Ron Fehd, SAS-L’s Macro Maven, Answers Your Macro Questions
Ronald J. Fehd, Stakana Analytics

**Abstract**

**Description:** SAS® software consists of two languages, SAS and its macro language.

**Purpose:** The purpose of this talk is to provide overview and perspective of how SAS works and how the macro language can work both within and before SAS program statements and steps.

**Audience:** all levels

**Keywords:** macros, autoexec, conditionals, configuration, compile, execution, loops

---

**In this paper**

- **Introduction**
- **How SAS works**
- **Conditionals**
- **Loops: List Processing**
- **Summary**
- **References**

---

**Introduction**

**Overview**

These slides discuss my overview of programming and software design.

**what is Information?**

Information is the difference

Gregory Bateson, 1904–1980
Steps to an Ecology of Mind, 1972

Vilfredo Pareto, 1896
80/20 rule
80% of effects come from 20% of the causes
vital few and useful many
popularized by Joseph Juran, 1941
aspects of learning computer language

- variables
- constants
- scope: global or local symbol table

- conditions
- if... then... else...
- stop
- comparisons: \(lt, le, eq, ge, gt\)
- additional code
- logical operators: \(and, or, not\)

- loops
- conditional exit
- enumerate: iterative
- while, leave, until
- start, stop, \(step=n\)
- sequence: \(step eq 1\)
- series: \(step ne 1\)

- itemize: list of items

- functions
- macro, method, process, or procedure

HIPO, three aspects of every program

<table>
<thead>
<tr>
<th>Hierarchical</th>
<th>Input</th>
<th>Process</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>ingredients</td>
<td>algorithm</td>
<td>name of recipe</td>
</tr>
<tr>
<td>top</td>
<td>module</td>
<td>process</td>
<td></td>
</tr>
<tr>
<td>middle</td>
<td>routine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>subroutine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>executed:</td>
<td>always</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N steps</td>
<td>Q: ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>else</td>
</tr>
<tr>
<td></td>
<td></td>
<td>executed:</td>
<td>always</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>conditionally</td>
</tr>
</tbody>
</table>

data structure or algorithm?

- 20% compiled
- data structure
- object
- attribute statement(s)
- less than 1% of program
  \((20\%)^3 \approx 0.8\%\)

slide 3

slide 4

slide 5
How SAS works

startup process

1. sas.exe
2. sasv9.cfg allocate environment variables assign options
3. command-line options *.cfg sysparm
4. autoexec add or change entries in global symbol table
5. initstmt initial statements
6. program
7. termstmt termination statements

autoexec process

adding entries to global symbol table Fehd [11]

environment variable(s) site_root filenames project, site_includes, site_macros libname library options libname=(project site _macros sasautos) running text: titles footnotes macro variables definitions

Conditionals

Conditionally execute global statements using sysfunc and ifc, [3]
True is not false, [14]

Truth table [2]
Writing testing-aware programs, [1]

if fail then stop

%let data = sashelp.class;
%sysfunc(ifc(%sysfunc(exist(&data))
, %nrstr(%put info: exist &=data;)
, %nrstr(%put fail: not exist &=data;
endsas;
)
)

additional statements

DATA process_or_procedure;
attrib ...; *<----- data structure;
*...; *<----- algorithm
run;* update syslast;
*** echo data structure to log?
%sysfunc(ifc(%sysfunc(getoption(source2)) eq SOURCE2
, %nrstr(proc sql;
   describe table &syslast;
   quit;), ))
%macro demo(data = sashelp.class ,testing = 0);
%let testing=%eval( not(0 eq &testing)
or %sysfunc(getoption(mprint)) eq MPRINT);
*...;
%if &testing %then %do;
proc sql; describe table &syslast;
quit;
%end;

Note: the condition zero not equal &testing changes any value to true

Loops: List Processing

loops: list processing
list contains items sets \( n = \text{cardinality} \)
arrays contain elements \( n = \text{dimension} \)
an item can be an attribute, one variable with information or a list
a list does not contain data, i.e. summable numeric facts
each row contains a set of parameters
other terms: data- or table-driven dynamic programs
other terms: self-modifying

list processing tools
cx-include: call execute a parameterized include [8]
macro callmacro [10]
macro calltext [13]
macro dataloop [9]

using macro loops in functions
%let macro_name = proc_freq;
%let list = a bb ccc;
*count word(s) delimited by space;
%let n_items = %sysfunc(countw(&list,%str( )));

%do i = 1 %to &n_items;
%let item = %scan(&list,&i);
%put echo: &=item;
%&macro_name(item=&item) %* know semicolon!
%end;

Note: *count delimiters and add one;
%let n_items = %eval(%sysfunc(countc(&list,%str( )))+1);

issues when using macros

• autocall
  autoexec: filename macros '...';
  options sasautos = (macros sasautos);

• macro functions
  return tokens
  less than statement

• quoting
  %let x = %nrstr(...);
  %put %unquote(&x);

• suffix of reference is dot: &mvar. format[length.
  &libname..memname
  &filename..txt

reasons to use macros

• variables: programs have access to global symbol table
• control: add statements or branching
• loops: write macro definitions as functions

Notes: 
Do we need macros? [12]
Macro design ideas [7]

Summary

Conclusion

Using SAS software effectively requires gaining an understanding of software design issues.

Author Information

Ronald J. Fehd
Ron.Fehd.macro.maven@gmail.com
sco.wiki http://www.sascommunity.org/wiki/Ronald_J._Fehd
LinkedIn www.linkedin.com/Ronald.Fehd
affiliation Stakana Analytics, Senior Maverick
also known as macro maven on SAS-L, Theoretical Programmer
Programs: http://www.sascommunity.org/wiki/
Macro_CallMacro Fehd [5]
Macro_CallText [4]
Macro_DateLoop [6]
Routine_Call-Exec-Include [8]
Conditionally_Execute_Global_Statements [3]
List_Processing_Basics Fehd and Carpenter [15]
References


[13] Ronald J. Fehd. List processing macro call-text. In MidWest SAS Users Group Annual Conference Proceedings, 2016. Tools of Trade, 10 pp.; using %sysfunc with SCL functions to read a list, a control data set, and for each observation, call a macro with variable names and values as named parameters.


Hands On Workshop, 20 pp.; comparison of methods: making and iterating macro arrays, scanning macro variable, writing calls to macro variable, write to file then include, call execute; using macro function nrstr in call execute argument; 11 examples, bibliography.