

New Data Step Functions in SAS® 6.12 - Data Set and Variable Functions Caroline Bahler, ASG Inc., Cary, NC

Abstract

The Data step in SAS® 6.12 is the recipient of 50+ new functions. These new functions are broken into the following groupings data set functions, external file functions, library and catalog functions, special functions, variable functions, and statistical functions. The objective of this paper is to discuss the data set and their related variable functions and to illustrate potential uses of the data set functions.

Introduction

The data set functions and their related variable functions come originally from the SCL (Screen Control Language) realm and have been available to SCL programmers for years. These functions can be utilized within a macro or Data step and provide the programmer with an additional set of options for accessing information stored about a data set or within a data set. Within a macro, a programmer will reference these functions by using the %sysfunc and/or %qsysfunc macros. Whereas, within a Data step the functions are called in a similar manner used within SCL code¹. Functionally, this group of functions can be divided into five (5) usage groups - required, physical attributes, observation identification, observation selection and variable value retrieval.

Usage Groupings

Required functions consist of the OPEN and CLOSE functions (see table 1 for descriptions). The OPEN function MUST be used to create a variable containing a return code or data set identification number before all of the rest of the functions can be used. Subsetting of the data set can occur by including the WHERE option within the OPEN function. In addition, the mode of opening a data set is settable within this function.

The CLOSE function is used to close an open data set unlocking it for further use. An opened data set MUST be closed within the same macro that it is opened or the data set becomes LOCKED. If a data set becomes locked then it is no longer available to other Data steps or procedures that modify the locked data set. This is probably one of the most frustrating situations a SCL programmer can face and now this frustration can be felt by the unwary or careless programmer when a data set is opened within a macro and NOT closed at the completion of the macro. Closing the data set within a Data step is not as critical since the data set(s) that are open are

automatically closed when a given Data step is completed / terminated.

Physical attribute functions consists of the EXIST, DSNAME, ATTRC, ATTRN, VARFMT, VARINFMT, VARLABEL, VARLEN, VARNAME and VARNUM functions (see table 1 for descriptions). The purpose of these functions is to describe different characteristics of the data set specified within the OPEN function and it's variables. This can eliminate the need to produce an output data set from the PROC CONTENTS procedure, since similar information about a data set is returned via the functions.

Observation identification functions consist of the CUROBS, NOTE, POINT, REWIND and DROPNOTE (see table 1 for descriptions). The purpose of these functions is to specify a location within the opened data set by observation number. They are used to locate a specific observation within the data set.

Observation selection functions consist of the FETCH and FETCHOBS functions (see table 1 for descriptions). These functions copy the variable values of a selected observation from the data set to a Data Step or macro variable. They differ in that FETCH can be used to copy all observations within a data set and FETCHOBS extracts a specified observation number. Both functions copy the values of all of the variables within an observation into the Data Set Data Vector (DDV)². The DDV contains only one observation's variable values at a time. When the next observation is accessed all values within the DDV are overwritten by the new observations variable values.

Variable value retrieval functions include the GETVARC and GETVARN functions (see table 1 for descriptions). These functions retrieve information from the opened data set into a variable defined within a Data step or into a macro variable. These can be useful for driving decision making points within a macro or subsetting another data set based on the contents of another data set or group of data sets.

The main hallmark of all of these functions is that they are used in conjunction with each other to reach a particular goal. As stated previously, before any of the other functions can be used the data set must first be opened, then functions from the other usage groups can be utilized as required.

Table 1. Listing of the data set and related variable functions grouped by function usage.

Grouping	Function	Description
Required	OPEN	Opens data set so that other functions can access information from and about the data set. A where option can be used to subset the data set. Access to the data set whether it is read-only or read-write is controlled by utilizing the mode modifier (see SAS HELP documentation for description).
	CLOSE	Closes the data set so that it can be used by other Data steps. If a data set is opened and not closed then the data set is locked and cannot be used by other Data steps or procedures.
Physical Attributes	ATTRC	Allows the programmer to determine the value of a character attribute (see Help Documentation for listing)
	ATTRN	Allows the programmer to determine the value of a numeric attribute (see Help Documentation for listing)
	DSNAME	When a data set is opened it is assigned a numeric identifier value, this associates the data set's name with the identifier.
	EXIST	Determines whether a data set designation actually exists as specified. This function does not require that the data set be opened. Therefore, it is a useful for test before attempting to open a data set.
	VARFMT	Returns the format associated with a variable.
	VARINFMT	Returns the informat associated with a variable.
	VARLABEL	Returns the label associated with a variable.
	VARLEN	Returns the length of a variable.
	VARNAME	Returns the name associated with a variable number.
	VARNUM	Retrieves the variable number for a specific variable name within an opened data set and is required for the retrieval of all other information about that variable.
Observation Identification	VARTYPE	Returns the type of a data set variable.
	CUROBS	Returns the current observation number of the open data set.
	NOTE	Marks an observation number within an open data set for further action.
	POINT	Utilizes the observation id returned by the NOTE function to return/locate that observation number.
	REWIND	Repositions the current observation to the observation number 1.
Observation Selection	DROPNOTE	Removes the marked placed by the NOTE function on an observation number.
	FETCH	FETCH in conjunction with the Call Set Routine ^{1,3} can copy all observations within an opened data set into the Data Set Vector one observation at a time. The variable values from within each observation can then be passed to a data step or macro.
Variable Value Retrieval	FETCHOBS	FETCHOBS is similar to FETCH except retrieves a specific observation.
	GETVARC	Utilizes the variable number to copy the value of a character variable from the DDV into a macro or Data step variable.
	GETVARN	Utilizes the variable number to copy the value of a numeric variable from the DDV into a macro or Data step variable..

Documentation for each of the functions in Table 1 can be found within the HELP area of SAS 6.12, including syntax specifications. To access this information a HELP user needs to click on the HELP icon within the toolbar and select one of the following areas from the main HELP menu - What's New and Base SAS. A listing of all of the new functions occurs within the What's New section under Base SAS - Functions (Fig. 1). Or the information about the functions can be accessed within the complete listing of all Base SAS functions under Base SAS Documentation - Language Reference, SAS

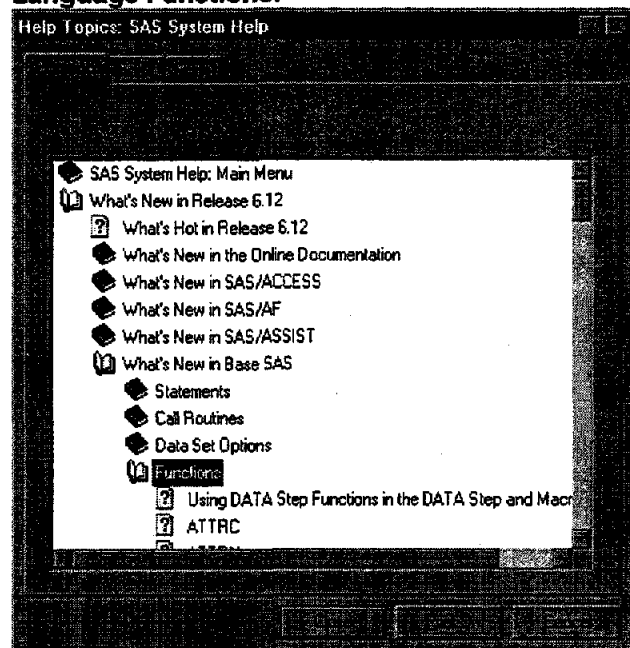
Functions (Fig. 2). Information is also available on SAS's web page about What's New in 6.12 (see reference 3).

Utilization of functions

Typically, the following steps should occur within a data step or macro for data retrieval to occur.

Illustration of steps utilizing the %sysfunc functions within a macro:

Fig 1. SAS Help screen indicating What's New - Language Functions.



```
%macro test;
```

Step 1 - Data set must be opened by using the OPEN function.

```
%let opened=%sysfunc(open(sashelp.company,i));
```

Step 2 - Call SET routine, this allows information from the data set to be copied into the DDV.

```
call set(&opened);
```

Step 3 - Copy the variable values of the next available observation into the DDV. In this case, using the %DO %UNTIL loop allow the macro to cycle through all observations within the opened data set. Typically programmers have used the variable name of RC for variables that contain what are considered to "return code" values. Any variable name could be used.

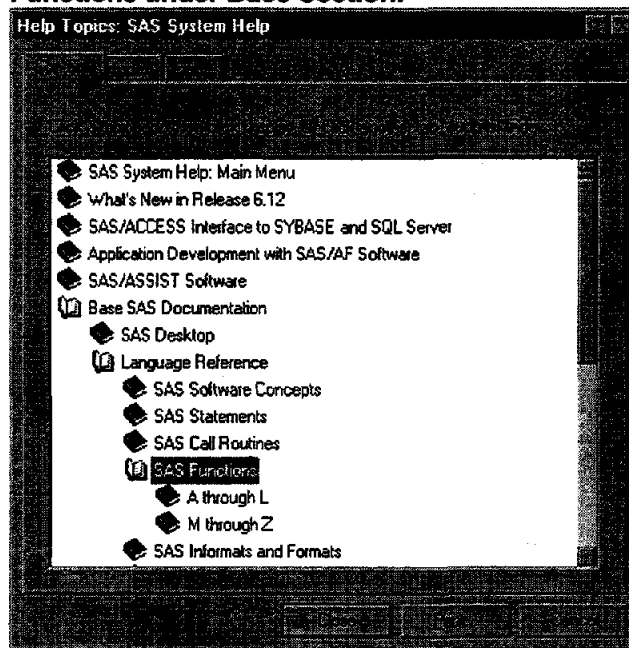
```
%do %until(&rc = -1);
  %let rc = %sysfunc(fetch(&opened));
```

Step 4 - Determine the number of the variable of interest within the data set and get the value of that variable from the DDV.

```
%let vnum=%sysfunc(varnum(&opened,LEVEL5));
%let emp=%sysfunc(getvarc(&opened,&vnum));
%put &emp;
%end;
%end;
```

Step 5 - Close opened data set unlocking the data

Fig 2. SAS Help screen indicating Language Functions under Base Section.



set and allowing use of this data set by other Data steps or procedures.

```
%let closed=%sysfunc(close(&opened));
```

```
%mend test;
```

```
%test;
```

Illustration of steps using a Data step:

```
data test;
  set sashelp.company;
  test="sasuser.england";
```

Step 1 -

```
opened=open(test,"i");
```

Step 2 -

```
call set(opened);
do until(rc=-1);
```

Step 3 -

```
rc=fetch(opened);
```

Step 4 -

```
vnum=varnum(opened,"employee");
employee=getvarc(opened,vnum);
if level5=employee then output;
end;
end;
run;
```

Please note that when using these functions within a Data Step, step 5 is no longer necessary because the opened data set is closed automatically.

However, including step 5 could be beneficial as a reminder especially if the data step is ever migrated into a SCL application.

Further Examples of Function Uses

Example 1 - Illustrates problem associated with opening a data set using the OPEN function and attempting to manipulate that data set while the data set still open. The scenario below requires information about the data set within the macro before manipulations to that data set can occur.

In this macro a macro variable *test* is create containing the data set name, which is then used to determine whether the data set exists or not.

```
%macro test1;
%let test=sashelp.company;
%if %sysfunc(exist(&test)) %then %do;
```

If the data set exists then the data set is opened and the number of variables within the data set are determined.

```
%let opened=%sysfunc(open(sashelp.company,i));
%let vars=%sysfunc(attrn(&opened,nvars));
```

Once the number of variables is determined then manipulation of data set can occur. As the macro processes the do loop the variable name is determined and the data set is sorted by that variable. At the end of the do loop the data set is closed and the macro completed.

```
%do i = 1 %to &vars;

%let sort=%sysfunc(varname(&opened,&i));

proc sort data=sashelp.company;
by &sort;
run;
/* etc... */

%end;

%let closed=%sysfunc(close(&opened));
%end;

%mend test1;
%test1;
```

The problem with the macro above is that the following error message will appear in the Log window every time the data set needs to be sorted.

ERROR: MEMBER lock is not available for SASHELP.COMPANY.DATA, lock held by conflicting use.

The conflicting use is the OPEN function at the beginning of the macro. The next two (2) macros are illustrations of how to get around the above problem.

Example 2 - Illustrate one solution to locking problem where a new data set SORT is created before the original data set is opened.

```
%macro test2;

data sort;
set sashelp.company;
run;

/* Set macro variable test to data set name. */
%let test=sashelp.company;

/* Determine whether data set exists. */
%if %sysfunc(exist(&test)) %then %do;

/* Open data set */
%let opened=%sysfunc(open(sashelp.company,i));

/* Determine number of variables */
%let vars=%sysfunc(attrn(&opened,nvars));

/* Perform manipulation of data set. */
%do i= 1 %to &vars;
%let s=%sysfunc(varname(&opened,&i));

proc sort data=sort;
by &s;
run;

/* etc... */

%end;
%let closed=%sysfunc(close(&opened));
%end;

%mend test2;
%test2;
```

Example 3 - Illustrates a second solution to the locking problem by using the OUT = option within the SORT procedure.

```
%macro test3;

/* Set macro variable test to data set name. */
%let test=sashelp.company;

/* Determine whether data set exists. */
%if %sysfunc(exist(&test)) %then %do;

/* Open data set */
%let opened=%sysfunc(open(sashelp.company,i));
```

```

/* Determine number of variables */
%let vars=%sysfunc(attrn(&opened,nvars));
/* Perform manipulation of data set. */
%do i= 1 %to &vars;
  %let s=%sysfunc(varname(&opened,&i));

  proc sort data=sashelp.company out=&s;
    by &s;
  run;

/* etc...*/

%end;
%let closed=%sysfunc(close(&opened));
%end;

%mend test3;
%test3;

```

SAS is a registered trademark or trademark of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

Caroline Bahler
 2000 Regency Parkway
 Suite 355
 Cary, NC 25711
 919-467-0505
 ccb@asg-inc.com

In both examples, the original data set was not the data set manipulated. This inability to manipulate the a data set while that data set is opened using the OPEN function can be a major factor in the design of a given macro.

This is not a factor within a Data Step as illustrated below -

```

data sasuser.test;
set sasuser.test;
test="sasuser.test";

opened=open(test,"i");
if opened > 0 then nobobs=attrn(opened,"NOBS");
rc=close(opened);

run;

```

The data set created contains both the original variables and opened, nobobs, and rc. The point if the data set example above is that the data set being manipulated and opened are the same data set, i.e. the problem of locking does not exist in this case.

In conclusion, these functions put a additional set of valuable tools in the hands of SAS programmers.

References

1. SAS Institute Inc., SAS Screen Control Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 91-374 pp.
2. SAS Institute Inc., SAS Screen Control Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 37 - 44 & 91pp.
3. "What's New in Release 6.12.", SAS Institute - Technical Support web page, <http://www.sas.com/service/techsup/wn612/wnps.html>.