SESUG Paper 113-2020

UFO Search
(User-Friendly-Optional-Search)

Dr. Kannan Deivasigamani, Centene - TSS

Abstract

This SESUG paper demonstrates how a SAS® macro can be used to easily search for column names in a Database-Table or a SAS variable name in a Library of datasets, without having to spell the exact name. Carpenter (2004), Mannan and Stevenson (2010), Chen and Yang (2012) have shown through their work of how they have uses SAS for varied applications in geospatial and mathematical analysis. Similarly, this search feature was also developed using the power of macros within SAS to help a user identify variables using partial names.

The User-Friendly-Optional-Search tool henceforth will be referred as UFOs goes one step further and can also help a SAS user find out even the SAS dataset name in a library or a Table name within a Database referenced by a libname statement, just by providing a partial name. Often SAS users may work with multiple variables from SAS datasets in a library and will find a need to locate all the tables that has a particular column. Normally, they would have to refer the documentation or search for the text in programs or datasets to locate the variations based on occurrence in the code. As an alternative, some programmers end up reaching out to team members and senior members who are familiar with the libraries or databases hoping that they can tap into their wisdom to get an answer.

Gone are those days and UFOs can help sooner than the time it takes to formulate a help memo. This paper presents the code with examples presented in simple fashion and one may find it very easy to understand. UFOs uses SAS meta-data as one can imagine via Dictionary feature and is explained in depth within the sections below.

INTRODUCTION

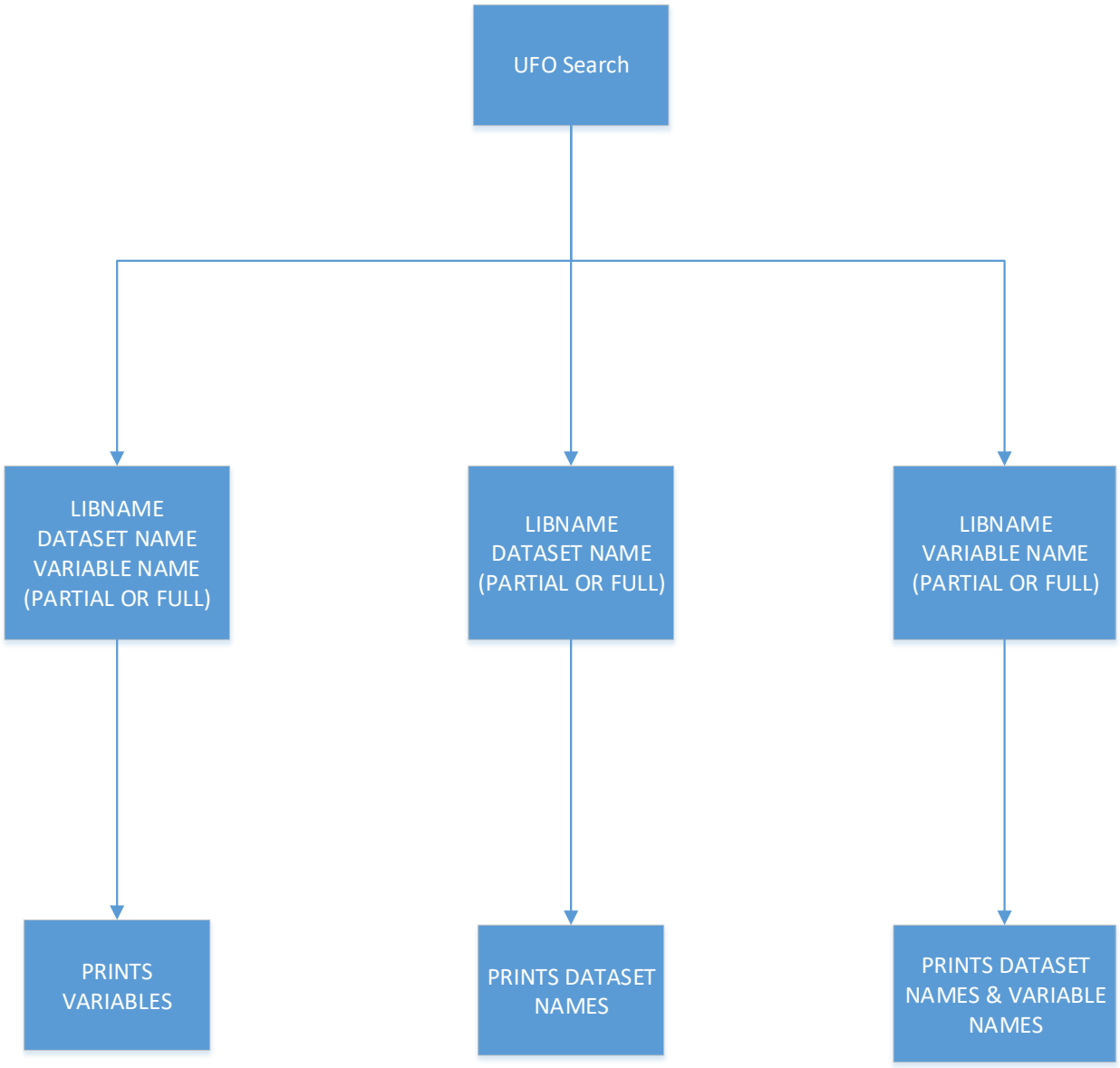
Development of new programs often result in multiple searches for variables and columns from various data sources from various libraries and databases. Time is a major factor while searching for dataset names that contain the targeted variables. Early on in a project, during the design phases, iterations of searches and refinements can happen. Having the ability to search quickly can save several hundreds of man-hours in a project that can potentially translate to cost-savings.

Repetitive coding can be potentially replaced by a SAS macro. Breheny and Novikov (2003) showed how they have taken advantage of the power of SAS macros in the field of genetic research and statistical simulations. Following this path, the task of searching through a series of columns or a series of datasets was automated through this macro, UFOs. This macro operation, design, and application have all been presented in the sections below with visual and code illustrations to help the reader use it effectively.

Any user with SAS Base Programming proficiency can utilize this macro as the call is simple with one line code and a few parameters. All the functionality is abstracted from the user and hidden via an auto-call library that does the crunching and filtering of the meta-data to present the end user with the results that can be readily used. While this may work for most requirements, there is room for improvement and it can be enhanced to include more flexibility but will proportionately also increase the complexity of not only the macro code but also to the call as there may be more parameters required.

The programmer gains more depending on the usage and directly correlates with the frequency of use. The macro can also be concurrently used by an entire team of developers placing a call with different parameters. One may find that there is less typing involved and less work involved and therefore, can help save time and also can mitigate one from fatigue due to developing code or manual perusal of meta-data to find the same.

A PICTORIAL REPRESENTATION OF THE PROCESS



MACRO OPERATION

UFOs has been designed to accept three parameters and is capable of looking for patterns in either dataset pattern within a library or variable pattern within a SAS dataset or a library. The macro first reads the variables provided by the caller and then decides which path to take. As shown in the pictorial representation above, there are 3 different pathways the code execution can flow through, depending on the parms provided and the missed values. The macro is built with the assumption that if a user calls the macro with DSN and VAR populated, then the DSN is considered to be a complete name and not a partial name.

The LIB parameter is always required without which the macro is not designed to work. The VAR parameter can be a partial value or a complete value that the macro can accept and retrieve the table names or the list of variable names matching the pattern of characters provided in the parameter. The source code below and the 4 examples worked out below with the specific macro calls and their corresponding results should be clear to a new user and will help utilizing the macro in his or her own local environment. The one step in preparation is to have the macro code housed in the AUTOCALL library or have it added to the SASRC so the maclib is effective and the macro can be called without predefining it within each program. The macro also takes advantage of the meta-data available through Dictionary tables and PROC DATASETS. The data is crunched to customize the type of search and produce the desired results for the user driven by the parameters.

Thus the macro provides a user friendly optional search feature for SAS developers.

SOURCE CODE

```
%MACRO ufos(LIB=,DSN=,VAR=);
     %IF &VAR=
     %THEN
          %DO;
               options nomlogic nomprint nosymbolgen;
               data _null_;
                    tbl1='"'||'%'||%sysfunc(trim(%sysfunc(upcase("&DSN"))))||'%'||'"';
                    call symput('tbl',tbl1);
               run;

               ods output datasets.members=Members;
               proc datasets lib=&lib;
               quit;

               proc sql;
                    create table temp_&sysjobid. as
                    select name
                      from members
                     where upcase(name) like &tbl;
               quit;
               ods listing;

               proc print data=temp_&sysjobid.;
                    title1 "=====================================";
                    title2 "RESULTS_OF_MACRO_CALL_ %NRSTR(%%)ST:";
                    TITLE3 "Tables like:&tbl. in library:&lib.";
                    title4 "=====================================";
               run;
               title ' ';
               options mlogic mprint symbolgen;
          %END;
     %ELSE
     %IF &DSN=
     %THEN
          %DO;
               options nomlogic nomprint nosymbolgen; %put &lib &var;
               data _null_;
                    var1='"'||'%'||%sysfunc(trim(%sysfunc(upcase("&VAR"))))||'%'||'"';
                    call symput('VAR',VAR1);
               run;

               ods output datasets.members=Members;
               ods output Datasets.DataSet.Variables = Variables;

               proc datasets lib=&lib nolist nodetails;
                    CONTENTS DATA=_ALL_;
```

```
47                quit;
48
49                proc sql;
50                     create table temp_&sysjobid. as
51                     select *
52                       from variables
53                      where upcase(variable) like &var;
54                quit;
55
56                ods listing;options linesize=200;
57                proc print data=temp_&sysjobid.;
58                     title1 "==============================================";
59                     title2 "RESULTS_OF_MACRO_CALL_ %NRSTR(%%)UFOs:";
60                     TITLE3 "Columns like:&var in Database/SAS Library:&lib.";
61                     title4 "==============================================";
62                run;
63                title ' ';
64                options mlogic mprint symbolgen;
65           %END;
66      %ELSE
67           %DO;
68                options validvarname=any nomlogic nomprint nosymbolgen;
69                data DSN;
70                  set &LIB..&DSN (OBS=0);
71                run;
72                %put 'col:' &col;
73
74                data _null_;
75                     col1='"'||'%'||%sysfunc(trim(%sysfunc(upcase("&VAR"))))||'%'||'"';
76                     put 'col1:' col1;
77                     call symput('col',col1);
78                run;
79
80                proc sql;
81                  create table _&sysjobid as
82                  select UPCASE(name)  AS NAME
83                    from dictionary.columns
84                   where libname='WORK'
85                     and memname='DSN'
86                     and upcase(name) like &col;
87                quit;
88
89                ods listing;
90                proc print data=_&sysjobid;
91                  title1 "==============================================";
92                     title2 "RESULTS_OF_MACRO_CALL_ %NRSTR(%%)UFOs:";
93                     TITLE3 "Columns like:&col in SAS Dataset/TABLE:&DSN";
94                     title4 "==============================================";
95                RUN;
96                title ' ';
97                options mlogic mprint symbolgen;
98           %END;
99
100  %mend ufos;
```

PRACTICAL APPLICATION

%UFOs (LIB=sashelp, DSN=baseball, VAR=rbi)

The macro uses 3 parameters to perform the search of which the first parameter is required and the remaining 2 are optional but atleast one of them should be assigned a value; atleast a partial value.

For demonstration purposes, SASHELP Tables are used. Below are the results

```
1  ================================================
2  RESULTS_OF_MACRO_CALL_ %UFOs:
3  Columns like:%RBI% in SAS Dataset/TABLE:baseball
4  ================================================
5
6  Obs     NAME
7
8   1      NRBI
9   2      CRRBI
10
```

%UFOs (LIB=sashelp, DSN=, VAR=rbi)

This call above was made without the SAS dataset name (only LIB & VAR were provided) and here is the result

```
1  ================================================
2  RESULTS_OF_MACRO_CALL_ %UFOs:
3  Columns like:%RBI% in Database/SAS Library:sashelp
4  ================================================
5
6  Obs        Member          Num    Variable    Type      Len    Pos    Format       Label          Informat    Flags
7
8   1     SASHELP.BASEBALL     14     CrRbi       Num         8     88              Career RBIs
9   2     SASHELP.BASEBALL      7     nRBI        Num         8     32              RBIs in 1986
10
```

In the next test, the call is made to the macro without the variable name however, just partial name of the dataset is provided to check the ability of the macro to find the related dataset names; and the results are below

%UFOs(LIB=sashelp, DSN=ball, VAR=)

```
1  ========================================
2  RESULTS_OF_MACRO_CALL_ %ST:
3  Tables like:%BALL% in library:sashelp
4  ========================================
5
6  Obs       Name
7
8   1      BASEBALL
```

This final test illustrates the ability of UFOs to find all datasets that have the partial name provided in the call as shown below

%UFOs(LIB=sashelp, DSN=geo, VAR=)

The results show that there were 4 tables in the SASHELP library that contained "geo" as part of their names.

```
1  ========================================
2  RESULTS_OF_MACRO_CALL_ %ST:
3  Tables like:%GEO% in library:sashelp
4  ========================================
5
6  Obs      Name
7
8   1      GEOEXM
9   2      GEOEXM
10  3      GEOEXP
11  4      GEOEXS
```

LIMITATION

The macro, in its current design takes up one parameter and searches for it in that call. In other words, the macro is capable of searching for either dataset names or variable names in a single call. The macro cannot look for both at the same time and account for the permutations and combinations of dataset names and variable names. This could be a potential enhancement if required by the user in a future version.

CONCLUSION

The macro can be further enhanced for comparing 3 or 4 or even multiple datasets at once if necessary just by cloning the code presented based on a user's requirement. The more frequently this process is used, the more benefit the team reaps as each time a comparison is created, the organization benefits directly saving time as one does not have to re-invent the wheel and developers have the results in a matter of seconds without the labor. This is just another example that shows how DICTIONARY tables can be harnessed to its maximum benefit and when used in conjunction with macros, programmers become enabled to develop powerful code.

REFERENCES

Breheny, P. Writing cleaner and more powerful SAS code using macros. Retrieved from https://myweb.uiowa.edu/pbreheny/misc/macros.pdf. Retrieved on 07/31/2020.

Carpenter, Art. 2004. Carpenter's Complete Guide to the SAS® Macro Language, Second Edition. Cary, NC: SAS Institute Inc.

Chen, V. Y.-J., & Yang, T.-C. (2012). SAS macro programs for geographically weighted generalized linear modeling with spatial point data: applications to health research. *Computer Methods and Programs in Biomedicine*, *107*(2), 262–273. https://doi.org/10.1016/j.cmpb.2011.10.006

Mannan, H., & Stevenson, C. (2010). SAS macros for point and interval estimation of area under the receiver operating characteristic curve for non-proportional and proportional hazards Weibull models. Journal of Evaluation in Clinical Practice, 16(4), 756–770. https://doi.org/10.1111/j.1365-2753.2009.01190.x

Novikov, I. (2003). A remark on efficient simulations in SAS. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *52*(1), 83–86. https://doi.org/10.1111/1467-9884.00343

RECOMMENDED READING

- Base SAS® Procedures Guide
- SAS® Macro Language: Reference Guide®
- SAS® Macro Programming

CONTACT INFORMATION

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

Dr. Kannan Deivasigamani

Centene Corporation

TEXTdrk@gmail.com

https://www.linkedin.com/in/kannandeivasigamani