ABSTRACT

%SYSFUNC was originally developed in SAS version 6.12 to allow the incorporation of SCL (SAS Component Language, formerly Screen Control Language) functions into the SAS macro programming environment. Among its many capabilities, %SYSFUNC can determine the existence of a SAS data set, count the number of its observations, and characterize the attributes of the data set’s variables. Using %SYSFUNC in a SAS macro can make the macro more flexible and general by allowing conditional execution of code based on the observed run-time characteristics of data sets and variables. %SYSFUNC can be used to conditionally execute SAS code in many types of macros, including those for data manipulation, statistical calculations, and report generation. A general data manipulation macro can use %SYSFUNC to automatically determine if the data’s “key” variable is character or numeric and then execute the appropriate version of code appropriate for the data type. A general statistical macro that generates an output dataset from a chi-square test can use %SYSFUNC make sure the output data set was created without error before proceeding with next analytic step. A general report macro can use %SYSFUNC to identify each report variable’s format to create a new text variable of each report variable’s formatted value, and can identify each report variable’s SAS label to identify the variable in the table.

INTRODUCTION

The SAS macro facility, a component of Base SAS, is one of the most powerful tools in the SAS system. Effective use of SAS macros can make SAS programs more general and reusable, shorter, less prone to error, and easier to understand. Several references are available that discuss the basics of SAS macro programming, and this paper assumes a basic knowledge of SAS macros.

%SYSFUNC was originally developed in SAS version 6.12 to allow the incorporation of SCL (SAS Component Language, formerly Screen Control Language) functions into the SAS macro programming environment. Among its many capabilities, %SYSFUNC functions can determine the existence of a SAS data set, count the number of observations in the SAS data set, and characterize the attributes of the data set’s variables.

Using %SYSFUNC in SAS macros can make the macros more flexible and general by allowing conditional execution of code based on the observed run-time characteristics of SAS data sets and variables. This paper gives simple examples of how to incorporate %SYSFUNC in SAS macros to conditionally execute SAS code based on characteristics of SAS datasets or SAS variables.

The three types of macros included in this paper as examples are:

- A general data manipulation macro that uses %SYSFUNC to automatically determine if a variable in a SAS data set is character or numeric, and then executes the appropriate version of code appropriate for that variable’s data type;
- A general statistical macro that generates an output dataset from a chi-square test, and then uses %SYSFUNC make sure the output data set was created without error before proceeding with next analytic step; and
- A general report macro that uses %SYSFUNC to identify each report variable’s SAS label format, and then creates a new text variable corresponding to the report variable’s formatted value.

BRING IN ‘DA %SYSFUNC (OR, “%SYSFUNC 101”)

Even though %SYSFUNC is a very useful tool that allows programmers to make their macros more powerful and general, %SYSFUNC has been described as a “mysterious, misunderstood macro function.” Other published papers give a much more detailed and complete description of %SYSFUNC and how to use it, but here are brief descriptions of the %SYSFUNC syntax needed for the examples in this paper.

The examples of SAS programs highlight the code specific to %SYSFUNC in bold text.
USING %SYSFUNC TO DETERMINE IF A SAS DATA SET EXISTS
To determine if a SAS data exists, the macro code using %SYSFUNC syntax in a variable assignment is as follows:

```sas
%let dsexist = %sysfunc(exist(&sasdsname));
```

where:
- `&sasdsname` = a macro variable assigned the name of the SAS data set
- `exist` = the SCL function used to determine if the data set exists
- `&dsexist` = a macro “flag” variable, where the value = 1 if the data set exists, and = 0 if the data set does not exist

The function can also be used directly as:

```sas
%if %sysfunc(exist(&sasdsname)) = 0 %then %do;
/*----------------------------------------------------------------*/
/*  Insert macro code that needs to be executed if the data set */
/*  &sasdsname does not exist.*/
/*----------------------------------------------------------------*/
%end;
```

USING %SYSFUNC TO DETERMINE THE NUMBER OF OBSERVATIONS OF A SAS DATA SET
To determine the number of observations in a SAS data set using %SYSFUNC, the macro code can be set up as:

```sas
%let dsid = %sysfunc(open((&sasdsname));
%let numobs = %sysfunc(attrn(&dsid, nobs));
%let rc = %sysfunc(close(&dsid));
```

where:
- `&sasdsname` = a macro variable assigned the name of the SAS data set to be opened
- `&dsid` = a macro variable with the “Data Set ID” of the opened data set
- `attrn` = a macro variable assigned to the name of a variable in the SAS data set `&sasdsname`
- `nobs` = a macro variable with the “Variable Number” of the variable in the dataset
- `&numobs` = a macro variable with the number of observations of the data set `&sasdsname`
- `&rc` = a macro “flag” variable that gives the “return code” for successfully closing the data set

USING %SYSFUNC TO DETERMINE CHARACTERISTICS OF A VARIABLE
The macro %SYSFUNC code to determine a characteristic of an existing variable in an existing SAS data set can be set up as:
DATA MANIPULATION MACRO THAT DETERMINES IF VARIABLE IS CHARACTER OR NUMERIC

I do many analyses on health-related data sets where patients enrolled in the study are identified by a "patient ID" variable. The patient ID variable is very important, and I don’t like to modify it from the way it is originally received in study data. However, I often do very similar analyses with data from different studies, and this important patient ID variable can be received as either character or numeric.

The simple macro given in the example %DELVAR shows how to determine the type of a variable using %SYSFUNC, and the macro will conditionally generate code based on the variable’s type (character or numeric).

The macro %DELVAR will delete all observations in the data set &DSN where the value of the variable &VARNAME is either missing (for a numeric variable) or blank (for a character variable).

```sas
%macro delvar(dsn=, varname=);

    /*-----------------------------*/
    /*  Get the type (Character or Numeric) of the variable  */
    /*-----------------------------*/
    %let dsid   = %sysfunc(open(&dsn));
    %let varnum = %sysfunc(varnum(&dsid, &varname));
    %let vartyp = %sysfunc(vartype(&dsid, &varnum));  /*  variable type  */
    %let rc     = %sysfunc(close(&dsid));

    /*----------------------------------------------------------------*/
    /*  Create a data set with no missing values of the variable */
    /*----------------------------------------------------------------*/
    data nomiss&dsn;
    set &dsn;
    %if &vartyp = C %then %do;
       if &varname = '' then delete;
    %end;
    %else %if &vartyp = N %then %do;
       if &varname <= .z then delete;
    %end;
    run;

%mend delvar;
```
STATISTICS MACRO THAT LOOKS AT CHI-SQ TEST OUTPUT

The input data for a routine statistical test, such as a chi-square test, sometimes might not be appropriate for that test due to missing values, small cell sizes or insufficient numbers of categories. If the input data are not appropriate for the statistical test, then no output data set with the test’s summary statistics will be created.

If the output data set for a statistical test is not created at run time, then the next DATA step that uses the output data set can not run. An even worse situation could occur if the next DATA step ends up using a prior version of the statistical output data set created with the prior input data.

The following macro uses %SYSFUNC to make sure the output chi-square data set generated by PROC FREQ was actually created before continuing to the next data step. If an output data set was not created, then the macro creates a data set with the relevant p-value variable set to missing.

```
%macro chisq(varname=);
    proc freq data=dsn nolimit;
        tables txgroup * &varname / chisq;
        output chisq out=chisq&varname;
    run;

    /*-----------------------------------------------*/
    /* Determine if chi-sq test was successful and output*/
    /* data set exists. If not, create an output data set*/
    /* with a missing p-value.*/
    /*-----------------------------------------------*/
    %if %sysfunc(exist(chisq&varname)) = 0 %then %do;
        data chisq&varname;
        p_pchi = .z;
        run;
    %end;

    /*-----------------------------------------------*/
    /* Create a data set with a character version of the chi-square*/
    /* for printing*/
    /*-----------------------------------------------*/
    data chisq&varname._prt (keep = pval_prt);
        set chisq&varname;
        length pval_prt $ 50;
        if p_pchi > .z then pval_prt = "p-value = "||
            trim(left(put(pc_pchi,pvalfmt.)));
        else if p_pchi = .z then pval_prt = "Invalid chi-square";
    %mend chisq;
```

Note that the name of the output data set (chisq&varname) will have a different data set name for each variable used as input. This naming convention further helps prevent an output data set created from prior analysis of a different variable to be used, if the current analysis data set was not created due to an error.
REPORT MACRO THAT DETERMINES A VARIABLE’S FORMAT AND LABEL

A common type of report generated with SAS gives the summary univariate statistics for several variables found in a SAS data set. If the variables included in the table are categorical variables with attached formats, then those variables cannot be included in the table as a single column, because the original variables all have different attached formats. A new character variable -- with a value corresponding to the text version of the categorical variable’s formatted values – will need to be created for the table.

The following example uses a SAS data set named PATVARS. Each variable in PATVARS is a numeric variable with an attached SAS format and a descriptive label. The data set PATVARS has 335 observations, and the value of the variable TXGROUP is always equivalent to the formatted value of “Drug A.”

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>agecat</td>
<td>Num</td>
<td>8</td>
<td>AGEFMT.</td>
<td>Age categorized</td>
</tr>
<tr>
<td>claim_source</td>
<td>Num</td>
<td>8</td>
<td>INSFMT.</td>
<td>Health insurance type</td>
</tr>
<tr>
<td>gender</td>
<td>Num</td>
<td>8</td>
<td>GENFMT.</td>
<td>Gender</td>
</tr>
<tr>
<td>regionn</td>
<td>Num</td>
<td>8</td>
<td>REGFMT.</td>
<td>Region</td>
</tr>
<tr>
<td>txgroup</td>
<td>Num</td>
<td>8</td>
<td>TXFMT.</td>
<td>Treatment Group</td>
</tr>
</tbody>
</table>

The macro %CATSTATS generates the summary statistics “n” and “%n” for the categorical variables included in the table. By using %SYSFUNC in the macro %CATSTATS, the only information needed to be passed to the macro are the name of the data set variable (&ROWVAR) and the variable’s row number (&ROWNUM) in the report. %SYSFUNC will allow the macro %CATSTATS to automatically determine both the name of the variable’s attached format, and text of the variable’s descriptive label for use in the report.

The table produced by the SAS program that calls %CATSTATS is shown below. All components of the table that were generated with %SYSFUNC are shown in bold:

<table>
<thead>
<tr>
<th>Patient Characteristic</th>
<th>Drug A n = 335</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age categorized (n, %)</td>
<td></td>
</tr>
<tr>
<td>18-24</td>
<td>15 (4.5%)</td>
</tr>
<tr>
<td>25-34</td>
<td>61 (18.2%)</td>
</tr>
<tr>
<td>35-44</td>
<td>110 (32.8%)</td>
</tr>
<tr>
<td>45-54</td>
<td>94 (28.1%)</td>
</tr>
<tr>
<td>55-64</td>
<td>49 (14.6%)</td>
</tr>
<tr>
<td>65+</td>
<td>6 (1.8%)</td>
</tr>
<tr>
<td>Gender (n, %)</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>156 (46.6%)</td>
</tr>
<tr>
<td>Female</td>
<td>179 (53.4%)</td>
</tr>
<tr>
<td>Geographic region (n, %)</td>
<td></td>
</tr>
<tr>
<td>Northeast</td>
<td>37 (11.0%)</td>
</tr>
<tr>
<td>North Central</td>
<td>102 (30.4%)</td>
</tr>
<tr>
<td>South</td>
<td>134 (40.0%)</td>
</tr>
<tr>
<td>West</td>
<td>62 (18.5%)</td>
</tr>
<tr>
<td>Health insurance type (n, %)</td>
<td></td>
</tr>
<tr>
<td>Commercial</td>
<td>326 (97.3%)</td>
</tr>
<tr>
<td>Medicare</td>
<td>9 (2.7%)</td>
</tr>
</tbody>
</table>

Table: Baseline Characteristics of Patients on Drug A

The CONTENTS Procedure
Alphabetic List of Variables and Attributes
The `%CATSTATS` macro is given below:

```sas
/*******************************/
/*  Summary statistics for categorical variables (n, %)  
/*******************************/
%macro catstats(rownum=, rowvar=);
   proc freq data=patvars noprint;
      tables &rowvar / out=&rowvar.stats0;
   run;
   data &rowvar.stats (keep = &rowvar count percent n);
      set &rowvar.stats0;
      n + 1;
   run;
   /*---------------------------*/
   /*  Get format and label associated with categorical variable  */
   /*---------------------------*/
   %let dsid = %sysfunc(open(&rowvar.stats,i));
   %let varnum = %sysfunc(varnum(&dsid,&rowvar));
   %let &rowvar.fmt = %sysfunc(varfmt(&dsid, &varnum));   /*  format  */
   %let &rowvar.lbl = %sysfunc(varlabel(&dsid, &varnum)); /*  label   */
   %let rc = %sysfunc(close(&dsid));
   /*-----------------------------------*/
   /*  Create row for printing  */
   /*-----------------------------------*/
   data row&rownum.prt (keep = patchar stats);
      set &rowvar.stats;
      label num = 'Row Number'
      patchar = "Patient Characteristic"
      stats = "Patient Statistics"
      ;
      length patchar stats $ 50;
   /*-----------------------------*/
   /*  Output label for variable  */
   /*-----------------------------*/
   if n = 1 then do;
      patchar = "&&&rowvar.lbl"||" (n, %)";
      stats = ",";
      output;
   end;
   /*-----------------------------------*/
   /*  Formatted value of the variable  */
   /*-----------------------------------*/
   patchar = 'A0A0A0A0'x||trim(left(put(&rowvar,&&&rowvar.fmt)));
   stats = trim(left(put(count,8.)))||'|' ||
          trim(left(put(percent,8.1)))||'%' ;
   output;
   run;
%mend catstats;
```
The program that calls the macro `%CATSTATS` and then generates the table is given below:

```
/**************************************************
/* Call %CATSTATS for each variable in the table */
/**************************************************/
%catstats (rownum=1, rowvar=agecat);
%catstats (rownum=2, rowvar=gender);
%catstats (rownum=3, rowvar=regionn);
%catstats (rownum=4, rowvar=claim_source);

/**************************************************
/* Put summary statistics for each variable together */
/**************************************************/
data allprt;
  set row1prt row2prt row3prt row4prt;
run;

/***********************************************************************/
/* Use %SYSFUNC in open code to determine the number of observations */
/* in the data set PATVARS for the table header */
***********************************************************************/
%let dsid = %sysfunc(open(patvars));
%let numobs = %sysfunc(attrn(&dsid, nobs));
%let rc = %sysfunc(close(&dsid));
%put &numobs;

/***********************************************************************/
/* Print out the report */
***********************************************************************/
proc report data=allprt nowindows headline headskip split='*' missing;
  column patchar stats;
  define patchar / group "Patient Characteristic"
      order=data style(column)=[cellwidth=3.4IN just=left];
  define stats / display "Drug A*n = &numobs"
      style(column)=[cellwidth=1.8IN just=center];
run;
```

The program in the example can be run on any formatted variables in a data set named PATVARS. The macro and program can be further generalized by setting the name of the input data set PATVARS as a macro variable, by adding additional columns for additional patient groups, and even by adding chi-square statistics to see if any significant differences exist across drug treatment groups.
CONCLUSION: WRITE THAT %SYSFUNC-Y MACRO
The macro examples included in this paper illustrate just a few of the many possible ways to incorporate %SYSFUNC into SAS macros. %SYSFUNC can be used in other different types of macros with other applications. %SYSFUNC also has other functions that can determine additional characteristics of SAS variables and data sets that will allow the macro to conditionally generate SAS code based on these characteristics. These examples of macros using %SYSFUNC should be viewed as starting points, and as inspiration for idea of how to start using %SYSFUNC in your own macros.

REFERENCES:

ACKNOWLEDGMENTS
SAS is a Registered Trademark of the SAS Institute, Inc. of Cary, North Carolina.

CONTACT INFORMATION
Please contact the author with any comments, questions, or additional puns on the word “%SYSFUNC”:
Kathy H. Fraeman
United BioSource Corporation
7101 Wisconsin Avenue, Suite 600
Bethesda, MD  20832
(240) 235-2525 voice
(301) 654-9864 fax
kathy.fraeman@unitedbiosource.com