Efficiency Programming with Macro Variable Arrays

Veronica Renauldo is an Associate Statistical Programmer at QST Consultations, LTD located in Allendale, MI. She is involved with data cleaning and data set/output production and quality control in numerous clinical trials, and works on both CDISC and non-CDISC pharmaceutical and health care device studies. She also assists in training new employees for the company. Veronica received a Bachelor’s degree in Statistics with a minor in Mathematics followed by a Master’s degree in Biostatistics in April of 2014 and 2016, respectively, from Grand Valley State University. She is a SAS certified Base SAS 9 programmer and has been using SAS for 7 years. Working in a fast paced contract research organization where efficiency and reproducibility are key in most aspects of her work, Veronica has developed strong macro processing skills that can be employed in a wide array of statistical inference and data set manipulation.
Outline

- Data step Arrays
- Macro Variables
- Macro Functions
- Array of Macro variables
- Macro Variable Arrays
  - Definition
  - Ways to create a macro variable array using examples
  - Advanced Macro Variable Array Examples
- Questions
Data Step Arrays

- Increases efficiency
- Applies logic to several variables of the same type
- Use looping structures
- The dimension of an array: explicitly defined or use of an asterisk

```plaintext
data one;
    array numz (10);
    array chrz (10) $200;
    do i = 1 to dim(numz);
        numz(i) = i;
        chrz(i) = 'Character Version '||strip(put(i, 2.));
    end;
run;
```
Macro Variables

- Character strings used for symbolic substitutions within SAS code
  - Numeric macro variables are technically character
    - `%let popn = 160;`
  - Maximum length = 65,534 characters.
  - Calls prefixed with at least one & compile with a period
    - `&popn`.
  - Have a constant value that is set in two different ways
    - Automatically by SAS
    - User defined
Automatic Macro Variables

- Macro variables that are set when a SAS product is deployed.
- `%PUT _ALL_;` will show all macro variables (automatic and user defined) in the log
- `SYSDATE =` date at which the SAS produced was invoked in the DATE7 format
User Defined Macro Variables

- Created by the user
- Anything the user deems worthy of being substituted
- Examples
  - Footnotes that will be used for multiple outputs
    \[
    \texttt{\%let fnote1 = Observed Data Only;}
    \]
  - Population N value that will be used in several calculations
    \[
    \texttt{\%let N = 68;}
    \]
  - Treatment arm n values needed for calculations
    \[
    \texttt{\%let N1 = 24;}
    \quad \texttt{\%let N2 = 44;}
    \]
Macro Functions
Macro Functions

• Like data step variable functions but on macros
• Different syntax than data step variable functions but most macro functions require the same format as data step variable functions
• Examples:
  • %UPCASE: up-cases all text within the specified macro variable
  • %LENGTH: returns the text of the specified macro variable
  • %LOWCASE: low-cases all text within specified macro variable
Macro Functions: %BQUOTE

%BQUOTE(string)

- Quoting Function
- Allows a user to mask any operator or special character in a macro variable.
  - Special Characters: \& % ' " ( ) + - * / < > = - ^ ~ ; , blank
  - Operators: AND OR NOT EQ NE LE LT GE GT
- Can handle unmatched pairs of symbols
- Execution time function: Masking occurs during macro execution (mask resolved values of a macro variable)

Ex. %let title = %BQUOTE(Best Graphic Ever;);
Macro Functions: %SCAN

%SCAN(argument, n, <delimiters>)

- Works the same as a DATA step SCAN function
- Searches the argument and returns the \textit{nth} word.
  - A word is one or more characters separated by one or more delimiters.
- Default delimiters: blank . < ( + & ! $ * ) ; ^ - / , % |
Macro Functions: COUNTW and %SYSFUNC

COUNTW(*string*, *chars*, *modifiers*)
- Counts the number of words in a character string

%SYSFUNC(*function*)
- Executes SAS datastep functions or user written functions
- Almost all functions that can be used in a data step can be used on macro variables when using this macro function

<table>
<thead>
<tr>
<th>SAS Functions Not Available with %SYSFUNC and %QSYSFUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Variable Information Functions</td>
</tr>
<tr>
<td>DIF</td>
</tr>
<tr>
<td>IORCMGS</td>
</tr>
<tr>
<td>LBOUND</td>
</tr>
<tr>
<td>LEXPERK</td>
</tr>
<tr>
<td>PUT</td>
</tr>
</tbody>
</table>
Array of Macro Variables
Array of Macro Variables

%let race1 = white;
%let race2 = asian;
%let race3 = other;

%macro print();
  %do i = 1 %to 3;
    %put &&race&i..;
  %end;
%mend;

%print();
Macro Variable Array
Macro Variable Arrays (MVA)

- Singular macro variable that contains an entire array of elements to be processed
- Array elements can be data set names, variable names, variable values, etc.
- Elements are parsed out of the array to be used in the desired macro process

```%let race = white asian other;
%let dats = one two three;
%let varz = weight height temp sysbp diabp pulse;```
What are Macro Variable Arrays Used For?

- Used for repetitive macro processing
- Maximizes the capabilities of macros
- Increases efficiency (reduces programmer errors)
Methods to create a MVA

1. Explicitly Defined
   a. %LET statement
   b. Macro Variable parameter within a Macro
2. Data step with CALL and SYMPUT functions
3. PROC SQL
MVA Creation: %LET Statement

```sas
%let race = white asian other;

%macro print();
  %do i = 1 %to 3;
    %put %scan(&race., &i.);
  %end;
%mend;

%mend;
%print();
```
MVA Creation: %LET Statement

MLOGIC(PRINT): Beginning execution.
MLOGIC(PRINT): %DO loop beginning; index variable i; start value is 1; stop value is 3; by value is 1.
MLOGIC(PRINT): %PUT %scan(&race., &i.)
SYMBOLGEN: Macro variable RACE resolves to white asian other
SYMBOLGEN: Macro variable i resolves to 1
SYMBOLGEN: white
MLOGIC(PRINT): %DO loop index variable i is now 2; loop will iterate again.
MLOGIC(PRINT): %PUT %scan(&race., &i.)
SYMBOLGEN: Macro variable RACE resolves to white asian other
SYMBOLGEN: Macro variable i resolves to 2
SYMBOLGEN: asian
MLOGIC(PRINT): %DO loop index variable i is now 3; loop will iterate again.
MLOGIC(PRINT): %PUT %scan(&race., &i.)
SYMBOLGEN: Macro variable RACE resolves to white asian other
SYMBOLGEN: Macro variable i resolves to 3
SYMBOLGEN: other
MLOGIC(PRINT): %DO loop index variable i is now 4; loop will not iterate again.
MLOGIC(PRINT): Ending execution.
Array of Macro Variables vs Macro Variable Array

Array of Macro Variables

```sas
%let race1 = white;
%let race2 = asian;
%let race3 = other;

%macro print();
  %do i = 1 %to 3;
    %put &&race&i..;
  %end;
%mend;

%print();
```

Macro Variable Array

```sas
%let race = white asian other;

%macro print();
  %do i = 1 %to 3;
    %put %scan(&race., &i.);
  %end;
%mend;

%print();
```
MVA Creation: Macro Variable Parameter

- Macro will read in several permanent SAS data sets and outputs them to the work library for temporary use.

```sas
%macro rin(lib, var, dat);
  %do i = 1 %to %sysfunc(countw(&dat.));
    proc sort data=&lib..%scan(&dat., &i.) out=r%scan(&dat., &i.);
      by &var.;
    run;
  %end;
%mend;
%rin(sashelp, height, class classfit fish gridded heart);
```
MVA Creation: Macro Variable Parameter

MLOGIC(RIN): Beginning execution.
MLOGIC(RIN): Parameter LIB has value sashelp.
MLOGIC(RIN): Parameter VAR has value height.
MLOGIC(RIN): Parameter DAT has value class classfit fish gridded heart.
MLOGIC(RIN): %DO loop beginning; index variable 1; start value is 1; stop value is 5; by value is 1.

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.RCLASS has 19 observations and 5 variables.
NOTE: Compressing data set WORK.RCLASS increased size by 100.00 percent. Compressed is 2 pages; un-compressed would require 1 pages.
NOTE: PROCEDURE SORT used (Total process time):
real time 0.03 seconds
cpu time 0.03 seconds
MVA Creation: Macro Variable Parameter

MLOGIC(RIN): %DO loop index variable l is now 5; loop will iterate again.
MPRINT(RIN): proc sort data=sashelp.heart out=rheart;
MPRINT(RIN): by height;
MPRINT(RIN): run;

NOTE: There were 5209 observations read from the data set SASHHELP.HEART.
NOTE: The data set WORK.RHEART has 5209 observations and 17 variables.
NOTE: Compressing data set WORK.RHEART decreased size by 31.19 percent.
      Compressed is 75 pages; un-compressed would require 109 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time 0.04 seconds
      cpu time 0.04 seconds

MLOGIC(RIN): %DO loop index variable l is now 6; loop will not iterate again.
MLOGIC(RIN): Ending execution.
MVA Creation: CALL SYMPUT

- A method of assigning a value to a macro variable within a DATA step
- Can be employed a regular DATA step or a DATA NULL step
- Objective of example: to obtain frequency data for origin and type for each manufacturer in the SASHELP.CARS dataset
MVA Creation: CALL SYMPUT

Desired output: frequency data for origin and type for each manufacturer in the SASHELP.CARS dataset

data _null_;  
set sashelp.cars end = last;  
by make;  
length brand $5000;  
retain brand;  
if first.make then brand = strip(brand)||'*'||strip(make);  
if last then do;  
   brand = strip(substr(brand, 2));  
   call symput('brands', strip(brand));  
end;  
run;
# MVA Creation: CALL SYMPUT

<table>
<thead>
<tr>
<th>Obs</th>
<th>Make</th>
<th>Model</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Acura</td>
<td>3.5 RL w/Navigation 4dr</td>
<td>*Acura</td>
</tr>
<tr>
<td>8</td>
<td>Audi</td>
<td>A4 1.8T 4dr</td>
<td><em>Acura</em>Audi</td>
</tr>
<tr>
<td>27</td>
<td>BMW</td>
<td>X3 3.0i</td>
<td><em>Acura</em>Audi*BMW</td>
</tr>
</tbody>
</table>
MVA Creation: CALL SYMPUT

Desired output: frequency data for origin and type for each manufacturer in the SASHELP.CARS dataset

There are 38 manufacturers.
%macro frqz();
  %do i = 1 %to %sysfunc(countw(&brands., *));
  title "%scan(&brands., &i., *)";
  proc freq data=sashelp.cars;
    where make = "%scan(&brands., &i., *)";
    tables origin*type/list;
  run;
  title;
  %end;
%end;

%frqz();
MVA Creation: CALL SYMPUT

MLOGIC(FRQZ): Beginning execution.
MLOGIC(FRQZ): %DO loop beginning; index variable i; start value is 1; stop value is 38; by value is 1.
MPRINT(FRQZ): title "Acura";
MPRINT(FRQZ): proc freq data=sashelp.cars;
MPRINT(FRQZ): where make = "Acura";
MPRINT(FRQZ): tables origin*type/list;
MPRINT(FRQZ): run;

NOTE: Writing HTML Body file: sashtml.htm
NOTE: There were 7 observations read from the data set SASHHELP.CARS.
      WHERE make='Acura';
NOTE: PROCEDURE FREQ used (Total process time):
      real time 1.49 seconds
      cpu time 0.71 seconds
MVA Creation: CALL SYMPUT

MLOGIC(FRQZ): %DO loop index variable l is now 19; loop will iterate again.
MPRINT(FRQZ):   title "Land Rover";
MPRINT(FRQZ):   proc freq data=sashelp.cars;
MPRINT(FRQZ):   where make = "Land Rover";
MPRINT(FRQZ):   tables origin*type/list;
MPRINT(FRQZ):   run;

NOTE: There were 3 observations read from the data set SASHHELP.CARS,
      WHERE make='Land Rover';
NOTE: PROCEDURE FREQ used (Total process time):
      real time          0.05 seconds
      cpu time           0.04 seconds
MVA Creation: CALL SYMPUT

MLOGIC(FRQZ): %DO loop index variable I is now 38; loop will iterate again.
MPRINT(FRQZ): title "Volvo";
MPRINT(FRQZ): proc freq data=sashelp.cars;
MPRINT(FRQZ): where make = "Volvo";
MPRINT(FRQZ): tables origin*type/list;
MPRINT(FRQZ): run;

NOTE: There were 12 observations read from the data set SASHHELP.CARS.
    WHERE make='Volvo';
NOTE: PROCEDURE FREQ used (Total process time):
    real time 0.06 seconds
    cpu time 0.04 seconds

MPRINT(FRQZ): title;
MLOGIC(FRQZ): %DO loop index variable I is now 39; loop will not iterate again.
MLOGIC(FRQZ): Ending execution.
MVA Creation: CALL SYMPUT

### Acura

The FREQ Procedure

<table>
<thead>
<tr>
<th>Origin</th>
<th>Type</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia</td>
<td>SUV</td>
<td>1</td>
<td>14.29</td>
<td>1</td>
<td>14.29</td>
</tr>
<tr>
<td>Asia</td>
<td>Sedan</td>
<td>5</td>
<td>71.43</td>
<td>6</td>
<td>85.71</td>
</tr>
<tr>
<td>Asia</td>
<td>Sports</td>
<td>1</td>
<td>14.29</td>
<td>7</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### Volvo

The FREQ Procedure

<table>
<thead>
<tr>
<th>Origin</th>
<th>Type</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe</td>
<td>SUV</td>
<td>1</td>
<td>8.33</td>
<td>1</td>
<td>8.33</td>
</tr>
<tr>
<td>Europe</td>
<td>Sedan</td>
<td>9</td>
<td>75.00</td>
<td>10</td>
<td>83.33</td>
</tr>
<tr>
<td>Europe</td>
<td>Wagon</td>
<td>2</td>
<td>16.67</td>
<td>12</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### Land Rover

The FREQ Procedure

<table>
<thead>
<tr>
<th>Origin</th>
<th>Type</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe</td>
<td>SUV</td>
<td>3</td>
<td>100.00</td>
<td>3</td>
<td>100.00</td>
</tr>
</tbody>
</table>
MVA Creation: PROC SQL

Desired output: one report of demographic information per student in SASHELP.CLASS

```
proc sql noprint;
select distinct(name) into :names separated by ' ' from sashelp.class;
select count(distinct(name)) into :counts trimmed from sashelp.class;
quit;
```

```
99  %put counts = &counts.;
counts = 19
100  %put names = &names.;
names = Alfred Alice Barbara Carol Henry James Jane Janet Jeffrey John Joyce Judy Louise Mary Philip Robert Ronald Thomas William
```
%macro reports();
  %do i = 1 %to &counts.;
    title ":%scan(&names., &i.);"
    proc report data=sashelp.class nowd;
      where name = "%scan(&names, &i.);"
      columns sex age height weight;
      define sex / 'Sex';
      define age / 'Age';
      define height / 'Height';
      define weight / 'Weight';
    run;
  title;
%end;
%mend;
MVA Creation: PROC SQL

```
103  %reports();
NOTE: Writing HTML body file: sashtml3.htm

NOTE: There were 1 observations read from the data set SASHELP.CLASS.
      WHERE name='Alfred';
NOTE: PROCEDURE REPORT used (Total process time):
      real time 1.26 seconds
cpu time 0.56 seconds
...

NOTE: There were 1 observations read from the data set SASHELP.CLASS.
      WHERE name='William';
NOTE: PROCEDURE REPORT used (Total process time):
      real time 0.02 seconds
cpu time 0.03 seconds
```
MVA Creation: PROC SQL

<table>
<thead>
<tr>
<th>Alfred</th>
<th>William</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sex</strong></td>
<td><strong>Sex</strong></td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>
Advanced Examples
Multiple Macro Variable Arrays Example

• 4 correlations to be produced using SASHELP.CARS
  • Invoice amount vs MPG highway
  • Invoice amount vs number of cylinders
  • MSRP vs MPG highway
  • MSRP vs number of cylinders
• 2 macro variable arrays will produce 4 outputs
Multiple Macro Variable Arrays Example

```sas
%macro corr(dat, var1, var2);
  %do i = 1 %to %sysfunc(countw(&var1.));
    %do k = 1 %to %sysfunc(countw(&var2.));
      proc corr data=sashelp.&dat. outp=c_%scan(&var1., &i.)_%scan(&var2., &k.);
      var %scan(&var1., &i.) %scan(&var2., &k.);
      run;
    %end;
  %end;
%end;
%mend;
```

Macro Variable

Macro Variable Arrays
%macro corr(dat, var1, var2);
  %do i = 1 %to %sysfunc(countw(&var1.));
    %do k = 1 %to %sysfunc(countw(&var2.));
      proc corr data=sashelp.&dat. outp=c_%scan(&var1., &i.)_%scan(&var2., &k.);
        var %scan(&var1., &i.) %scan(&var2., &k.);
      run;
    %end;
  %end;
%end;
%end;
%mend;

%corr(cars, invoice msrp, mpg_highway cylinders);
Multiple Macro Variable Arrays Example

```
130 %corr(cars, invoice msrp, mpg_highway cylinders);
MLOGIC(CORR): Beginning execution.
MLOGIC(CORR): Parameter DAT has value cars
MLOGIC(CORR): Parameter VAR1 has value invoice msrp
MLOGIC(CORR): Parameter VAR2 has value mpg_highway cylinders
MLOGIC(CORR): %DO loop beginning; index variable I; start value is 1; stop value is 2;
             by value is 1.
MLOGIC(CORR): %DO loop beginning; index variable K; start value is 1; stop value is 2;
             by value is 1.
MPRINT(CORR): proc corr data=sashelp.cars noprint outp=c_invoice_mpg_highway;
MPRINT(CORR): var invoice mpg_highway;
MPRINT(CORR): run;

NOTE: Writing HTML Body file: sashtml1.htm
NOTE: The data set WORK.C_INVOICE_MPG_HIGHWAY has 5 observations and 4 variables.
NOTE: Compressing data set WORK.C_INVOICE_MPG_HIGHWAY increased size by 100.00 percent.
     Compressed is 2 pages; un-compressed would require 1 pages.
NOTE: PROCEDURE CORR used (Total process time):
     real time 1.26 seconds
     cpu time 0.73 seconds
```
Multiple Macro Variable Arrays Example

MPRINT(CORR): proc print data=c_invoice_mpg_highway noobs label;
MPRINT(CORR): where lowercase(_name_) = "invoice";
MPRINT(CORR): var mpg_highway;
MPRINT(CORR): label mpg_highway = "Corr invoice and mpg_highway";
MPRINT(CORR): run;

NOTE: There were 1 observations read from the data set WORK.C_INVOICE_MPG_HIGHWAY.
    WHERE LOWCASE(_name_)="invoice";
NOTE: PROCEDURE PRINT used (Total process time):
    real time 0.02 seconds
    cpu time 0.03 seconds
Multiple Macro Variable Arrays Example

<table>
<thead>
<tr>
<th>Corr invoice and mpg_highway</th>
<th>Corr invoice and cylinders</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.43459</td>
<td>0.64523</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corr msrp and mpg_highway</th>
<th>Corr msrp and cylinders</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.43962</td>
<td>0.64974</td>
</tr>
</tbody>
</table>
%BQUOTE Function with MVAs Example

• Desired output: Mean summaries of Mother’s age and pregnancy weight gain from Infant birth weight (bweight) SASHELP data set with the title of each summary containing the label of the variable used

\[
\text{%macro means(dat, var, lab);}
\]
\[
\text{%do i = 1 %to %sysfunc(countw(&var.));}
\]
\[
\text{title "%scan(&lab., &i., ~)";}
\]
\[
\text{proc means data=sashelp.&dat.;}
\]
\[
\text{var %scan(&var., &i.);}
\]
\[
\text{run;}
\]
\[
\text{title;}
\]
\[
\text{%end;}
\]
\[
\text{%mend;}
\]
%macro means(dat, var, lab);
  %do i = 1 %to %sysfunc(countw(&var.));
    title "%scan(&lab., &i., ~)"
    proc means data=sashelp.&dat.;
    var %scan(&var., &i.);
    run;
    title;
  %end;
%mend;

%means(bweight, mom_age m_wtgain, %bquote(Mother's Age)~%bquote(Mother's Pregnancy Weight Gain));
%BQUOTE Function with MVAs Example

146   %means(bweight, mom_age m_wtgain, %bquote(Mother’s Age)^%bquote(Mother’s Pregnancy 146! Weight Gain));
MLOGIC(MEANS):  Beginning execution.
MLOGIC(MEANS):  Parameter DAT has value bweight
MLOGIC(MEANS):  Parameter VAR has value mom age m wtgain
MLOGIC(MEANS):  Parameter LAB has value ‘Mother’es Age’~’Mother’es Pregnancy Weight Gain’
MLOGIC(MEANS):  %DO loop beginning; index variable l; start value is 1; stop value is 2;
                by value is 1.

NOTE:  There were 50000 observations read from the data set SASHELP.BWEIGHT.
NOTE:  PROCEDURE MEANS used (Total process time):
                  real time         0.04 seconds
                  cpu time          0.06 seconds

MLOGIC(MEANS):  %DO loop index variable l is now 2; loop will iterate again.

NOTE:  There were 50000 observations read from the data set SASHELP.BWEIGHT.
NOTE:  PROCEDURE MEANS used (Total process time):
                  real time         0.03 seconds
                  cpu time          0.03 seconds

MLOGIC(MEANS):  %DO loop index variable l is now 3; loop will not iterate again.
MLOGIC(MEANS):  Ending execution.
%BQUOTE Function with MVAs Example

Mother’s Age

The MEANS Procedure

<table>
<thead>
<tr>
<th>Analysis Variable : mom_age</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>50000</td>
</tr>
</tbody>
</table>

Mother’s Pregnancy Weight Gain

The MEANS Procedure

<table>
<thead>
<tr>
<th>Analysis Variable : m_wtgain</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>50000</td>
</tr>
</tbody>
</table>
Conclusion

• Macros optimize repetitive tasks on data sets, variables, or variable values
• Macro variable arrays take this a step further
  • Combines array processing with macro language
  • Increase robustness of macros
  • Can be automated
  • Reduce programmer error
• Ways to create a macro variable array
  • Explicitly defined
    • %LET Statement
    • Macro parameter
  • CALL SYMPUT
  • PROC SQL

Image from: https://psychologycompass.com/blog/tactics-to-stop-worrying/
Any questions?

Name: Veronica Renauldo
Organization: QST Consultations, LTD
Work Phone: (616) 892-3723
E-Mail: vrenauldo@qstconsultations.com