT=DefaultValues DBMS=EXCEL REPLACE;
  GETNAMES=YES;
  SHEET="Default list";
RUN;
```

Sample observations from dataset DefaultValues are shown in the Output 2 table below.

VIEWTABLE: Work.Defaultvalues							
	State	TST1_47_2	TST2_49_2	TST3_51_2	TST4_53_2	TST5_55_2	TST6_57_2
1	DC	99	98	97	99	98	97
2	MD	90	90	91	91	99	99
3	VA	89	88	87	88	88	88

OUTPUT 2: THE DEFAULT VALUES DATA SET

Subsequently, the original subjects file is now merged to the Default Values file using the SQL procedure with the lookup variable (merging variable) being the two-byte State Abbreviation.

```
PROC SQL;
  CREATE TABLE All
  AS SELECT A.*, B.*
  FROM Subjects A LEFT JOIN DefaultValues B
  ON A.State=B.State;
QUIT;
```

CONSTRUCT AN ARRAY OF ELEMENTS

Two array statements are defined within a macro called Array1. Each array consists of a total of six elements that are to be recoded using the default values table. The elements represents the variables of Test scores 1 through 6. Of note is the VAR_DEFAULTS array that also contains the position and length of that particular variable (separated by underscores) in the ASCII record, whereas the array VAR_INPUT consists of the name of the variables only.

```
%MACRO ARRAY1;
ARRAY VAR_DEFAULTS
  TST1_47_2
  TST2_49_2
  TST3_51_2
  TST4_53_2
  TST5_55_2
  TST6_57_2;
ARRAY VAR_INPUT
  TST1 TST2 TST3 TST4 TST5 TST6
  ;
%MEND ARRAY1;
```

CALL ROUTINE TO ASSIGN MISSING VALUES

The SAS program below has another data step that changes the test score values to defaults only when the data is missing. For each iteration of the DO loop, the VNAME call routine assigns the value of the Default to a temporary variable called VALUE, which is then used to extract the column position and the length of the variable in the ASCII data file.

Another set of temporary variables called *Cinput* and *Cvalue* are used to hold the corresponding variable references for that particular DO loop iteration. The missing function is then used to check for the presence of data in the *Cinput* variable. If the expression *missing(Cinput)* returns a non-zero value, then the RECORD variable is substituted with the appropriate default value from the lookup table array reference.

```
DATA ALL1;
    Retain Record NewRecord;
    Length Cinput Cvalue Value $32;
    Set All;

%Array1;
    Do I=1 To Dim(Var_Input);

        Call Vname(Var_Defaults(I),Value);
        Cinput=Var_Input(I);
        Cvalue=Var_Defaults(I);

        start=input(scan(value,2,'_'),best.);
        len=input(scan(value,3,'_'),best.);

        If missing(Cinput) Then Do;
            newrecord=record;
            Var_Input(I)=Var_Defaults(I);
            substr(newrecord,start,len)=Var_Defaults(i);
        End;

        record=newrecord;

    End;

    final_miss = cmiss(of tst1 -- tst6); /*should be all 0's*/
Run;
```

Once the final data step is executed, the observations from final dataset ALL1 are shown in the table below.

VIEWTABLE: Work.ALL1														
	Type	Name	BirthDate	State	Tst1	Tst2	Tst3	Tst4	Tst5	Tst6	Address	Var_mis	record_id	n
1	103	JAMES^ROBERT^W	19400101	DC	99	98	97	99	98	97	3RD ST NW^^WASHINGTON^DC^20011	6	02MAY2016 13:58:15 2	3
2	103	DOE^JOHN^A	19600101	MD	22	90	91	91	99	99	4 NEW PLACE ROCKVILLE^MD^20850	5	02MAY2016 13:58:15 0	1
3	103	SMITH^JANE^E	19501231	VA	44	55	87	88	88	88	56 SENECA ST^^VIENNA^VA^20210	4	02MAY2016 13:58:15 1	2

OUTPUT 3: THE FINAL DATA SET WITH THE REPLACED TEST SCORES

USE DS2CSV MACRO TO EXPORT TO A TEXT FILE

Finally, we use the %DS2CSV macro to export the ALL1 data set to a comma-separated value file.

By default, the DS2CSV macro converts SAS data sets to comma-separated value (CSV) files. However, if you want to create some other type of output file (e.g., a tab-separated value file), then you should specify the hexadecimal code for the delimiter character within the parentheses.

```
%DS2CSV (data= ALL1, runmode=b, csvfile=C:\Data\Output\Final.csv);
```

CONCLUSION

This method is very useful for situations where there are a lot of variables, often hundreds or thousands of them, which need to be recoded in bulk with the appropriate default values stored in a SAS lookup table. During this process, we save a considerable amount of time by using arrays combined with new and powerful SAS functions like CMISS and MISSING. Using the CALL VNAME routine with dynamic array references, using the versatile PROC SQL tool for simple instances of merging data and timely usage of the SYSFUNC macros, all enhance the accuracy and presentation of the output.

REFERENCES

SAS Institute Inc., 2015. Base SAS® 9.4 Procedures Guide, Fifth Edition.

SAS Institute Inc., 2015. SAS® 9.4 Language Reference: Concepts, Fifth Edition

Cody, Ronald. 2010. SAS Functions by Example, 2nd edition. Cary, NC: SAS Institute Inc.

Carpenter, Art. 2012. Carpenter's Guide to Innovative SAS Techniques. Cary, NC: SAS Institute Inc.

ACKNOWLEDGEMENTS

The author would like to thank Michael Raithel for the suggestions and critiques which were greatly appreciated.

CONTACT INFORMATION

Name: Raghav Adimulam

Enterprise: Westat

Address: 1600 Research Blvd

Rockville, MD 20850

E-mail: RaghavAdimulam@westat.com



The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

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.