PhUSE 2016

Paper CS09

Top 10 uses of macro %varlist - in proc SQL, Data Step and elsewhere

Jean-Michel Bodart, Business & Decision Life Sciences, Brussels, Belgium

ABSTRACT

The SAS® macro-function %VARLIST(), a generic utility function that can retrieve lists of variables from one or more datasets, and process them in various ways in order to generate code fragments, was presented at the PhUSE 2015 Annual Conference in Vienna. That first paper focused on generating code for SQL joins of arbitrary complexity. However the macro was also meant to be useful in the Data Step as well as in other procedure calls, in global statements and as a building block in other macros.

The full code of macro %VARLIST is freely available from the PhUSE Wiki (http://www.phusewiki.org/wiki/index.php?title=SAS_macro-function_%25VARLIST).

The current paper reviews and provides examples of the top ten uses of %VARLIST() according to a survey of real-life SAS programs developed to create Analysis Datasets, Summary Tables and Figures in post-hoc and exploratory analyses of clinical trials data. It is meant to provide users with additional information about how and where to use it best.

INTRODUCTION

SAS Macro Language has been available for a long time. It provides users with the tools to control program flow execution, execute repeatedly and/or conditionally single or multiple SAS Data steps, Procedure steps and/or Global statements that are aggregated, included in user-written macro definitions. However, the macro statements that implement the repeating and conditional loops cannot be submitted on their own, as “open code”, but only as part of calls to the full macro definition.

On the opposite side, at a smaller level, macro-functions can be used to manipulate character strings passed as arguments, either as literals, as macro-variables references, as values returned by other macro-functions calls, or as combinations of these. The character string values returned by macro-functions can be used (directly or via assignment to macro-variables) as (or combined with) literal values in dataset variables, labels, titles or footnotes. Alternatively they can be utilized in building (fragments of) base SAS code: this is the code-generating aspect of SAS macros. A big advantage of macro-functions is that they can be used in open code, they don't need to be included or submitted as part of a larger macro-definition. This comes handy when developing or debugging code through step-by-step interactive execution. Moreover they don't require the users to master the macro-language themselves, only to know the usage and syntax of the macro-function.

To be complete, let's mention that small or big macro-programs (that must be included in full macro-definitions) can also be written to generate (non-macro) SAS code and/or literals. While these typically offer more flexibility than macro-functions, they also require a good understanding of the Macro language to create, maintain and understand them, and may be longer to write and debug.

The SAS System comes with a number of built-in macro-functions, and most data step functions can be called from macro code via the %SYSFUNC() macro-function, but the macro language also allows advanced users to create new macro functions and thus extend the available functionality. %VARLIST() is one of such user-written macro-functions, which can then be made available, with appropriate documentation, to whole communities of users. The purpose of this paper is to contribute to the documentation of macro %VARLIST().

MATERIALS AND METHODS

In April 2016, the author conducted a search of commonly used statistical programming areas for SAS programs containing calls to %VARLIST() Macro. The search was performed using the UltraEdit Professional Text/HEX Editor (IDM Computers Solutions, Inc.) function “Search in Files”, and was updated in July 2016. These macro calls were reviewed and categorized according to their general purpose, context (Data Step, Procedure Step, Macro, and Global Statements), context details and detailed purpose. The corresponding program types were also identified. The data were summarized in Excel, and the 10 most frequent uses by context were identified and illustrated by representative code excerpts.

RESULTS

A total of 727 occurrences in 67 files (SAS programs) were identified and categorized.

PROGRAM TYPES

The corresponding program types were as follows:

Program type	Number of distinct programs	Number of %VARLIST() calls
Table/ Figure (development)	47	604
Analysis Dataset (development)	15	72
Table/ Figure (validation)	4	43
Miscellaneous checks	1	8
Total	**67**	**727**

PROGRAMMING CONTEXT

The %VARLIST() calls were observed in the following contexts, but most frequently (79.9% of total) in PROC SQL, DATA STEP, and Macros, and rather frequently (13.6% of total) in PROC SGPLOT, PROC TEMPLATE, Global Statements and PROC PRINT. Other contexts represented less than 5% of the total calls.

Context	Number and Percentage of calls	
proc sql	238	32.7%
data step	235	32.3%
macro	108	14.9%
proc sgplot	49	6.7%
proc template	31	4.3%
global statement	19	2.6%
proc print	16	2.2%
data step; macro	10	1.4%

Context	Number and Percentage of calls	
proc sort	7	1.0%
proc report	5	0.7%
proc summary	4	0.6%
proc compare	2	0.3%
macro-definition	1	0.1%
proc freq	1	0.1%
proc transpose	1	0.1%

CONTEXT DETAILS

In PROC SQL, the simple ORDER BY, SELECT and GROUP BY clauses together made more than 85% of the %VARLIST() calls, with an additional 5.9% comprised of more elaborate versions of the SELECT clause; the FROM, ON and WHERE clauses together made less than 5% of the calls.

In the DATA STEP, the LABEL statements together with Input and Output DATASET OPTIONS accounted for 77.8% of the %VARLIST() calls; the BY and IF statements made a bit more than 5% each. Value-modifying statements (variable assignment and Call Missing statements together) were also involved in an additional 5.2%. ATTRIB and LENGTH statements together represented 3.9%. MERGE and PUT statements were each found in less than 1% of the calls.

In macros, all of %VARLIST() calls occurred within %IF statements.

In PROC SGPLOT, almost 90% of the calls were made inside XAXIS, YAXIS and KEYLEGEND statements, while LABEL and SCATTER statements accounted for the rest.

In PROC TEMPLATE, %VARLIST() calls were spread between 4 statements as part of DEFINE STATGRAPH blocks: ENTRY, DISCRETELEGEND, COLUMNAXIS and ROWAXIS.

Occurrences of %VARLIST() in global statements were in %LET statements for about two thirds, and in TITLE statements for about one third.

GENERAL PURPOSES OF %VARLIST() CALLS

The general purposes of using %VARLIST() in the programs reviewed were distributed as follows (only those found in 5% or more of these macro calls are presented):

Purpose	Number and percentage of Calls	
Generate list of unique variable names (from literal variable names and/or macro-variables or macro parameters containing 0 or more variable names) and optionally insert specific separators; optionally excluding specific variable names	161	22.1%
Check whether (at least one of) a given (set of) variable(s) exists in a given dataset	127	17.5%
Transfer variable labels (+ optional modification)	95	13.1%
Retrieve and concatenate labels of one or more variables for use in title, footnote or graph element label	92	12.7%
Retrieve those unique variable names from an input list (possibly with macro-variable references) that exist in one or more specific dataset(s); optionally excluding specific variable names	66	9.1%
Insert delimiters between space-separated list of literal variable names	39	5.4%
Modify existing label of a variable	36	5.0%

PURPOSES OF %VARLIST() CALLS, BY CONTEXT

The detailed purposes of %VARLIST() calls by context are presented in the table below (only the categories with the most number of calls are presented):

General purpose / Purpose details \ Context	Order by clause	%If statement	Label statement	Select clause	(Input) dataset option	(Output) dataset option	By statement	Yaxis statement	Group by clause	Var statement	Alter table <table> modify <var> label=<label>	Grand Total
Generate list of unique variable names from literal variable names and/or macro-variables or macro parameters (containing 0 or more variable names) and optionally insert specific separators, with optional exclusion of specific variable names	87			14		11	19		12	2		**145**
Order by clause	87											87
Group by clause									12			12
Define First.<var> and Last.<var>							7					7
Check whether (at least one of) a given (set of) variable(s) exists in a given dataset		108										**108**
Conditional execution		108										108
Transfer variable labels (+ optional modification)			54	20							10	**84**
Copy label from one variable to another variable			19	7								26
Assign modified label to a variable			4	4							10	18
Retrieve those unique variables from an input list (possibly with macro-variable references) that exist in one or more specific dataset(s), with optional exclusion of specific variable names				1	32	19	5			1		**58**
Drop = <unique list of existing variables>					14	12						26
Insert delimiters between space-separated list of literal variable names	32								7			**39**
Keep list of variables unchanged for easy copy & paste between SQL and non-SQL code	32								6			38
Modify existing label of a variable			36									**36**
Assign modified label to a variable			12									12
Retrieve the exact name of a variable generated by SAS, that matches a pre-defined pattern (e.g. the bin variable from output dataset of SGPLOT HISTOGRAM procedure), and rename it to a specific name					24							**24**
Rename variable matching pattern to a standard name					24							24
Retrieve those unique variables that exist in one or more specific dataset(s), with optional exclusion of specific variable names, and apply a specific pattern				21								**21**
Generate comma-separated list of variables with a dataset name/alias qualifier for use in SQL JOIN				18								18
Retrieve and concatenate labels of one or more variables for use in title, footnote or graph element label								20				**20**
Assign axis label								18				18
Retrieve those unique variables that exist in one or more specific dataset(s), with optional exclusion of specific variable names				7		1				10		**18**
Assign style										10		10
Drop = <unique list of existing variables>						1						1
Generate a comma-separated list of variables for use in simple SQL SELECT clause				7								7
Grand Total	**119**	**108**	**92**	**63**	**56**	**31**	**24**	**20**	**19**	**13**	**10**	**447**

The top 10 uses of %VARLIST() by context (shaded cells in above table) have been identified as:

1. To check whether (at least one of) a given (set of) variable(s) exists in a given dataset, and conditionally execute some code accordingly, using %IF statement in macro definition (n=108).
2. To generate a list of unique variable names from literal variable names and/or macro-variables or macro parameters (containing 0 or more variable names) and optionally insert specific separators, with optional exclusion of specific variable names, in PROC SQL SELECT clause (n=87).
3. To insert additional separators (commas) between space-separated literal variable names, for easier code maintenance, so the list can be copied and pasted unchanged between SQL (e.g. ORDER BY clause) and non-SQL code (e.g. BY statement) (n=32).

4. To retrieve the exact name of a variable generated by SAS, that matches a pre-defined pattern (e.g. the bin variable from output dataset of the HISTOGRAM statement in SGPLOT procedure), and rename it to a specific name, using (input) dataset option RENAME, (e.g.) in DATA STEP SET statement (n=24).
5. To copy the label from one variable to another variable, (e.g.) in DATA STEP LABEL statement (n=19).
6. To retrieve those unique variables that exist in one (or more) specific dataset(s), with optional exclusion of specific variable names, and by means of a specific pattern to generate a comma-separated list of variables with a dataset name/alias qualifier for use in SQL JOIN (n=18).
7. To retrieve (and concatenate) the label(s) of one (or more) variable(s) for use in title, footnote or graph element label, e.g. to assign an axis label in PROC SGPLOT YAXIS statement (n=18).
8. To retrieve those unique variables from an input list (possibly with macro-variable references) that exist in one (or more) specific dataset(s), with optional exclusion of specific variable names, for use in Input dataset option DROP, (e.g.) in DATA STEP SET statement (n=14).
9. To retrieve those unique variables from an input list (possibly with macro-variable references) that exist in one (or more) specific dataset(s), with optional exclusion of specific variable names, for use in Output dataset option DROP, (e.g.) in DATA STEP SET statement (n=12).
10. To generate a list of unique variable names from literal variable names and/or macro-variables or macro parameters (containing 0 or more variable names) and optionally insert specific separators, with optional exclusion of specific variable names, in PROC SQL ORDER BY clause (n=12).

ILLUSTRATION BY CODE EXCERPTS

Code extracts from a few real-life programs were selected to illustrate the most frequent uses of %VARLIST() as identified above. The general purpose of the original program is provided first, followed by annotated sections of interest. Relevant parts of the SAS log and outputs are also presented. The original code has generally been simplified and slightly amended in order to focus on the points of interest, but remains fully functional.

FIRST EXAMPLE: A SUMMARY TABLE PROGRAM

The program creates summary tables with descriptive statistics (n, mean, 95% CI …) of change from baseline in various efficacy variables, by categories of plasma concentration of distinct study drugs over time and by treatment sequence. It's also possible to summarize across time points if required.

The code excerpts below illustrate how %VARLIST() can be used :

1) to generate comma-separated lists of unique variable names from a combination of literal names and names specified in macro-variables, to use (a) in PROC SQL GROUP BY and (b) in ORDER BY clauses.
2) to dynamically specify variables to DROP in corresponding (input) dataset option, only when they exist, and without generating an ERROR message and having SAS stop processing the (data) step when they don't. This is exemplified for the SET statement in a DATA step.
3) to make variable names (passed as literals and/or macro-variables references) unique (by default), e.g. in a BY statement.
4) finally, the code shows how in a DATA step the last name from a list of unique variable names returned by %VARLIST() can be easily retrieved by the %SCAN() macro-function for use in FIRST.<variable> or LAST.<variable> references, after being defined by a similar %VARLIST() call in a BY statement.

The summarized dataset sum2a that is first used contains a Cartesian product of the possible values of all combinations of the various categorical variables, which was created in order to present n=0 for cells with zero observations, rather than presenting empty cells.
However to save some space it was later requested to suppress the ranges of concentrations that had no observation for any treatment sequence. The suppression of the unwanted observations is done in the PROC SQL code below. The subsequent DATA STEP derives a page variable which is later used to limit to three the number of blocks of summary statistics displayed per page.

```
%macro summary(bygrpn=, bygrpl=, xvarn=, xvarl=, grp=, subgrpn=, subgrp=, subgrpl=);
[..]
proc sql noprint;
  create table sum2b as
    select * from sum2a
      group by

            %varlist(var=PARAMN PARAMCD PARAM &bygrpn &bygrpl &xvarn &xvarl , sep=#cs#)
      having sum(n)>0

      order by

            %varlist(var=PARAMN PARAMCD PARAM &bygrpn &bygrpl &xvarn &xvarl &grp.n &grp
                     &grp.l &subgrpn &subgrp &subgrpl statn, sep=#cs#);
quit;
```

1) a) %VARLIST() call that generates a list of unique variable names (from the names specified by the var= parameter), separated by comma and space (as specified by sep= parameter – the keyword #cs# stands for "comma, then space") – the generated list is passed to the GROUP BY clause.

1) b) %VARLIST() call similar to (a) but with more variable names – the generated list is passed to the ORDER BY clause.

```
data sum2;
  label page = "Page bloc for table (every 3 blocks of summary statistics)";
  set sum2b(drop=
```

2) %VARLIST() call that retrieves from dataset sum2b (as specified by parameter data=) the variable names matching (case insensitive) those specified by parameter var=, only if found in that dataset. The generated list is passed to the DROP= (input) dataset option. This ensures variables 'page' and 'vn' are not read from the input dataset, which is necessary to (implicitly) retain those variables across observations, for counting. Note: an empty DROP dataset option would be considered valid SAS code.

```
%varlist(data=sum2b, var=page vn));
  by
```

3) %VARLIST() call that generates a list of unique (variable) names from those specified by parameter var=. By default, the generated list is made of the first occurrence of each variable name specified in the var= parameter, in the same order. The generated list is passed to the BY statement, in order to define First.<var> and Last.<var> variables for by-group processing in the next statements.

```
    %varlist(var=PARAMN PARAMCD PARAM &bygrpn &bygrpl &xvarn &xvarl &grp.n &grp
                &grp.l &subgrpn &subgrp &subgrpl statn);
```

4) The %SCAN() macro-function uses the unique list of variables names returned by the %VARLIST() call as its first argument, from which it extracts the last word (=variable name) in order to identify the first occurrence of each new value of that variable.

```
  if first.%scan(%varlist(var=PARAMN PARAMCD PARAM &bygrpn &bygrpl), -1) then do;
    vn=0; *- At the start of each parameter group, re-initialise vn (counter
             of by-groups with distinct values of &xvarn) -*;
  end;
  if first.&xvarn then do;
    if mod(vn, 3)=0 then page+1; *- increment page number every 3 by-groups
                                    with distinct values of &xvarn -*;
    vn+1; *- increment vn at the start of each by-groups
             with a new value of &xvarn -*;
  end;
run;
[..]
%mend summary;

*- Summarize by parameter, analysis week, analysis value category and treatment sequence -*;
%summary(bygrpn=AWEEKn, bygrpl=AWEEK, xvarn=AVAL, xvarl=AVALGRP, grp=TRTSEQA );
*- Summarize across weeks, by parameter, analysis value category and treatment sequence -*;
%summary(bygrpn=, bygrpl=, xvarn=AVAL, xvarl=AVALGRP, grp=TRTSEQA );
```

The text returned by the %VARLIST() calls appears in the LOG, and has been highlighted and rendered as bold for easier identification. Note that with the MPRINT option set, lines that are completely generated by %VARLIST() are identified with "MPRINT(VARLIST):", contrary to lines that are only partially generated by %VARLIST():

```
MPRINT(SUMMARY):   proc sql noprint;
MPRINT(SUMMARY):   create table sum2b as select * from sum2a group by
MPRINT(VARLIST):   PARAMN, PARAMCD, PARAM, AWEEKn, AWEEK, AVAL, AVALGRP
MPRINT(SUMMARY):   having sum(n)>0 order by
MPRINT(VARLIST):   PARAMN, PARAMCD, PARAM, AWEEKn, AWEEK, AVAL, AVALGRP, TRTSEQAn, TRTSEQA, TRTSEQAl, statn
MPRINT(SUMMARY):  ;
[..]
MPRINT(SUMMARY):   data sum2;
MPRINT(SUMMARY):   label page = "Page bloc for table (every 3 blocks of summary statistics)";
MPRINT(SUMMARY):   set sum2b(drop
MPRINT(SUMMARY):  = PAGE VN);
MPRINT(SUMMARY):   by
MPRINT(VARLIST):   PARAMN PARAMCD PARAM AWEEKn AWEEK AVAL AVALGRP TRTSEQAn TRTSEQA TRTSEQAl statn
MPRINT(SUMMARY):  ;
MPRINT(SUMMARY):   if first.AWEEK then do;
MPRINT(SUMMARY):   vn=0;
MPRINT(SUMMARY):   *- At the start of each parameter group, re-initialise vn (counter of by-groups with distinct values
of &xvarn) -*;
MPRINT(SUMMARY):   end;
MPRINT(SUMMARY):   if first.AVAL then do;
MPRINT(SUMMARY):   if mod(vn, 3)=0 then page+1;
MPRINT(SUMMARY):   *- increment page number every 3 by-groups with distinct values of &xvarn -*;
MPRINT(SUMMARY):   vn+1;
MPRINT(SUMMARY):   *- increment vn at the start of each by-groups with a new value of &xvarn -*;
MPRINT(SUMMARY):   end;
MPRINT(SUMMARY):   run;
```

The dataset sum2 looks as follows (only a selection of records of variables are displayed):

PAGE	PARAMN	AWEEKN	AWEEK	AVAL	AVALGRP	TRTSEQAN	N	MEAN	SD	MEDIAN	Q1	Q3	MIN	MAX	STATN	STAT	VALUE
25	1	96	Week 96	75	72.5 - <77.5	1	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	1	n	1
25	1	96	Week 96	75	72.5 - <77.5	1	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	2	Mean (SD)	-1.306 (NC)
25	1	96	Week 96	75	72.5 - <77.5	1	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	3	95% CI	[NC , NC]
25	1	96	Week 96	75	72.5 - <77.5	1	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	4	Median	-1.306
25	1	96	Week 96	75	72.5 - <77.5	1	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	6	Min-Max	0
25	1	96	Week 96	75	72.5 - <77.5	2	0								1	n	0
25	1	96	Week 96	75	72.5 - <77.5	2	0								2	Mean (SD)	
25	1	96	Week 96	75	72.5 - <77.5	2	0								3	95% CI	
25	1	96	Week 96	75	72.5 - <77.5	2	0								4	Median	
25	1	96	Week 96	75	72.5 - <77.5	2	0								6	Min-Max	
25	1	96	Week 96	75	72.5 - <77.5	99	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	1	n	1
25	1	96	Week 96	75	72.5 - <77.5	99	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	2	Mean (SD)	-1.306 (NC)
25	1	96	Week 96	75	72.5 - <77.5	99	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	3	95% CI	[NC , NC]
25	1	96	Week 96	75	72.5 - <77.5	99	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	4	Median	-1.306
25	1	96	Week 96	75	72.5 - <77.5	99	1	-1.306		-1.306	-1.306	-1.306	-1.31	-1.31	6	Min-Max	0
25	1	96	Week 96	95	92.5 - <97.5	1	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	1	n	1
25	1	96	Week 96	95	92.5 - <97.5	1	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	2	Mean (SD)	-1.483 (NC)
25	1	96	Week 96	95	92.5 - <97.5	1	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	3	95% CI	[NC , NC]
25	1	96	Week 96	95	92.5 - <97.5	1	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	4	Median	-1.483
25	1	96	Week 96	95	92.5 - <97.5	1	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	6	Min-Max	0
25	1	96	Week 96	95	92.5 - <97.5	2	0								1	n	0
25	1	96	Week 96	95	92.5 - <97.5	2	0								2	Mean (SD)	
25	1	96	Week 96	95	92.5 - <97.5	2	0								3	95% CI	
25	1	96	Week 96	95	92.5 - <97.5	2	0								4	Median	
25	1	96	Week 96	95	92.5 - <97.5	2	0								6	Min-Max	
25	1	96	Week 96	95	92.5 - <97.5	99	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	1	n	1
25	1	96	Week 96	95	92.5 - <97.5	99	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	2	Mean (SD)	-1.483 (NC)
25	1	96	Week 96	95	92.5 - <97.5	99	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	3	95% CI	[NC , NC]
25	1	96	Week 96	95	92.5 - <97.5	99	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	4	Median	-1.483
25	1	96	Week 96	95	92.5 - <97.5	99	1	-1.483		-1.483	-1.483	-1.483	-1.48	-1.48	6	Min-Max	0

For AWEEK="WEEK 96", the AVALGRP categories "77.5 - <82.5", "82.5 - <87.5" and "87.5 - < 92.5" have been deleted (their position is indicated by the thick horizontal line).

The categories "72.5 - <77.5" and "92.5 - <97.5" were kept because one observation is present in First Treatment Sequence (TRTSEQAN=1) and in Overall Treatment sequence (TRTSEQAN=99).

SECOND EXAMPLE: A PROGRAM EXCERPT THAT COMBINES MULTIPLE DATASETS AND CHECKS FOR DUPLICATES

The following program excerpt combines data from different sources: Plasma Samples (plasma1b), Subject-Level Analysis Dataset (ADSL), additional subject characteristics retrieved from other sources (pATFN and RFsta), then checks for duplicates before further processing the data.

The code illustrates how %VARLIST() can be used to :

1) Retrieve those unique variables that exist in one or more specific dataset(s);
2) Exclude specific variable names;
3) Generate comma-separated list of variables, with a pre-pended dataset name/alias qualifier for use in SQL JOIN;
4) Keep list of variables unchanged for easy maintenance (copy & paste without having to insert / remove commas) between SQL and non-SQL code.

```
proc sql noprint;
  create table plasma1b as
    select subject, a.usubjid, a.SITEID, AVISITN, AVISIT, VISITNUM, VISIT, SAMPLEID
         , PARAMCD, METHOD, PCNAM, AVALC, AVAL, PARAM, PCDTC
         , c.pATNF
         , d.RFsta
         ,%varlist(data=adsl
                  ,var=#all# #not# usubjid
                  ,sep=#cs#, pattern=e.#var#)
    from  plasma1 as a
      left join
        pATNF as c
        on c.usubjid=a.usubjid
      left join
        RFsta as d
        on d.usubjid=a.usubjid
      left join
        ADSL as e
        on e.usubjid=a.usubjid
```

The %VARLIST() call :

1) Retrieves unique variables names from specified dataset (data=adsl)
2) Keeps all of them except 'usubjid' (var=#all# #not# usubjid)
3) Returns the list of variable names separated by comma and space (sep=#cs#), and preceded by 'e.' (pattern=e.#var#, where the keyword #var# refers to the bare variable name) as part of the SQL SELECT clause, in order to refer specifically to those variables from dataset ADSL (which is assigned the alias name 'e' in the SQL LEFT JOIN clause)

```
      order by
```

4) The %VARLIST() call that inserts commas between variable names passed as literals allows keeping the list of variables unchanged for easy maintenance (copy & paste without having to insert / remove commas) between SQL and non-SQL code.

```
               %varlist(var=subject avisitn avisit visitnum visit sampleid paramcd pcdtc method
                             pcnam
                      , sep=#cs#);
quit;
data plasma1c plasma1b_dups;
   set plasma1b;
   by subject avisitn avisit visitnum visit sampleid paramcd pcdtc method pcnam;
   if (not last.pcnam) or (not first.pcnam) then output plasma1b_dups;
   if first.pcnam then output plasma1c;
run;
```

The text returned by the %VARLIST() calls appears in the LOG, and has been rendered as bold for easier identification. Note that with the MPRINT option set, lines that are completely generated by %VARLIST() are identified with "`MPRINT(VARLIST):`" :

```
1407  proc sql noprint;
1408    create table plasma1b as
1409      select subject, a.usubjid, a.SITEID, AVISITN, AVISIT, VISITNUM, VISIT, SAMPLEID
1410           , PARAMCD, METHOD, PCNAM, AVALC, AVAL, PARAM, PCDTC
1411           , c.pATNF
1412           , d.RFsta
1413           ,%varlist(data=adsl
1414                    ,var=#all# #not# usubjid
1415                    ,sep=#cs#, pattern=e.#var#)
MPRINT(VARLIST):  e.SEX, e.SEXN, e.AGE, e.WEIGHT, e.BMI, e.BMICAT, e.ARMCD, e.ARM
1416      from  plasma1 as a
1417        left join
1418          pATNF as c
1419          on c.usubjid=a.usubjid
1420        left join
1421          RFsta as d
1422          on d.usubjid=a.usubjid
1423        left join
1424          ADSL as e
1425          on e.usubjid=a.usubjid
1426        order by
1427                %varlist(var=subject avisitn avisit visitnum visit sampleid paramcd pcdtc
1428                         method pcnam
1429                       , sep=#cs#)
MPRINT(VARLIST):   subject, avisitn, avisit, visitnum, visit, sampleid, paramcd, pcdtc, method, pcnam
1430        ;
```

THIRD EXAMPLE: A FIGURE PROGRAM THAT CREATES BLAND-ALTMAN PLOTS AND HISTOGRAMS

After (A) creating a dummy set of data, the following program (modified excerpts are presented, annotated with comments) generates (B) Bland-Altman plots (plots of the difference between two variables against their average) and (C) histograms plots and corresponding datasets, in order to evaluate the results consistency of different methods to measure a common parameter (originally, plasma concentrations of investigational drug and anti-drug antibodies).

The code illustrates how %VARLIST() can be used to easily:

1) drop some unwanted variables, but only if they exist in input dataset (and avoid a WARNING message in case they did not exist), using (output) dataset option DROP, e.g. in a DATA STEP;
2) retrieve a variable label and assign it back to the same or another variable with or without modifications, in DATA or PROC steps, and even if the source and/or target variables are dynamically specified, e.g. passed as macro-parameters;
3) retrieve a variable label and assign its text (with or without modifications) as label to a graph element (e.g. an axis) in a graphical procedure such as PROC SGPLOT;
4) execute code conditionally, according to the existence of a specific variable in a specific dataset (e.g. for error checking), when used as part of a %IF statement within a macro definition;
5) rename variables with long and inconsistent, automatically generated names to standard, shorter names, using (input) dataset option RENAME e.g. in a DATA STEP.

Code part (A) - creating a dummy set of data:

```
*- Create dummy data for Bland-Altman comparison of Methods A vs B and A vs C
   in measurement of Parameter P -*;
data BlandAltman;
   set sashelp.class(keep=Name Height Weight Age Sex
                     rename=(Height=MethA Weight=MethB Name=ID age=X));
   MethA = MethA + 50;
   MethC = 200 + X - MethB;
   MeanAB = (MethA + MethB) / 2;
   DiffAB = MethA - MethB;
   MeanAC = (MethA + MethC) / 2;
   DiffAC = MethA - MethC;
```

```
   *- Assign variable labels with variable name appended in square brackets,
      for easier identification -*;
   label   ID = "Subject ID [ID]"
           X = "Age [X]"
           SEX = "Gender [Sex]"
           MethA = "Parameter P (Method A) [MethA]"
           MethB = "Parameter P (Method B) [MethB]"
           MethC = "Parameter P (Method C) [MethC]"
           MeanAB = "Average (Method A & Method B) in Parameter P [MeanAB]"
           DiffAB = "Difference (Method A - Method B) in Parameter P [DiffAB]"
           MeanAC = "Average (Method A & Method C) in Parameter P [MeanAC]"
           DiffAC = "Difference (Method A - Method C) in Parameter P [DiffAC]";
run;
title "Print dataset BlandAltman (show labels - only 3 obs are shown for brevity)";
proc print data=BlandAltman(obs=3) label noobs;
run;
```

Output of part (A):

Print dataset BlandAltman (show labels - only 3 obs are shown)

Subject ID [ID]	Gender [Sex]	Age [X]	Parameter P (Method A) [MethA]	Parameter P (Method B) [MethB]	Parameter P (Method C) [MethC]	Average (Method A & Method B) in Parameter P [MeanAB]	Difference (Method A - Method B) in Parameter P [DiffAB]	Average (Method A & Method C) in Parameter P [MeanAC]	Difference (Method A - Method C) in Parameter P [DiffAC]
Alfred	M	14	119.0	112.5	101.5	115.75	6.5	110.25	17.5
Alice	F	13	106.5	84.0	129.0	95.25	22.5	117.75	-22.5
Barbara	F	13	115.3	98.0	115.0	106.65	17.3	115.15	0.3

Code part (B) – generating Bland-Altman plots:

```
*- Use a macro to print and plot the data, and compare two different methods in each call -*;
%macro test(data=, x=, y=, group=, num=);
```

1) %VARLIST() is used in DATA step, (output) dataset option DROP, to drop variables X and Y (if they existed in input dataset) - this is necessary before renaming &x to X and &y to Y.

```
  data renamed&num.(drop= %varlist(data=&data, var= X Y)
                    index=(&group)
                    rename=(&x=X &y=Y));
     set &data;
     *- Replace label of variable Y by some pre-specified text (for illustration) -*;
     label &y = "Variable Y [&y]";
```

2) a) %VARLIST() is used together with %SYSFUNC(TRANWRD()) in DATA step LABEL statement, to retrieve the label of variable &x (var=&x) from dataset &data (data=&data), within double quotes (pattern=#vlabelq#), and to insert split character "|" (between "+" signs for easier visualisation) before word "in".

```
     label &x=%sysfunc(tranwrd(%varlist(data=&data, var=&x, pattern=#vlabelq#)
                               , %str( in ), %str( +|+in )));
  run;

  title "(Test &num) Print dataset renamed&num (showing labels - only 3 obs are shown)";
  title2 "Varlist() in data step, label statement: In dataset renamed&num, the label of variable
&x (renamed as X) was modified (split character '|' added between '+' signs before word 'in')";
   proc print data=renamed&num(obs=3) label split="|" noobs;
    title3 "Varlist() in proc step, label statement: Modify label of variable &y (renamed as Y):
replace parentheses by split characters '|', and append '***' ";
```

2) b) %VARLIST() is used in proc (print) step together with %SYSFUNC(TRANSLATE()), to retrieve the label of variable &y (var=&y) from dataset &data (data=&data), as non-quoted text (pattern=#vlabel#), to replace parentheses by split characters "|", and append 3 stars (for easier visualization)

```
    label Y="%sysfunc(translate(%varlist(data=&data, var=&y, pattern=#vlabel#), ||, ())) ***";
  run;

  title "(Test &num) Plot dataset renamed&num";
  proc sgplot data=renamed&num.;
    scatter x=X y=Y / group=&group;
    refline 0 / axis=Y;
    title2 "VARLIST() in proc SGPLOT, XAXIS statement: Retrieve the original label of variable
&x in dataset &data and assign it unchanged to the X axis." ;
```

3) a) %VARLIST() is used in proc (sgplot) to retrieve the label of variable &x (var=&x) from dataset &data (data=&data), as quoted text (pattern=#vlabelq#), and assign it as label to the X axis.

```
    xaxis label=%varlist(data=&data, var=&x, pattern=#vlabelq#);
    title3 "VARLIST() in proc SGPLOT, YAXIS statement: Retrieve the label from variable &y in
dataset renamed&num, shorten it to only the text between parentheses, and assign it to the Y axis." ;
```

3) b) %VARLIST() is used together with %SCAN() in proc (sgplot) to retrieve the label of variable &y (var=&y) from dataset &data (data=&data), as non-quoted text (pattern=#vlabel#), extract the part between parentheses, and assign it as label to the Y axis.

```
    yaxis label="%scan(%varlist(data=&data, var=&y, pattern=#vlabel#), 2, ())";
  run;
%mend test;

%test(data=BlandAltman, x=MeanAB, y=DiffAB, group=Sex, num=1);
     *- note input dataset 'BlandAltman' has a variable named X (to be dropped) -*;
%test(data=renamed1,    x=MeanAC, y=DiffAC, group=Sex, num=2);
     *- note input dataset 'renamed1' has variables named X and Y (to be dropped) -*;
```

Outputs of part (B):

(Test 1) Print dataset renamed1 (showing labels - only 3 obs are shown for brevity)

Varlist() in data step, label statement: In dataset renamed1, the label of variable MeanAB (renamed as X) was modified (split character '|' added between '+' signs before word 'in')

Varlist() in proc step, label statement: Modify label of variable DiffAB (renamed as Y): replace parentheses by split characters '|', and append '***'

Subject ID [ID]	Gender [Sex]	Parameter P (Method A) [MethA]	Parameter P (Method B) [MethB]	Parameter P (Method C) [MethC]	Average (Method A & Method B) + +in Parameter P [MeanAB]	Difference Method A - Method B in Parameter P [DiffAB] ***	Average (Method A & Method C) in Parameter P [MeanAC]	Difference (Method A - Method C) in Parameter P [DiffAC]
Alfred	M	119.0	112.5	101.5	115.75	6.5	110.25	17.5
Alice	F	106.5	84.0	129.0	95.25	22.5	117.75	-22.5
Barbara	F	115.3	98.0	115.0	106.65	17.3	115.15	0.3

(Test 2) Print dataset renamed2 (showing labels - only 3 obs are shown for brevity)

Varlist() in data step, label statement: In dataset renamed2, the label of variable MeanAC (renamed as X) was modified (split character '|' added between '+' signs before word 'in')

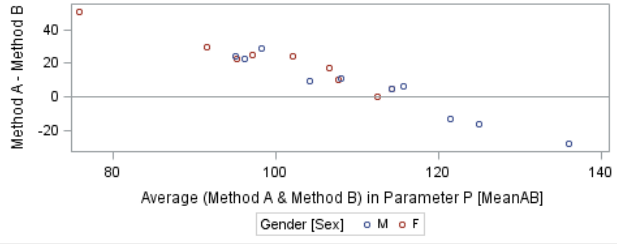
Varlist() in proc step, label statement: Modify label of variable DiffAC (renamed as Y): replace parentheses by split characters '|', and append '***'

Subject ID [ID]	Gender [Sex]	Parameter P (Method A) [MethA]	Parameter P (Method B) [MethB]	Parameter P (Method C) [MethC]	Average (Method A & Method C) + +in Parameter P [MeanAC]	Difference Method A - Method C in Parameter P [DiffAC] ***
Alfred	M	119.0	112.5	101.5	110.25	17.5
Alice	F	106.5	84.0	129.0	117.75	-22.5
Barbara	F	115.3	98.0	115.0	115.15	0.3

(Test 1) Plot dataset renamed1

VARLIST() in proc SGPLOT, XAXIS statement: Retrieve the original label of variable MeanAB in dataset BlandAltman and assign it unchanged to the X axis.

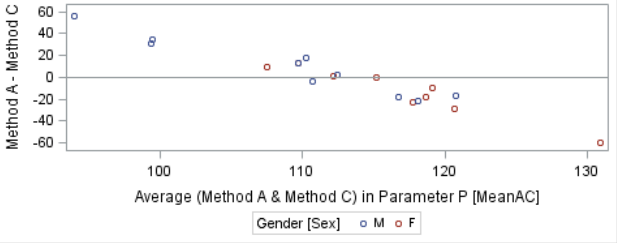
VARLIST() in proc SGPLOT, YAXIS statement: Retrieve the label from variable DiffAB in dataset renamed1, shorten it to only the text between parentheses, and assign it to the Y axis.



(Test 2) Plot dataset renamed2

VARLIST() in proc SGPLOT, XAXIS statement: Retrieve the original label of variable MeanAC in dataset renamed1 and assign it unchanged to the X axis.

VARLIST() in proc SGPLOT, YAXIS statement: Retrieve the label from variable DiffAC in dataset renamed2, shorten it to only the text between parentheses, and assign it to the Y axis.



Log Excerpts of Part (B):

```
2833   %test(data=BlandAltman, x=MeanAB, y=DiffAB, group=Sex, num=1);
MPRINT(TEST):   data renamed1(drop
MPRINT(TEST):  = X index=(Sex) rename=(MeanAB=X DiffAB=Y));
MPRINT(TEST):   set BlandAltman;
MPRINT(TEST):   *- Replace label of variable Y by some pre-specified text (for illustration) -*;
MPRINT(TEST):   label DiffAB = "Variable Y [DiffAB]";
MPRINT(TEST):   label MeanAB=
MPRINT(TEST):   "Average (Method A & Method B) +|+in Parameter P [MeanAB]";
MPRINT(TEST):   run;
```

```
MPRINT(TEST):   proc print data=renamed1(obs=3) label split="|" noobs;
MPRINT(TEST):   title3 "Varlist() in proc step, label statement: Modify label of variable DiffAB (renamed as
Y):replace parentheses by split characters '|', and append '***' ";
MPRINT(TEST):   label Y=
MPRINT(TEST):   "Difference |Method A - Method B| in Parameter P [DiffAB] ***";
MPRINT(TEST):   run;
```

```
MPRINT(TEST):   title "(Test 1) Plot dataset renamed1";
MPRINT(TEST):   proc sgplot data=renamed1;
MPRINT(TEST):   scatter x=X y=Y / group=Sex;
MPRINT(TEST):   refline 0 / axis=Y;
MPRINT(TEST):   title2 "VARLIST() in proc SGPLOT, XAXIS statement: Retrieve the original label of variableMeanAB in
dataset BlandAltman and assign it unchanged to the X axis." ;
MPRINT(TEST):   xaxis label=
MPRINT(VARLIST):  "Average (Method A & Method B) in Parameter P [MeanAB]"
MPRINT(TEST):  ;
MPRINT(TEST):   title3 "VARLIST() in proc SGPLOT, YAXIS statement: Retrieve the label from variable DiffAB indataset
renamed1, shorten it to only the text between parentheses, and
assign it to the Yaxis." ;
MPRINT(TEST):   yaxis label=
MPRINT(TEST):  "Method A - Method B";
MPRINT(TEST):   run;
```

Code part (C) – generating histograms plots, and storing histogram data for later use:

```
%macro histogr(data=, var=, group=, binstart=, binwidth=, num=);
```

4) %VARLIST() is used in %if statement within a macro-definition to conditionally execute a different piece of code depending whether a variable named &var (var=&var) exists in dataset &data (data=&data) and is not part of the list of character variables (#not# #char#) retrieved from that dataset.

```
  %if (%length(%varlist(data=&data, var=&var #not# #char#))=0) %then %do;
    %put %str(WAR)NING: (histogr): <<<< Variable &var not found in dataset &data or not numeric,
cannot generate histogram. >>>>;
  %end; %else %do;
   *- Generate histogram and create output dataset -*;
   ods output sgplot=&var._&num;
   title "(Histogr &num &var) SGPLOT Histogram of variable &var. from dataset &data";
   proc sgplot data=&data;
     by &group;
     histogram &var / scale=count binstart=&binstart binwidth=&binwidth;
   run;
   *- Show generated dataset -*;
   title "(Histogr &num &var) Print Histogram Data: dataset &var._&num (showing variable names -
only 3 obs are shown)";
   title2 "Note the long variable names automatically generated by PROC SGPLOT.";
   proc print data=&var._&num(obs=3);
   run;
   *- Rename variables with very long, SAS-generated names into shorter ones. -*;
   data &var.Bins&num;
```

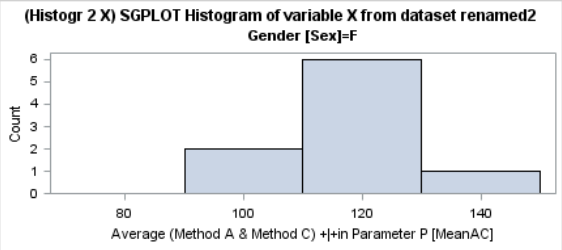
5) %VARLIST() is used in the RENAME (input) dataset option of a DATA step to retrieve the exact names of variables in dataset &var._num, that begin with 'Bin_&var._' and end respectively with '_Y' or '_X' (the ':' character matches any sequence of characters except spaces); those variables are then renamed respectively as 'Count' and as '&var.Bin'.

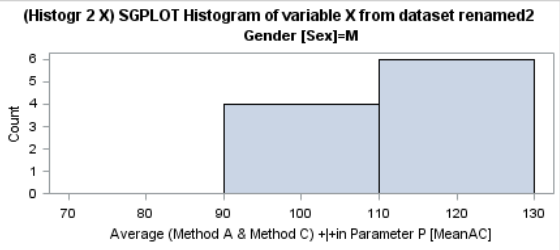
```
     set &var._&num (rename=(%varlist(data=&var._&num, var=Bin_&var._:_Y)=Count
                             %varlist(data=&var._&num, var=Bin_&var._:_X)=&var.Bin));
   run;
   *- Show renamed dataset -*;
   title "(Histogr &num &var) Print Histogram Data: dataset &var.Bins&num (showing variable
names)";
   proc print data=&var.Bins&num(drop=&var where=(Count>0));
   run;
  %end;
%mend histogr;

%histogr(data=renamed1, var=X, group=Sex, binstart=80, binwidth=20, num=1);
%histogr(data=renamed1, var=Y, group=Sex, binstart=-25, binwidth=10, num=1);
%histogr(data=renamed2, var=X, group=Sex, binstart=80, binwidth=20, num=2);
%histogr(data=renamed2, var=Y, group=Sex, binstart=-50, binwidth=20, num=2);

*- test basic error checking in macro HISTOGR -*;
%histogr(data=sashelp.class, var=Teacher, group=Sex, binstart=-50, binwidth=20, num=2);
%histogr(data=sashelp.class, var=Name, group=Sex, binstart=-50, binwidth=20, num=2);
```

Output Excerpts of Part (C):





(Histogr 2 X) Print Histogram Data: dataset X_2 (only 3 obs are shown)
Note the long variable names automatically generated by PROC SGPLOT.

Obs	Sex	BIN_X_SCALE_count_BINSTART_80__X	BIN_X_SCALE_count_BINSTART_80__Y	X
1	F	80	0	117.75
2	F	100	2	115.15
3	F	120	6	112.15

(Histogr 2 X) Print Histogram Data: dataset XBins2 (showing variable names)

Obs	Sex	XBin	Count
2	F	100	2
3	F	120	6
4	F	140	1
11	M	100	4
12	M	120	6

Log Excerpts of Part (C):

```
MPRINT(HISTOGR):   *- Rename variables with very long, SAS-generated names into shorter ones. The generated variable
names may vary, so they are identified by matching a specific
pattern. -*;
MPRINT(HISTOGR):   data XBins2;
MPRINT(HISTOGR):   set X_2 (rename
MPRINT(HISTOGR):  =(BIN_X_SCALE_count_BINSTART_80__Y=Count BIN_X_SCALE_count_BINSTART_80__X=XBin));
MPRINT(HISTOGR):   run;
```

```
1453  *- test basic error checking in macro HISTOGR -*;
1454  %histogr(data=sashelp.class, var=Teacher, group=Sex, binstart=-50, binwidth=20, num=2);
WARNING: (histogr): <<<< Variable Teacher not found in dataset sashelp.class or not numeric, cannot generate
histogram. >>>>
1455  %histogr(data=sashelp.class, var=Name, group=Sex, binstart=-50, binwidth=20, num=2);
WARNING: (histogr): <<<< Variable Name not found in dataset sashelp.class or not numeric, cannot generate histogram.
>>>>
```

CONCLUSION

Although initially presented as a tool to help generate PROC SQL JOINs of relatively high complexity, the macro-function %VARLIST() was shown to have been used in much more diverse contexts, including macros, DATA step, PROC PRINT and PROC SGPLOT.

Its uses range from very simple ones (e.g., insert commas between variable names, retrieve text from labels of existing variables) to more complex ones (e.g., generate lists of variable names found in one or more datasets but not others, excluding variables of a specific type or in an exclusion list, adding dataset alias prefix and separators for use in SELECT clause of SQL JOINs).

The author believes %VARLIST() constitutes a useful tool to add to all experienced SAS Base programmers' toolboxes. Moreover it can still be enriched to meet additional needs.

REFERENCES

Bodart, Jean-Michel. "Dynamic, Powerful, User-friendly SQL JOINS Made Easy with %VARLIST, a Generic SAS Macro Function", PhUSE 2015, CC05.

SAS Institute, "SAS 9.3 Macro Language: Reference"

SAS Institute, "SAS 9.3 Functions and Call Routines: Reference"

SAS Institute, "SAS 9.3 SQL Procedure: User's Guide"

RECOMMENDED READING

The full macro code (and additional examples of use) are freely available at
http://www.phusewiki.org/wiki/index.php?title=SAS_macro-function_%25VARLIST

CONTACT INFORMATION

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

Jean-Michel Bodart
Business & Decision Life Sciences
Rue Saint-Lambert 141
1200 Brussels (Belgium)
Work Phone: +32 (0)2 774 11 00
Fax: +32 (0)2 774 11 99
Email: jean-michel.bodart@businessdecision.com
Web: www.businessdecision-lifesciences.com