Using Macros to Standardize and Streamline Report Writing

Christopher Louden
University of Texas Health Science Center at San Antonio

Outline

- Overview of Macro Language
- Report Writing
  - The REPORT procedure
  - The Output Delivery System (ODS)
- Macro Examples
  - Utility Macros
  - A Macro for Categorical Variables
  - A Macro for Continuous Variables
- Conclusion

Why Use Macros?

- To reduce programming time.
- To standardization programming.
- To declare constants (no magic numbers).

```
data simulated;
   do i = 1 to 100;
      x = rannorm(123);
      output;
   end;
run;
```

```
%LET n_sims = 100;
%LET seed = 123;

data simulated;
   do I = 1 to &n_sims;
      x = rannorm(&seed);
      output;
   end;
run;
```

SAS Macro Language

- The macros language provides the instructions.
- The macro processor does the work.
- Macro are based on text substitution in the code.

Macro Variables

- Two ways to declare
  - `%LET` statement
    - Used in open code.
    - Known value.
    - `%LET n_sims = 200;`
  - `call symput`
    - Used in data steps or PROC IML.
    - Computed value (as a character).
    - `call symput('macro_var_name', value);`

Macro Variables

- Reference macro variables with &.

```
%LET test = chisq;

proc freq data = a;
   table x*y / &test;
run;
```

```
%LET test = chisq;

proc freq data = a;
   table x*y / chisq;
run;
```

- Nested references use &&.

```
%LET city = Cary;
%LET state = NC;

&&city._&state;
```

```
&city._NC
```

```
Cary_NC
```

Note: the period after city denotes the end of the variable name (i.e. the variable is 'city' not 'city_').

Macro Functions

- Advantages:
  - Allow compartmentalization of code.
  - Facilitate code reuse between programs.
  - Reduce programming time.
  - Decrease the number of programming errors.

- Limitations:
  - Do not return values directly.

Macro Functions

- Called with *%macro_name*
  - *%my_macro1*;
- Arguments can be passed to the macro.
  - Positional arguments in order.
    - *%my_macro2*(x, y, z);
  - Keyword arguments proceeded by name.
    - *%my_macro3*(x, arg2=z, arg3=y);
    - *%my_macro3*(x, arg3=z, arg2=y);
- Arguments from within macro using &.

Macro Function (Example)

A macro to format p-values in a data step:

```
%macro format_p_val(in, out);
   if missing(&in) then &out = ' ';
   else if 0.05 le &in lt 0.0505 then &out=compress('0.051');
   else if 0.045 le &in lt 0.05 then &out=compress(round(&in,0.001));
   else if 0 le &in lt 0.001 then &out=compress('<0.001');
   else if 0.001 le &in lt 0.01 then &out=compress(round(&in,0.001));
   else if 0.01 le &in le 1 then &out=compress(round(&in,0.001));
%mend format_p_val;
```

Context:

```
data pvalues;
   length cpvalue $6.;
   input npvalue;
   %format_p_val(npvalue,cpvalue);
   datalines;
```

Macro Programing

- If-else-if statements:
  - `%IF condition %THEN %DO; <statements> %END;`
    `%ELSE %DO; <statements> %END;`
- Do statements (loops):
  - `%DO i = 1 %TO &max_iter;`
    `    <statements>`
    `%END;`
- System calls:
  - `%SYMEXIST(&macro_var_name);`

Scope of Macro Variables

- Global – can be used in anywhere in the code.
  - Declared in open code.
  - When the `%global` macro is used.
- Local – can be used only in the macro in which the variable was declared.
  - Declared in a macro function.
  - Passed as an argument.

Report Writing

- The REPORT procedure is a versatile tool for summarizing data.

- Can be used with ODS and PROC TEMPLATE to create well laid out tables.

Report Writing

- The Output Delivery System (ODS) directs the output of procedures to different output streams.

- Use with statistical procedures to capture their output:

```
ods trace on;
ods output CrossTabFreqs = freqs;
proc freq data = a;
   table x*y;
run;
ods trace off;
```

Report Writing

- Use with procedures to direct the output to a file:

```
options orientation = landscape number nodate;
ods noproctitle;
ods rtf body = "C:\Documents\Output.rtf"
  style = styles.LPG
  bodytitle;
<Proc report statements>
ods rtf close;
```

- Very powerful when used with templates

Planning Macros for PROC REPORT

- What will the data set need to look like?
- How do you get the pieces you need?
- What aspects might change from macro call to macro call?

Layout of Data Set

- The table is divided into panels.

Label	Group_1	Group_2	Total	P_value	Row	Panel
Age				0.003	1	1
N	193	203	396		2	1
Mean (SD)	52.4 (4.3)	51.3 (3.1)	51.8 (3.7)		3	1
SEM	0.3	0.2	0.2		4	1
Median [IQR]	51.9 [5.6]	51.2 [4.3]	51.4 [4.7]		5	1
Min, Max	43.7, 64.1	44, 59.2	43.7, 64.1		6	1

Where to Get the Pieces

- Summary Statistics
  - The MEANS or UNIVARIATE procedures
  - The FREQ procedure
- P-Values
  - The GLM or NPAR1WAY procedures
  - The FREQ procedure
- Panel Number
  - Argument or Macro Counter

What Aspects Might Change

- Parametric or Non-Parametric Tests
- Labels, Number of Groups
- Panel Number

Order of Operations

- Error Checking
- Gathering Statistics
- Manipulating Data Sets
- Clean Up (if desired)

Error Checking

- Check parameters for bad values.

```
data _NULL_;
   set _temp_contents;
   %LET _testerr = 1;
   if variable="&testvar" then call symput(_testerr, "0");
run;
```

- Use %RETURN or %GOTO.

```
%IF _testerr="1" %THEN %GOTO TESTERR;
<statements>
%TESTERR:
%PUT ERROR: Variable &testvar does not exist in data set &dat;
```

Counting Levels

- The *%scan* macro.
  - Scans a macro variable and return the i^{th} element.
  - Elements separated by delimiter (default is space).

```
%LET i=0;
%DO %UNTIL (&_test=);
    %LET i=%EVAL(&i+1);
    %LET _test = %SCAN(&testvarlevels,&i);
%END;
%LET n_row=%EVAL(&i-1);
```

Getting Variable Names

- The FREQ procedure lists all the possible values for a variable in the data set.

```
ods trace on;
ods output OneWayFreqs = _names (keep=F_&classvar);
proc freq data = &dat; table &classvar; run;
ods trace off;

%LET n_level = 0;
data _NULL_;
   set _names;
   call symput('n_level',_n_);
run;
```

Getting Variable Names

- The ODS output of the FREQ procedure has the names of the variables.

```
%LET levels =;
%DO i = 1 %TO &n_level;
   data _NULL_;
      set _names;
      if &i = _n_ then call
        symput(compress("clevel"||"&i"),F_&classvar);
   run;
   %LET levels = &levels &&clevel&i;
%END;
%LET n_col=%EVAL(&i-1);
```

Gathering Statistics

- Use conditional programming to decide which test to use.

- Use ODS and the `output` statement to retrieve the statistics of interest as datasets.

Manipulation of Data Sets

- Use of small temporary data sets facilitates the construction of the panel.
  - P-value
  - Summary Statistics
  - Blank Line
- `%scan` macro to select the appropriate label.
- `Set` and `Merge` statements to combine them.

Clean Up

- All temporary data set names begin with _.
- Use PROC DATASETS to delete them.

```
proc datasets;
   delete _:;
quit;
```

- The colon wildcard signifies every dataset beginning with _.

Utility Macros

- A global counter

```
%macro initialize_counter(start=1);
   %global counter;
   %let counter = &start;
%mend;
```

- The date

```
%macro fdate(fmt);
   %global fdate;
   data _null_;
      call symput("fdate",trim(left(put("&sysdate9"d,&fmt))));
   run;
%mend fdate;
%fdate(worddatx.);
```

Utility Macros

- Paginate the table

```
%macro paginate(dat, foot=4, totalrows=22);
```

- Sorts the data by panel then row.
- Keeps a tally of the rows used on a page.
- Checks if a panel will fit on the page.
  - If it can, the panel is assigned that page number.
  - If not, the page number is incremented.

Putting it Together

Error Check Parameters

Determine Groups and Labels

Gather Statistics

Put Together Data Set

Example

Questions?