PharmaSUG 2015 - Paper QT10

A Simple Macro to Select Various Variables Lists

Ting Sa, Cincinnati Children's Hospital, Cincinnati, OH

Yanhong Liu, Cincinnati Children's Hospital, Cincinnati, OH

ABSTRACT

Often we need to select a subset of variables from a dataset. SAS® software has provided the "SAS variable list" from which we can select the variables without typing their names individually. However, the "SAS variable list" cannot be used in all SAS procedures, such as the SELECT statement in the SQL procedure. Also, there is no "SAS variable list" available to select variables that share a common suffix or middle part in their names. In this paper, we introduce a macro that not only incorporates the "SAS variable list" to select a subset of variables, but also can be used to select variables that share common patterns in their names. Additionally, the results from the macro can be applied to all SAS procedures.

INTRODUCTION

A SAS variable list is an abbreviated method of referring to a list of variable names [1]. There are four types of SAS variable list: a numbered range list, a name range list, a name prefix list and a special SAS name list.

The numbered range list refers to a list of variables having the same name, except for the last character or characters which are consecutive numbers. In Table 1 data set, variable list a1, a2, a3 is a numbered range list, and you could use "a1-a3" in SAS to select this list.

a1	b1	a2	b2	a3	b3	a4	b4
a	1	b	2	c	3	d	4

Table 1. A Sample Data Set

The name range list is a list that includes the variables based on the order of the variables in the data set. For example, in SAS, "a1--b2" refers to a name range list a1, b1, a2, b2. In another example, "a1- character-b2" refers to a name range list that includes the character variables from a1 to b2, which are variables a1 and a2. In a third example, "a1-numeric-b2" refers to a name range list that includes the numerical variables from a1 to b2, which are variables b1 and b2.

The Name Prefix List is used to refer to a variable list that shares the same prefix. In SAS, you could use the prefix followed by the ":" to select the variables. For example, "b:" could be used to select all the variables that have prefix "b" In the above data set, variables b1, b2, b3, b4 will be selected.

The Special SAS Name Lists include using the keyword "_NUMERIC_" to select all numeric variables in a data set, using the keyword "_CHARACTER_" to select all character variables in a data set, and using the keyword "_ALL_" to select all variables in a data set.

The "SAS variable list" is a convenient way for us to refer a list of variable names in SAS. However, the limitation of the "SAS variable list" is that it cannot be used for all the SAS procedures. The SELECT statement in PROC SQL is one example where the "SAS variable list" is inaccessible.

Also, we may want to select variables that share common naming patterns in their variable names. For example, we may want to subset those variables whose names start with "a," have "123" in the middle, and end with "b." Is there a way to do this without typing the variable names one by one?

The "selectVars" macro introduced in this paper is one solution! The features of this macro include:

a. Providing the same "SAS variable list" functions to select the variables
b. Providing the ability to select the variables with common patterns in their names
c. Allowing the variable list generated from the macro to be applied to all SAS procedures

THE SELECTVARS MACRO SAS CODES

Presented below is the SAS code for the selectVars macro. After calling the macro, the selected variables will be saved into a macro variable "lstVars," and you could use this macro variable to refer to the variable list in any SAS procedure.

```
%macro selectVars(libname=,datanm=,range=_ALL_,pattern=%,separateby=%str( ));
	data tmp;
		set &datanm.;
	run;

	%if "&range." ^=%str() %then %do;
		data tmp;
			set tmp;
			keep &range.;
		run;
	%end;

	%global lstVars;
	%let lstVars=%str();*reset the macro variable lstVars;
	proc sql noprint;
		select name into :lstVars separated by "&separateby."
		from dictionary.columns
		where libname=upcase("&libname") and memname = upcase("tmp")
		and name like "&pattern." escape '#';

		drop table tmp;
	quit;
%mend;
```

- The macro parameter "libname" is used to identify the library name for the dataset from which you want to select the variables.
- The macro parameter "datanm" is used to identify the dataset name from which you want to select the variables.
- The macro parameter "range" is used to identify the range in the dataset from which you want to select the variables. The values that are accepted by the "range" parameter are those that you can use for the "SAS variable list".

  Examples of using the "range" parameter:
  1. To select the variables a1, a2, a3, simply pass "a1-a3" to the parameter "range"
  2. To select the variables from a1 to b2 based on the variable order in the dataset, simply pass "a1--b2" to the parameter "range"
  3. To select the numerical variables from a1 to b2, simply pass a1-numeric-b2 to the parameter "range"
  4. To select variables whose name is prefixed with "a," simply pass "a:" to the parameter "range"
  5. Likewise, the keywords "_CHARACTER_", "_NUMERICAL_", "_ALL_" could also be passed to the parameter "range".

  By default, the value for the "range" parameter is _ALL_, if you want to define the variable range as all the variables, you don't need to pass a value to the "range" parameter.

- The macro parameter "patterns" is used to identify the naming patterns of the variables you want to select. In this paper, we use the "like" condition and the wildcard characters "%" and "_" in PROC SQL to help us identify the naming patterns. The wildcard character "_" could be used as a substitute for a single character, and the wildcard character "%" could be used as a substitute for zero or more characters.

  The sample code below illustrates use of the wildcard characters. In this example, we want to select those variables whose names have the suffix "_abc" from the dataset "test" saved in the "work" library.

  ```
  proc sql;
  ```

```
        select name
        from dictionary.columns
        where libname= upcase("work") and memname = upcase("test")
        and name like "%#_abc" escape '#';
quit;
```

The dictionary.columns is a table in SAS that saves all the variable names in the SAS datasets. The "libname" variable in the dictionary.columns table saves the library names of the SAS datasets. The values are saved in upper case, and hence the syntax uses the upcase function. The "memname" variable in the dictionary.columns table saves the data set names of the SAS datasets, and the values are also saved in upper case. The "like" condition in PROC SQL selects rows by comparing character strings with a pattern-matching specification. [2]

The "escape" clause is used to search for literal instances of the percent (%) and underscore (_) characters, which are usually used for pattern matching. SAS® allows for variable names beginning with "_," and it happens that "_" is a wildcard character. Thus, to select variables that contain the "_" in their names, we could use an ESCAPE character and add this ESCAPE character before the "_" to tell SAS® the "_" is not a wildcard character. In the above syntax, the "#" is used as the ESCAPE character.

Suppose you want to select those variables whose names start with an "a," followed by a character, then followed by "_123" in the middle, and end with "b." To accomplish this, you could pass "a_#_123%b" to the parameter "pattern."

The default value for the "pattern" parameter is %, which means to select all the patterns. If you want to select all the patterns, you don't need to pass a value to the "pattern" parameter.

- The macro parameter "separateby" instructs SAS® to separate the selected variables by a space or by a comma. By default, the variables are separated by a space. If you want to separate by comma, you could pass "%str(,)" to the "separateby."

EXAMPLES OF CALLING THE MACRO

The following SAS code is used to generate a SAS dataset called "testdata":

```
%macro testingdata();
      data testdata;
            a1=1;
            a2="a";
            a3=1;
            %do i=1 %to 6 %by 2;
                  %let j=%eval(&i.+1);
                  a&i._b&i._c&i.=1;
                  a&j._b&j._c&j. ="a";
            %end;
            b1=1;
            b2="a";
            b3=3;
      run;
%mend;
%testingdata();
```

Table 2 is the sample SAS data set called "testdata" generated by the above code:

a1	a2	a3	a1_b1_c1	a2_b2_c2	a3_b3_c3	a4_b4_c4	a5_b5_c5	a6_b6_c6	b1	b2	b3
1	a	1	1	a	1	a	1	a	1	a	3

Table 2. The "testdata" SAS Data Set

Example1: Select the variables a1, a2, a3 and separate the variables by comma:

```
%selectVars(libname=work,datanm=testdata,range=a1-a3,separateby=%str(,));
%put &lstVars.;
```

The result is: a1, a2, a3

Example2: Select the numerical variables from a1_b1_c1 to a6_b6_c6 and separate the selected variables by space:

```
%selectVars(libname=work,datanm=testdata,range=a1_b1_c1-numeric-a6_b6_c6);
%put &lstVars.;
```

The result is: a1_b1_c1 a3_b3_c3 a5_b5_c5

Example3: Select the variables whose names have prefix “a” and separate the variables by comma:

```
%selectVars(libname=work,datanm=testdata,pattern=a%,separateby=%str(,));
%put &lstVars.;
```

The pattern “a%” means the names’ first character is “a”, the “%” is the wildcard character that substitutes zero or more characters after “a”.

The result is: a1, a2, a3, a1_b1_c1, a2_b2_c2, a3_b3_c3, a4_b4_c4, a5_b5_c5, a6_b6_c6

Example4: Select the variables whose names’ middle parts are “_b”, followed by a character, and followed by “_c”, separate the variables by space:

```
%selectVars(libname=work,datanm=testdata,pattern=%#_b_#_c%);
%put &lstVars.;
```

“#_” means treating “_” as underscore instead of the wild card.

The result is: a1_b1_c1 a2_b2_c2 a3_b3_c3 a4_b4_c4 a5_b5_c5 a6_b6_c6

Example5: Select the variables whose names end with “_c”, followed by a character, separate the variables by comma:

```
%selectVars(libname=work,datanm=testdata,pattern=%#_c_, separateby=%str(,));
%put &lstVars.;
```

The result is: a1_b1_c1, a2_b2_c2, a3_b3_c3, a4_b4_c4, a5_b5_c5, a6_b6_c6

Example6: Select the character variables from a1 to a6_b6_c6 whose names’ suffix is 2; separate the selected variables by comma:

```
%selectVars(libname=work,datanm=testdata,range=a1-character-a6_b6_c6,
pattern=%2,separateby=%str(,));
%put &lstVars.;
```

The result is: a2, a2_b2_c2

AN EXAMPLE OF USING THE LSTVARS IN THE PROC SQL

Below is an example about how to use the macro variable lstVars in the SELECT statement in the SQL procedure, make sure you have used the comma to separate the variables saved in the lstvars:

```
proc sql;
        select &lstVars.
        from testdata;
quit;
```

CONCLUSION

The macro presented in this paper provides an easy way to select the variables you want from a data set. The result of the variable list is saved in the macro variable "lstVars" and you could use &lstVars for any SAS procedure. This macro can be modified for use in wider settings, such as to rename the variables in batches or select the date sets in batches.

REFERENCES

1. SAS(R) 9.2 Language Reference: Concepts, Second Edition. "SAS Variable Lists". Available at http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000695105.htm

2. Base SAS(R) 9.2 Procedures Guide. "LIKE condition". Available at http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473693.htm

ACKNOWLEDGMENTS

The authors wish to thank the Division of Biostatistics and Epidemiology at Cincinnati Children's Hospital Medical Center for its support, and particularly Matthew Fenchel, Robert Tamer and the SAS Programming Group for their helpful feedback.

CONTACT INFORMATION

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

Name: Ting Sa
Enterprise: Division of Biostatistics and Epidemiology, Cincinnati Children's Hospital Medical Center
Address: 3333 Burnet Ave
City, State ZIP: Cincinnati, OH 45229
Work Phone: 513-636-3674
E-mail: ting.sa@cchmc.org

Name: Yanhong Liu
Enterprise: Division of Biostatistics and Epidemiology, Cincinnati Children's Hospital Medical Center
Address: 3333 Burnet Ave
City, State ZIP: Cincinnati, OH 45229
Work Phone: 703-263-0161
E-mail: yanhong.liu@cchmc.org