Using SAS Functions in Data Steps

Yue Ye, The R.W. Johnson Pharmaceutical Research Institute, Raritan, NJ
Yong Lin, The Cancer Institute of New Jersey, New Brunswick, NJ

ABSTRACT

SAS base software has plenty of built-in functions. Proper use of these functions can save a lot of programming time and effort. In this paper, we will illustrate how to use some of new functions in version 6.12 and later. These new functions include data set functions, variable functions, external file functions, library and catalog functions, and some other special functions.

We will concentrate on using these functions in data steps. Quite often, we need to get the information from SAS data sets, libraries or external files in data steps. The conventional way of getting the information is that we first run several SAS procedures, then merge the results with the SAS data set. Using SAS functions directly in data step provides a simple solution in many situations.

INTRODUCTION

There are many new SAS data step functions introduced since version 6.12. These new functions include data set functions, variable functions, external file functions, library and catalog functions, and some other special functions. In this paper, we want to show how to use these functions through several specific examples. We try to use as many functions as possible so the reader will be able to see the use of as many of these functions. We hope this paper can serve as a tutorial for the use of SAS data step functions.

Although all the functions discussed in this paper can be used both in data step and in macro facility, we will demonstrate the use of these functions only in SAS data step. The uses of these functions in macro facility are similar to the uses in data step. The main differences will be discussed in FINAL NOTES section.

The data set generated by the following SAS codes will be used for demonstration through out this paper:

```
proc format;
  value $ gender 'F'='Female' 'M'='Male';

data income(label='Annual Income');
   input name &$20. street &$20. income +1 gender $1.;
   format income dollar11.2 gender gender.;
   label name='Last Name, First Name'
         street='Address'
         income='Annual Income'
         gender='Gender';
   cards;
Leverling, Janet      55 Hazel Way        54789  F
Peacock, Margaret     101 Broadway         4565  F
Smith, John           23 Mars Hill Dr.    86685  M
Buchanan, Steven      45 Gray Drive       23567  M
Suyama, Michael       213 Hillside Ave.   65778  M
King, Robert          345 Main St.        45654  M
Callahan, Laura       455 8th St.        134656  F
Dodsworth, Anne       57 Pleasant Blvd.    5433  F
Davolio, Nancy        65 Peanut Circle    57654  F
;
```

Most of data set access functions use data set identifier obtained from the OPEN function. The syntax of the OPEN function is

open('data-set-name', 'mode')

where mode may be 'i', meaning read but not modified, or 'in', meaning read sequentially and allowed to revisit an observation, or 'is', meaning observations are read sequentially but not allowed to revisit an observation. The OPEN function opens a SAS data set and returns a unique numeric data set identifier. For example, we may use the following data step to find the number of observations, the number of variables and the label assigned to the data set in the data set *income*

Example 1:

```
data ex1;
  dsid=open('income','i');
  n_obs=attrn(dsid,'nobs');
  n_vars=attrn(dsid,'nvars');
  dslabel=attrc(dsid,'label');
run;
proc print;run;
```

Here we open SAS data set *income* in read only mode. The return value of the OPEN function is assigned to the variable *dsid*. This value is used in the ATTRN function to get the number of observations and the number of variables in the data set, and also used in the ATTRC function to obtain the label assigned to the data set. The following is the output from the program:

OBS	DSID	N_OBS	N_VARS	DSLABEL
1	1	9	4	Annual Income

All data sets opened within a DATA step are closed automatically at the end of the DATA step. Using the CLOSE function, you can close any opened SAS data sets as soon as they are no longer needed. The syntax of the CLOSE function is

close(data-set-id)

where data-set-id is the data set identifier of the corresponding data set. You may assign the return value of the CLOSE function to any variable in a data step. As an example, you may add the statement

```
rc=close(dsid);
```

to the data step in Example 1 as the last statement of the data step.

Similar to the OPEN and CLOSE functions for SAS data sets, the FOPEN and FCLOSE functions are used for external files and the DOPEN and DCLOSE functions are used for directories. The arguments of both FOPEN and DOPEN use the file reference assigned to a file or a directory. The FILENAME function can be used to assign or deassign a file reference for an external file, directory, or output device. The FILEREF function is used to verify if a file reference has been assigned. A file can also be opened by directory ID and member name using the MOPEN function. We will show the use of these functions along the way.

USING DATA SET FUNCTIONS

As we discussed before, the OPEN and the CLOSE functions are the data set functions. Other useful new data set functions introduced since version 6.12 are ATTRC, ATTRN, EXIST, DSNAME, FETCH, FETCHOBS, CUROBS, NOTE, DROPNOTE, POINT, and REWIND. The ATTRC and ATTRN functions are used to retrieve the value of a character and numeric attributes of a SAS data set, respectively. The EXIST function is used to verify the existence of a SAS data library member. The DSNAME function is

used to find the data set name associated with a data set identifier. The FETCH and FETCHOBS functions are used to read observations from a SAS data set. The CUROBS, NOTE and DROPNOTE functions are used to find the current observation number, to create an observation ID (note marker) for the current observation, and to delete a note marker created by a NOTE function, respectively. The REWIND function will move the data set pointer to the beginning of a SAS data set.

We have used the ATTRN function to find the number of observations and the number of variables in a data set and the ATTRC function to find the label assigned to a data set in Example 1. Other values of attributes for a specified SAS data set can be retrieved by specifying the corresponding attribute names in the second argument of the ATTRN or ATTRC functions. The following example demonstrates the use of some other data set functions. In this example, we try to find observation numbers, values and variable names of the longest value of character variables in the data set *income,* and to find the observation numbers of the first and the last of such observations and assign them to macro variables *firstobs* and *lastobs*.

Example 2:

```
data ex2;
  dsid=open('income','i');
  n=attrn(dsid,'nvars');

  *>> Find the length of the longest variables;
  maxlen=0;
  do while(fetch(dsid)=0);
    do i=1 to n;
      if (vartype(dsid,i)='C') then do;
        value=getvarc(dsid,i);
        newlen=length(left(trim(value)));
        if newlen>maxlen then do;
          maxlen=newlen;
        end;
      end;
    end;
  end;

  *>> Find the obs. number, var. name and var.
  value of the longest length;
  first=.;
  rc=rewind(dsid);
  do while(fetch(dsid)=0);
    do i=1 to n;
      if (vartype(dsid,i)='C') then do;
        value=getvarc(dsid,i);
        newlen=length(left(trim(value)));
        if newlen=maxlen then do;
           varname=varname(dsid,i);
           obsnum=curobs(dsid);
           if first=. then first=note(dsid);
           output;
        end;
      end;
    end;
  end;

  *>> Find the obs. number of first and last
  obs. with longest length;
  rc=point(dsid,first);
  rc=fetch(dsid);
  rc=dropnote(dsid, note); * not needed;
  call symput('firstobs',put(curobs(dsid),3.));
  rc=fetchobs(dsid,obsnum);
  call symput('lastobs',put(curobs(dsid),3.));
  keep obsnum varname value maxlen;
run;

proc print;run;

%put _users_;
```

The following is the output from the above data step:

OBS	MAXLEN	VALUE	VARNAME	OBSNUM
1	17	Peacock, Margaret	NAME	2
2	17	213 Hillside Ave.	STREET	5
3	17	57 Pleasant Blvd.	STREET	8

and in the log file, we see that

```
GLOBAL FIRSTOBS   2
GLOBAL LASTOBS    8
```

From the output, we see that the maximum length of the values of the character variables is 17. There are three of such values, and they are in the observation numbers 2, 5, and 8. The values and the corresponding variables are also listed in the output.

The data set functions used in the program have been highlighted in boldface. Some variable functions are also used. In the program, first we open the data set and find the number of variables in the data set. The length of the longest variables is found in the second block (or paragraph) of the program. In this block, the program reads observations from the data set through the FETCH function. The return code of the FETCH function is 0 if it reads the observation successfully, so the program exits the do loop when the end of the data set is reached. We also used the VARTYPE function to find the type of variables and the GETVARC function to find the value of the character variables. We will see more of the variable functions in the next section.

In the third block, the variable's values of the longest length are found along with the corresponding observation numbers and the variable names. The REWIND function move the pointer to the beginning of the data set so that the following FETCH function will read the data again from the first observation. If a variable's value in an observation has the longest length, the corresponding observation number will be obtained through the CUROBS function. The NOTE function is used to mark, i.e. to assign an observation ID to, the first of such observation. This marker will be used in the fourth block with the POINT function to return the pointer to this observation.

In the fourth block, the FETCHOBS function is used to return the pointer directly to the observation specified by the observation number. This is different from the POINT function, which uses the note marker created by the NOTE function. The last value of the variable *obsnum*, assigned from the CUROBS function in third block, is the observation number of the last observation having a variable with a value of the longest length. Since we try to use as many different functions as possible for demonstration purpose, we used the NOTE function in third block and used the POINT function to place the pointer to the corresponding observation in the fourth block. We can use the CUROBS function and the FETCHOBS function instead.

USING VARIABLE FUNCTIONS

The variable functions introduced since version 6.12 include GETVARC, GETVARN, VARFMT, VARTYPE, VARINFMT, VARLABEL, VARLEN, VARNAME, and VARNUM. The GETVARC and GETVARN functions are used to find the value of a character variable and the value of a numeric variable, respectively. The VARTYPE function is used to find the type of a variable (numeric or character type). The VARFMT, VARINFMT, and VARLABEL functions are used to find the format, informat and label assigned to a data set variable. The length of a variable can be found using the VARLEN function. We can find the number of a variable's position in the SAS data set using the VARNUM function. If we know the number of a variable's position in the data set, we can use the VARNAME function to find the corresponding variable name.

We have used variable functions GETVARC, VARTYPE, and VARNAME in Example 2. To demonstrate the use of some other variable functions, we consider the task of generating a listing table for the data set *income*. If a variable has a format, the value with the corresponding format will be used instead of the 'raw' value. The variable labels will be used for the corresponding column headings. For this purpose, we need to find the numbers of columns(spaces) needed for each variable. In Example 3, we will find such numbers. To be more specific:

- For character variables, we need to find the larger value of the length of the variable's label and the maximum length of the formated character values over all observations for each character variable.
- For numeric variables, we need to find the larger value of the length of the variable's label and the variable's width for each numeric variable. Here we assume that all numeric variables have formats.

Example 3:

```
data ex3;
  array lens len1-len8;
  dsid=open('income','i');
  n=attrn(dsid,'nvars');

  *>> Find the lengths of variable labels;
  do i=1 to n;
    lens(i)=length(left(trim(varlabel(dsid,i)
                                          )));
  end;

  *>> Compare with the maximum lengths of the
  values of each character variable;
  do while (fetch(dsid)=0);
    do i=1 to n;
      if (vartype(dsid,i)='C') then
        lens(i)= max(length(
           putc(getvarc(dsid,i),varfmt(dsid,i))
                                    ),lens(i));
    end;
  end;

  *>> Get the widths of the formats for each
  numeric variable;
  do i=1 to n;
    if (vartype(dsid,i)='N') then do;
      format=varfmt(dsid,i);
      numpos=indexc(format,'123456789');
      fmtwidth=input(substr(format,numpos,
                    indexc(format,'.')-1),2.);
    end;
  end;
  keep len1-len8;
run;

proc print;run;
```

The following is the output from the above program

```
 OBS  LEN1  LEN2  LEN3  LEN4  LEN5  LEN6  LEN7  LEN8

  1    21    17    13    6      .     .      .     .
```

From the output, we see that the numbers of columns needed for the table are 21, 17, 13 and 6 for the four variables. Notice that for simplicity we have assigned more variables than needed for lengths (i.e., len1 to len8,) in our example. It is not difficulty to modify the program so that the number of the variables needed for lengths will be calculated directory in SAS program.

The variable functions used in the program have been highlighted in boldface. In the second block of the program, the lengths of the variable labels for each variable are found. The variable function VARLABEL is used to get the label for each variable.

In the third block of the program, the lengths of variable labels are compared with the lengths of the values of each variable, the maximum of them are the numbers of the columns needed for the table. If the variable has an assigned format, the corresponding values with format will be used. The VARTYPE function is used to determine the types of variables. The GETVARC function is used to get the values of character variables, and the VARFMT function is used to get the format values of variables.

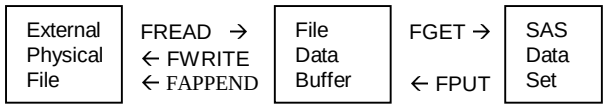
For numeric variables, the lengths of the variable labels are compared with the widths of the formats for the numeric variables in the last block. The maximum of the two numbers will be the number of columns needed for the table.

The SAS code in this example will be used in the next example to complete our task of generating listing tables.

USING EXTERNAL FILE FUNCTIONS

Many external file functions are available now. These include functions for directories and for files. The names of functions for directories begin with a D, like in DCLOSE and DNUM. The names of functions for files begin with an F, like in FGET and FEXIST. Many data set functions, like EXIST, NOTE, POINT, REWIND and so on, have the corresponding external file functions.

As we discussed before, the DOPEN, FOPEN, DCLOSE and FCLOSE functions are used to open and close directories and files. The DNUM and DREAD functions are used to find the number of members in a directory and the name of a directory member, respectively. The FEXIST and FILEEXIST functions are used to verify the existence of an external file, the former uses an associated file reference and the latter uses the physical name of the file. The FNOTE, FPOINT and FREWIND functions have their counterparts in data set functions. We can use the FDELETE function to delete an external file, and the FRLEN function to find the size of the last record read. It is helpful to understand that the communication between SAS data sets and external files are through so called File Data Buffer (FDB). The following graph illustrates this communication:



The FGET and FPUT are used to copy data from the File Data Buffer (FDB) to a SAS data variable and to move data to FDB, respectively. The FREAD and FWRITE are used to read a record from an external file into the FDB and write a record from the FDB to an external file, respectively. The FAPPEND can also be used to append the current record to an external file. The FCOL and FPOS functions are used to find the current position in the FDB and to position the column pointer in the FDB, respectively.

The following is an example of using external file functions. In this example, we try to write the listing tables of the data set *income* to different files by gender. The following is the goals we want to achieve:

- The table is in the center of the page;
- Leave 3 spaces between adjacent variable fields;
- Use variable labels for the column headings;
- Center the title and the column headings;
- If a variable has a format, use the format in the output;

Example 4:

```
%let path=c:\PharmaSug\;

data ex4;
  array lens len1-len10;
  array pos pos1-pos10;
  ls=76;

  *>> Find the column position for each var.;
  dsid=open('income','i');
  n=attrn(dsid,'nvars');
  do i=1 to n;
     lens(i)=length(left(trim(varlabel(dsid,i)
                                          )));
  end;
  do while (fetch(dsid)=0);
    do i=1 to n;
      if (vartype(dsid,i)='C') then
         lens(i)=max(length(
           putc(getvarc(dsid,i),varfmt(dsid,i))
                                    ),lens(i));
    end;
  end;
  do i=1 to n;
    if (vartype(dsid,i)='N') then do;
      format=varfmt(dsid,i);
      numpos=indexc(format,'123456789');
      fmtwidth=input(substr(format,numpos,
                     indexc(format,'.')-1),2.);
    end;
  end;
  sumlen=0;
  do i=1 to n; sumlen=sumlen+lens(i); end;
  pos1=floor((ls-sumlen-3*(n-1))/2);
  do i=2 to n;
     pos(i)=pos(i-1)+lens(i-1)+3;
  end;

  *>> Open files to write;
  rc=filename('male',"&path\male.txt");
  if rc=0 and fexist('male') then
     rc=fdelete('male');
  rc=filename('female',"&path\female.txt");
  if rc=0 and fexist('female') then
     rc=fdelete('female');
  fid1=fopen('male','a',,'p');
  fid2=fopen('female','a',,'p');

  *>> Write titles to files;
  ttl1='Annual Income for Male';
  ttl2='Annual Income for Female';
  rc=fpos(fid1, floor((ls-length(ttl1))/2));
  rc=fpos(fid2, floor((ls-length(ttl2))/2));
  rc=fput(fid1,ttl1); rc=fput(fid2,ttl2);
  rc=fwrite(fid1,'-'); rc=fwrite(fid2,'-');

  *>> Write variable labels to files;
  do i=1 to n;
    labpos=pos(i)+floor((lens(i)-
           length(left(varlabel(dsid,i))))/2);
    rc=fpos(fid1,labpos);rc=fpos(fid2,labpos);
    rc=fput(fid1, varlabel(dsid,i));
    rc=fput(fid2, varlabel(dsid,i));
  end;
  rc=fwrite(fid1,'0'); rc=fwrite(fid2,'0');
  do i=1 to n;
    rc=fpos(fid1,pos(i));rc=fpos(fid2,pos(i));
    rc=fput(fid1, repeat('-',lens(i)-1));
    rc=fput(fid2, repeat('-',lens(i)-1));
  end;
  rc=fappend(fid1); rc=fappend(fid2);

  *>> Write records to files;
  rc=rewind(dsid);
  gennum=varnum(dsid,'gender');
  do while (fetch(dsid)=0);
    if getvarc(dsid,gennum)='M' then fid=fid1;
    else fid=fid2;
    do i=1 to n;
      rc=fpos(fid,pos(i));
      if (vartype(dsid,i)='C') then
        rc=fput(fid,putc(getvarc(dsid,i),
                         varfmt(dsid,i)));
      else rc=fput(fid,putn(getvarn(dsid,i),
                         varfmt(dsid,i)));
    end;
    rc=fwrite(fid);
  end;
 run;
```

The above program generates two external files named 'male.txt' and 'female.txt' in the directory c:\ParmaSug. The file 'male.txt' contains the table for male:

Annual Income for Male

Last Name, First Name	Address	Annual Income	Gender
Smith, John	23 Mars Hill Dr.	$86,685.00	Male
Buchanan, Steven	45 Gray Drive	$23,567.00	Male
Suyama, Michael	213 Hillside Ave.	$65,778.00	Male
King, Robert	345 Main St.	$45,654.00	Male

and the file 'female.txt' contains the table for female:

Annual Income for Female

Last Name, First Name	Address	Annual Income	Gender
Leverling, Janet	55 Hazel Way	$54,789.00	Female
Peacock, Margaret	101 Broadway	$4,565.00	Female
Callahan, Laura	455 8th St.	$134,656.00	Female
Dodsworth, Anne	57 Pleasant Blvd.	$5,433.00	Female
Davolio, Nancy	65 Peanut Circle	$57,654.00	Female

The external file functions used in the program have been highlighted in boldface. The line size is set to 76, and this value is assigned to the variable *ls*. The second block of the program has been discussed in Example 3, except that in the last part we used the numbers of columns (or spaces) needed for each variable to calculate the starting columns (positions) for each variable's field in the external file.

The third block of the program opens the external files named 'male.txt' and 'female.txt'. The files will be deleted if the physical files exist. We used the FILENAME function to assign file references to the corresponding external files. The FEXIST function has been used to find the existence of the files, and the FDELETE function has been used to delete the files if they exist. Notice that we can also use the MOPEN function to open an external file. To use MOPEN, first we need to use the DOPEN function to open the directory where the file is located, then use the directory identifier, which is the return value of DOPEN, along with the physical file name for the arguments of the MOPEN function.

In the fourth block, the titles, placed in the center of the line, are written to the external files. The FPOS function is used to move the pointer to the correct position so that the title will be in the center of the line. After that the FPUT function is used to move the titles to the corresponding FDBs. Finally the FWRITE function is used to write the record in the FDBs to the corresponding physical external files.

Similarly, in the next block, the variable labels are written to the external files. The variable labels are used as the column headers of the corresponding variables in the table. Each label is written to the center of the variable's column using the number of spaces needed, which was calculated in the second block. We used the FAPPEND function instead of the FWRITE function in this block just to show the alternatives.

The variable values are written to the external files in the last block. Notice that the REWIND function is needed in order to move the pointer to the beginning of the data set and to prepare to read the data from the first observation. Since we want to write the records to different external files according to gender, the variable number of the gender variable is determined first using the VARNUM function.

The VARFMT function is used so that the corresponding formats will be used in the final output. Since the VARFMT function will return a blank string if no format has been assigned to the variable, the PUTC and PUTN functions will return the correct values regardless of whether the variable has a format or not.

USING LIBRARY AND CATALOG FUNCTIONS

The CEXIST, LIBNAME and LIBREF functions can be used in data step to verify the existence of a SAS catalog or SAS catalog entry, to assign or deassign a library reference for a SAS data library, and to verify that a library reference has been assigned, respectively. The following is an example of using these functions.

Example 5:

```
data test;
  rc1=cexist('sashelp');
  rc2=cexist('sashelp.devices');
  rc3=cexist('sashelp.devices.cgmmwwc.dev');
  rc4=libref('mylib');
  if rc4^=0 then
     libn=libname('mylib','g:\paper');
  rc5=libref('mylib');
  name=pathname('mylib');
run;
proc print;run;
```

The following is the output from the program:

```
OBS  RC1  RC2  RC3   RC4   LIBN  RC5  NAME

 1    0    1    1   70006    0    0  g:\paper
```

Since 'SASHELP' is a SAS library, not a catalog or catalog entry, and there is no SAS catalog or SAS catalog entry named 'SASHELP', the value of RC1 is 0, meaning non-existence. On the other hand both RC2 and RC3 have values of 1, meaning that the catalog 'SASHELP.DEVICES' exists, and so does the catalog entry 'SASHELP.DEVICES.CGMMWWC.DEV'. In this program, the library function LIBREF is used to verify that the library reference 'MYLIB' has been assigned. The return code RC4 of LIBREF is not zero (70006), that means the operation was not successful, ie, 'MYLIB' has not been assigned. So we assign the reference to the SAS data library in 'G:\PAPER'. The external directory name for the library reference 'MYLIB' is found using the external file function PATHNAME.

USING SOME OTHER FUNCTIONS

There are some other functions introduced since version 6.12. For example, you can use the GETOPTION function to find SAS system or graphics options in a data step, use the SYSTEM function to issue an operating system command, etc. Furthermore, new statistical functions have been introduced in Release 6.12. These statistical functions include the CDF, PDF (alias PMF), LOGPDF (alias LOGPMF), SDF and LOGSDF functions for many distributions. The CDF function computes the cumulative distribution function. The PDF function computers the probability density or mass function. The LOGPDF function computes the logarithm of the probability density or mass function. The SDF and LOGSDF functions compute the survival function and logarithm of the survival function, respectively. We will not give examples on how to use these functions because it should pose no difficulties. The following section will show you how to find the references for the functions discussed in this paper.

USING ONLINE DOCUMENTATION

SAS on-line documentation is a very good source of references for the functions discussed in this paper. You can find the detailed information about a specific function from the SAS on-line help. In order to find the complete list of SAS data step functions when you run SAS 6.12 in interactive mode, first start from the SAS system help window, then choose "Base SAS Documentation", next select "Language Reference", and then "SAS functions". To find the information on data step functions since version 6.12, start from SAS system help window, choose "What's New in Release 6.12", select "What's New in Base SAS", and finally choose "Functions".

FINAL NOTES

In this paper, we have concentrated on using the data step functions in data step only. The macro functions %SYSFUNC and %QSYSFUNC can be used to call all these functions. For example, you can use the following code to find the number of variables in the data set *income*:

```
%let dsid=%sysfunc(open(income,i));
%let n=%sysfunc(attrn(&dsid,nvar));
%let rc=%sysfunc(close(&dsid));
```

The number of observations in the data set will be assigned to the macro variable *n*. If you submit the command

```
%put _users_;
```

after the macro assignment statements above, you will see the following output in the SAS log window (or the log file depending on whether you are running interactive mode or batch mode):

```
GLOBAL DSID 1
GLOBAL N 4
GLOBAL RC 0
```

So the macro variable *n* has a value of 4, which is the number of variables in the data set *income*. Because %SYSFUNC is a macro function, we do not need to enclose character values in quotation marks as we do in data step. See reference [2] for more information on the macro functions %SYSFUNC and %QSYSFUNC.

REFERENCES

1. SAS Language: Reference, V.6, first edition.
2. SAS Macro Language: Reference, First Edition (1997).
3. SAS on-line documentation in Release 6.12.

CONTACT INFORMATION

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

Yue Ye
The R.W. Johnson Pharmaceutical Research Institute
Raritan, NJ
Phone: (908) 704-4909 Fax: (908) 526-2567
Email: yye@prius.jnj.edu

Yong Lin, Ph.D.
The Cancer Institute of New Jersey
195 Little Albany Street
New Brunswick, NJ 08901
Phone: (732) 235-9621 Fax: (732) 235-8096
Email: linyo@umdnj.edu