Paper BtB-18

List Processing with SAS: A Comprehensive Survey

Jiangtang Hu
d-Wise Technologies, Inc., Morrisville, NC

ABSTRACT

In SAS, a list is not the data structure well known in other general programming languages like Python; it's rather a series of values to facilitate data driven programming. This paper takes a comprehensive survey on SAS list processing techniques (function like macros), from basic operations (create, sort, insert, delete, search, quote, reverse, slice, zip and etc.) to real life applications. Sources were taken from varies of papers and online macro repositories by SAS gurus Roland Rashleigh-Berry, Ian Whitlock, Robert J. Morris, Richard A. DeVenezia and Chang Chung. The author also contributes some macros (like how to slice a list) to fill the holes in this big picture. All codes were hosted in github.com (https://github.com/Jiangtang/SAS_ListProcessing) and comments, forks are welcome (don't reinvent wheels!).

INTRODUCTION

List is by definition a collection of data which is widely used as an important data structure in bunch of general programming languages like Lisp, Python, Java. In SAS, since there is no corresponding concept of list strictly speaking, we can't play around list like the other programming languages above. But similar with array in SAS which is not a data structure but a collection of variables, list as a data container can also serve as a versatile programming tool, especially for data driven programming.

Usually we throw multiple macro variables (&&var&i, the metadata, or control data, such as distinct) into a loop for batch processing in so called data driven programming. List can hold such metadata in a single macro variable (macro list) with series of values rather than series of macro variables. Usually it will reduce the consumption of system resource.

Lots of SAS programmers implemented and distributed such list programming techniques for years in user conferences (see References session). Also, even not presented in meeting, lots of code snippets on list processing were scattered in blogs, personal websites and discussion forums. I once wrote a function like macro (recursive version) to create a list of sequence:

```
%macro _list(n,pre=ff);
    %if &n=1 %then &pre.1;
    %else %_list(%eval(&n-1)),&pre.&n;
%mend _list;
%put %_list(3); *produces ff1, ff2, ff3;
```

But when I read one of Ian Whitlock's papers, *Names, Names, Names – Make Me a List* (SGF 2007, SESUG 2008), I say: stop! I'm gonna use Ian's %range and I create Github page to hold it (with minimum modifications due to personal preference).

This paper serves as a display of my personal collection of SAS list techniques which is hosted in Github:

https://github.com/Jiangtang/SAS_ListProcessing

A. SELECITON CRITERION

There are lots of SAS techniques and tricks to create and manipulate lists. For generalization and portal purpose, I only select such function-like macros to my repository, which means these macros all have return values, rather than generated outputs like datasets, listings or logs.

B. REPOSOTRY OVERVIEW

The followings are the complete macro functions for list processing I collected so far, including

- List creating
- Formatting
- Quoting
- Counting
- Sorting
- Slicing
- Zipping
- Join
- Adding
- Appending
- Replacing
- Renaming
- Editing
- Dropping
- Matching
- Reversing

For the original authors, please check the individual code files. You can call it in your program via:

```
filename list url
"https://raw.github.com/Jiangtang/SAS_ListProcessing/master/_ListProcessing";
%inc list;
```

LIST CREATING

- %range: produces a sequence like 1 2 3 or f1 f2 f3 or 1a 2a 3a
- %range_non_int: increment a macro do loop by a non-integer value
- %suffix_counter: Create a list of variable names formed by adding a numeric counter suffix to a base name.
- %getVar: get all variables (N, C or all) from a dataset
- %qreadpipe: read the output of a system command
- %dir: return a list of members of a directory
- %dirfpq: return a list of full-path quoted of members of a directory

LIST FORMATING

- %changesep: change the separator for a list
- %seplist: Emit a list of items separated by some delimiter
- %splitmac: insert split characters in a macro string
- %capmac: capitalise the first letter of each word in a macro string

QUOTING

- %qt: add quotes to each element in a list
- %quotelst: quote the elements of a list
- %upt: remove quotes from each element of a list
- %qdequote: remove front and end matching quotes from a macro string
- %dequote: remove front and end matching quotes from a macro string
- %noquotes: remove all quoted strings from a macro expression
- %quotecnt: count quoted strings in a macro expression
- %quotescan: scan for a quoted string in a macro expression

LIST PROPERTIES

- %num_tokens: Count the number of “tokens” (variables) in a list.
- %countW: Retrieve the number of words in a macro variable
- %words: return the number of words in a string
- %windex: return the word count position in a string

LIST MANIPULATION

- %slice: return a sub-list sliced by a index
- %zip: zips two lists together by joining correponding elements, see, a b and c d ==> ac bd
- %parallel_join: Join two variable lists by connecting each variable in the first list to its correspondingvariable in the second list
- %add_string: Add a text string to each variable in a list as either a prefix or suffix
- %xprod: take cross product of two lists, see, a b and c d ==> ac ad bc bd
- %appmvar: append a string onto an existing macro variable
- %prefix: return a list with a prefix added
- %suffix: return a list with a suffix added
- %replace: replace symbolic variable in block of code with each element of a list, see, a b and code = #=# **==> a=**a b=__b
- %rename_string: Create a list suitable for the rename statement
- %editlist: edit a list of space delimited items
- %nodup: drop duplicates in a space-delimited list
- %match: return elements of a list that match those in a reference list
- %remove: remove all occurrences of the target string(s) from another string
- %removew: remove all occurrences of the target word(s) from a source list of words.
- %reverse: Reverse a macro variable's value (use %sysfunc(reverse) since v6.12)
- %MLSORT: sort a macro list

C. APPLICATION 1 AND HOW THESE LIST MACRO FUNCTIONS ARE DIFFERENT

To apply data driven programming technique, we need to fetch metadata from source data dynamically, for example, to get all variables from an input dataset.

You can easily make it by

1. PROC SQL, from a SAS dictionary table, or macro variable by SELECT INTO; or

2. Proc Contents; or

3. even a smart data step with RESOLVE function and with CALL SYMPUT, like

```
%let namelist=;
data _null_;
      set sashelp.class;
      call symput('namelist', trim(resolve('&namelist'))||' '||trim(name));
run;
%put &namelist;
```

In the repository, there is an elegant function-like macro where SAS file processing functions like open(), close() are used, %getVar. Below follows examples to fetch variables from a dataset, based on variable type, numeric or character:

```
%put %getVar(%str(sashelp.class));
%put %getVar(%str(sashelp.class),n);
%put %getVar(%str(sashelp.class),C);
```

Outputs:

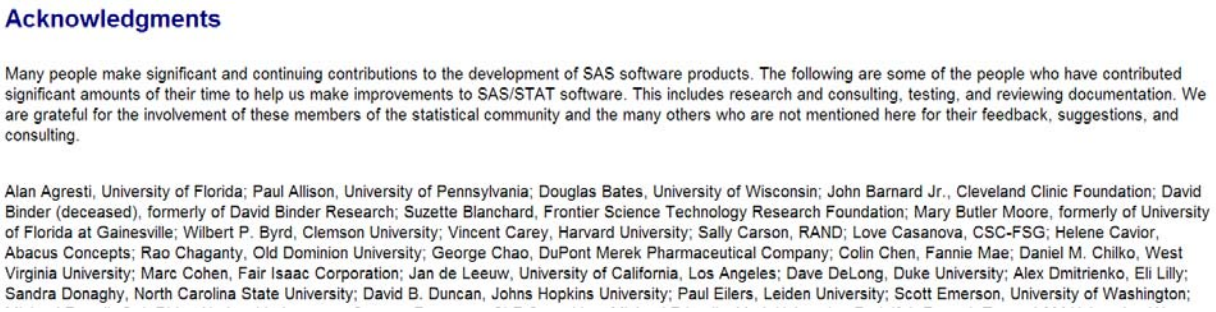
```
Name Sex Age Height Weight
Age Height Weight
Name Sex
```

The benefits of these macro functions are:

- have more control options
- have return value which can be seamlessly embedded into your codes just like any other functions
- usually consume less system resource because only one macro variable needed in do loop. In other dynamic macro programming techniques, you always create bunch of macro variables (&&var&i) to hold the control values. For example, you should create 100 macro variables to compete a macro list with 100 elements.

D. APPLICATION 2 AND THE LESSONS

There are lots of non-SAS employees contributing to the development of SAS/STAT software. Their names were list in the acknowledgements session of the SAS/STAT documentation.



Acknowledgments

Many people make significant and continuing contributions to the development of SAS software products. The following are some of the people who have contributed significant amounts of their time to help us make improvements to SAS/STAT software. This includes research and consulting, testing, and reviewing documentation. We are grateful for the involvement of these members of the statistical community and the many others who are not mentioned here for their feedback, suggestions, and consulting.

Alan Agresti, University of Florida; Paul Allison, University of Pennsylvania; Douglas Bates, University of Wisconsin; John Barnard Jr., Cleveland Clinic Foundation; David Binder (deceased), formerly of David Binder Research; Suzette Blanchard, Frontier Science Technology Research Foundation; Mary Butler Moore, formerly of University of Florida at Gainesville; Wilbert P. Byrd, Clemson University; Vincent Carey, Harvard University; Sally Carson, RAND; Love Casanova, CSC-FSG; Helene Cavior, Abacus Concepts; Rao Chaganty, Old Dominion University; George Chao, DuPont Merek Pharmaceutical Company; Colin Chen, Fannie Mae; Daniel M. Chilko, West Virginia University; Marc Cohen, Fair Isaac Corporation; Jan de Leeuw, University of California, Los Angeles; Dave DeLong, Duke University; Alex Dmitrienko, Eli Lilly; Sandra Donaghy, North Carolina State University; David B. Duncan, Johns Hopkins University; Paul Eilers, Leiden University; Scott Emerson, University of Washington;

Since the name list was not in tabulate format, it is hardly to read. To get a table version of all the contributors and their institutions, you can copy all the free text into a macro variable,

```
%let name=%nrstr(Alan Agresti, University of Florida; Paul Allison…….(and more)
```

Then you need to count the number of names in the list. In our macro repository, there are 3 macro functions used to count the list elements: %countW, %num_tokens and %words:

```
%let n3= %countw(&name);
%put &n3;

%let n2= %num_tokens(&name, delim=%str(;));
%put &n2;

%let n1= %words(&name, delim=%str(;));
%put &n1;
```

The output:

```
%countW: 711
WARNING: Apparent symbolic reference M not resolved.
WARNING: Apparent symbolic reference M not resolved.
WARNING: Apparent symbolic reference M not resolved.
%num_tokens: 127
%words: 127
```

%countW produced the wrong number since there is no delimiter options available in this macro. %num_tokens returned the right number but with warning (it's because there are some contributors from Texas A&M University and

%num_tokens doesn't mask the special symbol, &). %words is the good choice for this case and we use it to move on using the standard loop based on the counting:

```
%macro doit(delim=%str(;));
data name;
       length a $100.;
       %do i=1 %to &n1;
              a="%qscan(%SUPERQ(name),&i,&delim)";
              name=scan(a,1,",");
              institute=substr(a,length(name)+2);
              output;
       %end;
       drop a;
run;
proc print data=name;run;
%mend;
%doit;
```

Then we just get the decent tabulate output (partial):

Obs	name	institute
1	Alan Agresti	University of Florida
2	Paul Allison	University of Pennsylvania
3	Douglas Bates	University of Wisconsin
4	John Barnard Jr.	Cleveland Clinic Foundation
5	David Binder (deceased)	formerly of David Binder Research
6	Suzette Blanchard	Frontier Science Technology Research Foundation
7	Mary Butler Moore	formerly of University of Florida at Gainesville
8	Wilbert P. Byrd	Clemson University
9	Vincent Carey	Harvard University

(It's pretty pressure to read this tabulate report and at least I got a bonus that I found my colleague Christopher Olinger is on the list!)

It's a good idea to check the repository if there is any macros function available for your need. Sometimes you will find more than one fit your need and what you need to do is just to test them and pick up one best for you!

Also, after checking the different macro functions for the same task, you will get good command of how these macro works and how they differentiate from each other. For example, the different behavior between %words and %num_tokens is due to that the former uses a macro function %qscan which can mask the special symbols (Q functions), and the latter, %scan.

At last, if you feel comfortable, you can modify the existing macros to get what you want. For example, you can add a delimiter parameter to %countW to produce the right number following the same approach from %words and/or %num_tokens.

E. DO YOUR SELF: WRITE A MACRO FUNCTION FOR LIST PROCESSING

After reviewing my collection of the macros, I found there was at least one important operation missing and I'd produce one. This is the %slice, to slice a list, or return a sub-list sliced by an index:

```
%macro slice(
    L,              /*list*/
    i,              /*index*/
    sep_L=%str( ),/*separator for list*/
    sep_i=%str( ) /*separator for index*/
    );
```

```
    %let VarList = ;
    %let count=%sysfunc(countw(&i,&sep_i));

    %do j = 1 %to &count;
      %let index=%qscan(&i,&j,&sep_i);
      %let VarList = &VarList.%str( )%qscan(&L,&index,&sep_L);
    %end;

    &VarList
%mend slice;
```

The followings all produce `a c d`:

```
%put %slice(a b c d,1 3  4);
%put %slice(%str(a, b, c, d),1 3  4,sep_L=%str(,));
%put %slice(%str(a, b, c, d),%str(1, 3, 4),sep_L=%str(,),sep_i=%str(,));
```

To write a function like macro:

- review the repository of all macros available, do not reinvent the wheels.
- learning the basic program style by existing macros
- reuse the functions like counting and scanning
- have and only have return value in your macros
- be careful for the quoting!

F. FINAL ADVICE: DO NOT OVERKILL

Suppose you have 10 datasets, literally ds1, ds2, …ds10 and you need to concatenate them all:

```
data a;
      set a1 a2 a3 a4 a5 a6 a7 a8 a9 a10;
run;
```

If such list members were generated dynamically, you will probably come out a macro solution:

```
%macro doit;
    data a;
      set %do i=1 %to &n; a&i %end; ;
    run;
%mend;
%doit
```

where &n =10. After learning the list repository from this paper, you can modify it as

```
set %range(to=10,opre=a);
```

But wait, you should first get a quick shortcut

```
set a1-a10;
set a:;
```

Use when it is needed. Also, suppose we need to fill out a list of datasets in a FROM clause of a SQL procedure. You cannot copy the %do loop above because all elements should be separated by a comma and you can't use the shortcuts, then you need to think about to take a look at the repository for a quick implementation (%range).

CONCLUSION

SAS list processing is pretty versatile technique in data drive programming. Fortunately, there are already lots of macros available online distributed separated by SAS users. This paper introduces the author's collation of macro functions for listing processing which is archived and categorized in Github. Advices for using and creating such function like macros also provided.

REFERENCES

[1] *Text Utility Macros for Manipulating Lists of Variable Names* by Robert J. Morris at SUGI 30 (2005)

[2] *List Processing - Make Light Work of List Processing in SAS* by Peter Crawford at SUGI 31 (2006)

[3] *Names, Names, Names - Make Me a List* by Ian Whitlock at SGF 2007

[4] List Processing Basics: Creating and Using Lists of Macro Variables by Ronald J. Fehd/ Art Carpenter at SGF 2007

[5] *Choosing the Best Way to Store and Manipulate Lists* in SAS by Dmitry Rozhetskin at WUSS 2010

ACKNOWLEDGMENTS

Very appreciate the varies of SAS users to contribute the list processing techniques.

CONTACT INFORMATION

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

Jiangtang Hu
Life Sciences Consultant
d-Wise Technologies, Inc.
1500 Perimeter Park Dr., Suite 150, Morrisville, NC, 27560
www.d-Wise.com

(O) 919-334-6096
(C) 919-801-9659
(F): 888-563-0931
(O) Jiangtang.Hu@d-Wise.com
(P) jiangtanghu@gmail.com
http://jiangtanghu.com/