Do We Need Macros?

R.J.Fehd

30+ yrs exp., 40+ titles, 90+ papers

	macro maven	senior maverick
memes:	**cardinality ratio**	**list processing**
papers:	**companions:** autoexec, configuration, sysparm	
lp macros:	**call-macro**	**call-text** **date-loop**
expertise:	**list processing**	advanced data step
	macros	test-driven development
	sql	scl

Do we need Macros? An Essay on the Theory of Application Development

Ronald J. Fehd, SAS-L's Macro Maven,
senior maverick, theoretical programmer,
Fragile-Free Software Institute

BASUG 2022-April-13 or April-31?

Learning a New Computer Language

- variables
- conditions
- loops
- functions
- syntax

Terms used here

function	returns a value; in SAS software a value is a token, and is less than a statement
module	calls routines and subroutines to process input and produce output; often called 'main'
program	a set of statements subprogram: a subset of a program
routine	a program or subprogram; performs one or more tasks, calls subroutines
subroutine	performs a single task, called by modules or routines

Programs, Subprograms, Steps

programs contain subprograms		HIPO: hierarchical input process output
SAS programs contain steps		data, proc, run
steps have two aspects	compile	data structure
	execute	algorithm produces result

The layers of a program

the varieties of program experience

- assignments to Global Symbol Table:
  filename, libname, options, title
- %include statement
  subprogram
- macro language compile variables: `%let mvar=...;`
  definitions: `%macro ... %mend;`
  execution variable references: `&mvar`
  macro calls: `%do_this(data=...)`
- SAS statements

The Global Symbol Table

- environment variables `sasv9.cfg`
- location names: filerefs and librefs used in options
- macro
  - variables: system- or user-defined
  - definitions: location of compiled code
- options reference location names for reuse
- running text: titles, footnotes

Optimization issues

- autocall: automatic search for reusable macros
- compiled and stored: macro definitions saved in catalog
- testing: unit and integration
  - options for debugging
  - remote control during testing

ApDev Strategy

- large — table-top rule: 10 pages, 50 lines/page = 500 lines

1	2	3	4
5	6	7	8
9			10

- reuse: — used often, compiled once
- centralization
- guarantee
- hide complexity: — simplification, standardization

ApDev Tactics: macro language

	syntax	
variables :	`%let mvar =`	pass values across steps
conditions:	`%if`	add or skip code
		branch
loops, iterative:	`%do i = 1 %to`	...
functions:	`%eval`	integer arithemetic
	`%sysevalf`	real numbers
	`%sysfunc`	access data step functions

Hard-code, find 2 examples

```
1  proc freq data = sashelp.class;
2            tables age;
```

```
1  proc freq data = sashelp.shoes;
2            tables region;
```

soft-coded program with parameters

```
1  %let data = sashelp.shoes;
2  %let var  = region;
3  proc freq data = &data;
4            tables &var;
```

split into two programs, caller and sub-program

```
1  *name: sub-program-1-test.sas
2  %let data = sashelp.class;
3  %let var  = sex;
4  *let data = sashelp.shoes;
5  *let var  = region;
6  %include 'sub-program-1.sas';
```

```
1  *name: sub-program-1.sas;
2  *let data = sashelp.shoes;
3  *let var  = region;
4  %put echo: &=data &=var;
5  proc freq data   = &data;
6            tables   &var    / noprint
7            out    = out_freq;
8  run;
9  %put trace: sub-program-1 ending;
```

---

make into macro, the test program

```
1  *name: sub-program-2-test.sas;
2  options mprint source2;
3  %sub_program_2(data = sashelp.class
4                 ,var  = sex)
```

make into macro

```
1  *name: sub_program_2.sas;
2  %macro sub_program_2
3        (data     = sashelp.shoes
4        ,var      = region
5        ,out_data = out_freq
6        ,testing  = 0);
7  %let testing = %eval(not(0 eq &testing)
8       or %sysfunc(getoption(mprint)) eq MPRINT);
9  %if  &testing %then %put _local_;
10 proc freq data   = &data;
11           tables   &var     / noprint
12           out    = &out_data;
13 run;
14 %if &testing %then %do;
15     proc sql; describe table &syslast;  quit;
16     %end;
17 %mend sub_program_2;
```

Do we need %includes or macros to reuse programs?

- autoexec needed for either, for location names of folders
- use %includes with macro variables as parameters
  until you need:
  - additional code within subprogram
  - macro functions or loops
- macro language
  - variables — passing values across step boundaries
  - autocall — need *filerefs* for options
  - compiled and stored — need *librefs* for options

Author Information

Ronald J. Fehd Ron.Fehd.macro.maven@gmail.com
Atlanta, GA, USA