SAS Macro Programming

July 8, 2003

Overview

- purpose is to make SAS programming more efficient and to reduce coding errors
- macro variables
  - enable substitution of text into SAS programs
- macro programs
  - enable performing the same task on different inputs without rewriting code

Example dataset

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
<th>Format</th>
<th>Informat</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>author</td>
<td>Char</td>
<td>50</td>
<td>115</td>
<td></td>
<td></td>
<td>First Author</td>
</tr>
<tr>
<td>8</td>
<td>cost</td>
<td>Num</td>
<td>8</td>
<td>8</td>
<td>DOLLAR9.2</td>
<td>Wholesale Cost</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>datesold</td>
<td>Num</td>
<td>4</td>
<td>32</td>
<td>MMDDY98</td>
<td>MDDY98</td>
<td>Date Book Sold</td>
</tr>
<tr>
<td>9</td>
<td>listpric</td>
<td>Num</td>
<td>8</td>
<td>16</td>
<td>DOLLAR9.2</td>
<td>List Price</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>publishr</td>
<td>Char</td>
<td>50</td>
<td>165</td>
<td></td>
<td></td>
<td>Publisher</td>
</tr>
<tr>
<td>2</td>
<td>saleid</td>
<td>Num</td>
<td>8</td>
<td>0</td>
<td>8.</td>
<td></td>
<td>Sale ID</td>
</tr>
<tr>
<td>3</td>
<td>saleinit</td>
<td>Char</td>
<td>3</td>
<td>62</td>
<td></td>
<td></td>
<td>Sales Person Initials</td>
</tr>
<tr>
<td>10</td>
<td>salepric</td>
<td>Num</td>
<td>8</td>
<td>24</td>
<td>DOLLAR9.2</td>
<td>Sale Price</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>section</td>
<td>Char</td>
<td>26</td>
<td>36</td>
<td></td>
<td></td>
<td>Section</td>
</tr>
<tr>
<td>5</td>
<td>title</td>
<td>Char</td>
<td>50</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Macro variables

- `%let` keyword defines a macro variable and assigns it a value
- use `&` before macro variable name when referencing variable
- use `%eval` keyword to convert a macro variable's value to numeric
- when referencing macro variables in character literals, use double quotes
Macro variables example

%let repmonth=4;
%let repyear=2001;
%let repword=%sysfunc(mdy(&repmonth,1,&repyear),monname9.);

data month&repmonth;
  set books.ytdsales;
  mosale=month(datesold);
  label mosale='Month of Sale';
run;

proc tabulate data=month&repmonth;
  title "Sales During &repword &repyear";
  where mosale=&repmonth and year(datesold)=&repyear;
  class section;
  var salepric listpric cost;
  tables section all='**TOTAL**',
    (salepric listpric cost)*(n*f=4. sum*f=dollar9.2);
run;

*proc gchart data=month&repmonth ;
proc chart data=month&repmonth
  (where=(mosale < %eval(&repmonth+1) and
    year(datesold)=&repyear));
  title "Sales Through &repword &repyear";
  pie section / sumvar=salepric noheading ;
runk

Using built-in SAS macro variables

title "Sales Report";
title2 "As of &systime &sysday &sysdate";
title3 "Using SAS Version: &sysver";
proc means data=books.ytdsales n sum;
  var salepric;
runk

Output

Sales Report
As of 06:38 Friday 19OCT01
Using SAS Version: 8.00

The MEANS Procedure

Analysis Variable : salepric Sale Price

<table>
<thead>
<tr>
<th>N</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>6959</td>
<td>210588.23</td>
</tr>
</tbody>
</table>
Using call symput to assign a value from a data step variable to a macro variable

- embedded put statement also formats the value before assigning it
- retain statement used in following example
  - initializes a variable at the beginning of a data step
  - tells SAS to carry its value forward as it sequentially processes records in the dataset

Example of call symput

data temp;
  set books.ytdsales end=lastobs;
  retain sumintwb 0;
  if section in ('Internet','Web Design') then
    sumintwb=sumintwb + salepric;
  if lastobs then
    call symput('INTWEBSL',put(sumintwb,dollar10.2));
run;
proc chart data=temp;
  title "Internet and Web Design Sales: &intwebsl";
  title2 "As of &enddate";
  hbar section / sumvar=salepric;
  format salepric dollar10.2;
run;

Output

<table>
<thead>
<tr>
<th>Section</th>
<th>Freq</th>
<th>Sale Price</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>1777</td>
<td>$53,998.95</td>
<td>$107,320.40</td>
</tr>
<tr>
<td>Networks and Com</td>
<td>649</td>
<td>$19,472.97</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>1877</td>
<td>$56,964.04</td>
<td></td>
</tr>
<tr>
<td>Programming Lang</td>
<td>900</td>
<td>$26,830.81</td>
<td></td>
</tr>
<tr>
<td>Web Design</td>
<td>1756</td>
<td>$53,321.45</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1795</td>
<td>$30,000.00</td>
<td></td>
</tr>
</tbody>
</table>

Writing macro programs

- like subroutines or functions
- macro function is defined by the following structure
  
  \%macro macro-name
  .
  .  < statements to be executed by macro >
  .
  \%mend macro-name

- code inside macro is essentially just SAS code
- but special macro keywords are used to control conditional and iterative processing
- macro must be defined before it can be called
Example of macro function

%macro daily;
  proc means data=books.ytdsales(where=(datesold=today()))
    maxdec=2 sum;
  title "Daily Sales Report for &sysdate";
  class section;
  var salepric;
  run;
%if &sysday=Friday %then %do;
  proc means data=books.ytdsales
    (where=(today()-6 le datesold le today()))
    sum maxdec=2;
  title "Weekly Sales Report Week Ending &sysdate";
  class section;
  var salepric;
  run;
%end;
%mend daily;

Calling the macro

- call a macro using %macroname

Example

%daily

Output

Daily Sales Report for 19OCT01

The MEANS Procedure

Analysis Variable : salepric Sale Price

<table>
<thead>
<tr>
<th>Section</th>
<th>N</th>
<th>Obs</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>7</td>
<td>212.66</td>
<td></td>
</tr>
<tr>
<td>Networks and Communication</td>
<td>5</td>
<td>123.76</td>
<td></td>
</tr>
<tr>
<td>Operating Systems</td>
<td>6</td>
<td>224.91</td>
<td></td>
</tr>
<tr>
<td>Programming Languages</td>
<td>3</td>
<td>81.36</td>
<td></td>
</tr>
<tr>
<td>Web Design</td>
<td>2</td>
<td>58.90</td>
<td></td>
</tr>
</tbody>
</table>

Weekly Sales Report Week Ending 19OCT01

The MEANS Procedure

Analysis Variable : salepric Sale Price

<table>
<thead>
<tr>
<th>Section</th>
<th>N</th>
<th>Obs</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>46</td>
<td></td>
<td>1391.95</td>
</tr>
<tr>
<td>Networks and Communication</td>
<td>15</td>
<td></td>
<td>420.37</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>36</td>
<td></td>
<td>1171.83</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>24</td>
<td></td>
<td>719.03</td>
</tr>
<tr>
<td>Web Design</td>
<td>35</td>
<td></td>
<td>1049.40</td>
</tr>
</tbody>
</table>
Example of macro to do iterative processing

- the following macro copies the book sales data into 12 separate datasets, one for each month of the year

```sas
%macro makesets;
  data
    %do i=1 %to 12;
      month&i
    %end;
  ;
  set books.ytdsales;
  mosale=month(datesold);
  if mosale=1 then output month1;
  %do i=2 %to 12;
    else if mosale=&i then output month&i;
  %end;
  run;
%mend makesets;
%makesets
```

Example of macro program with positional parameters

```sas
options mprint nlogic;
%macro listparm(start,stop,opts);
  title "Books Sold by Section Between &start and &stop";
  proc means data=books.ytdsales &opts;
    where "&start"d le datesold le "&stop"d;
    class section;
    var salepric;
  run;
%mend listparm;

***** First call to LISTPARM, all 3 parameters specified;
%listparm(01JUN1998,15JUN1998,n sum)

***** Second call to LISTPARM, first 2 parameters specified and;
***** third parameter is null;
%listparm(01SEP1998,15SEP1998,)
```

Passing parameters to macros

- parameters may be passed to a macro program
  - by position
  - by keyword
- parameters are named in parentheses after macro name in macro definition
- values are listed in parentheses after macro name in macro call

Output

Books Sold by Section Between 01JUN2001 and 15JUN2001

The MEANS Procedure

Analysis Variable : salepric Sale Price

<table>
<thead>
<tr>
<th>Section</th>
<th>N</th>
<th>N</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>84</td>
<td>84</td>
<td>2533.78</td>
</tr>
<tr>
<td>Networks and Communication</td>
<td>26</td>
<td>26</td>
<td>820.8350000</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>71</td>
<td>71</td>
<td>2092.13</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>46</td>
<td>46</td>
<td>1368.66</td>
</tr>
<tr>
<td>Web Design</td>
<td>66</td>
<td>66</td>
<td>2002.66</td>
</tr>
</tbody>
</table>
Passing parameters by keyword

- enables setting defaults in macros

Example

options mprint nlogic;

%macro keyparm(start=01JAN2001, stop=31DEC2001, opts=N SUM MIN MAX);
title "Books Sold by Section Between &start and &stop";
proc means data=books.ytdsales &opts;
  where "&start"d le datesold le "&stop"d;
  class section;
  var salepric;
run;
%mend keyparm;

*----First call to KEYPARM: specify all keyword parameters;
%keyparm(start=01JUN2001, stop=15JUN2001, opts=n sum)

*----Second call to KEYPARM: specify start and stop,;
*----opts is null: should see default stats for PROC MEANS;
%keyparm(start=01SEP2001, stop=15SEP2001, opts=)

*----Third call to KEYPARM: use defaults for start and stop,;
*----specify opts;
%keyparm(opts=n sum)
Options for macro processing

- **mprint (nomprint)**
  - specifies whether SAS statements that are generated by macro execution are displayed

- **mlogic (nomlogic)**
  - specifies whether SAS traces execution of the macro language processor. If `mlogic` is specified, trace information is written in SAS log