Stat 302
Statistical Software and Its Applications
SAS: A Start into Macros

Fritz Scholz

Department of Statistics, University of Washington

Winter Quarter 2013
SAS has some special built-in macro variables, such as &VAR, &SYSDATE9, &SYSTIME

When you run a SAS program it first checks for the special characters & and % and processes them first.

Any macro variable (starting with &) gets replaced by what it stands for. It is like shorthand code.

```sas
libname learn "U:\learn";
title "The Date is &sysdate9 - the Time is &systime";
proc print data=learn.test_scores noobs;
run;
```

In the title, substitution only works within double quotes " ".

```sas
title 'The Date is &sysdate9';
```
returns: The Date is &sysdate9
The Date is 07MAR2013 - the Time is 17:38

<table>
<thead>
<tr>
<th>ID</th>
<th>Score1</th>
<th>Score2</th>
<th>Score3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>95</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>78</td>
<td>77</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>91</td>
<td>92</td>
</tr>
</tbody>
</table>
&systime and &sysdate9 return time and date at the start of a SAS session.

To get time and date at the moment of a SAS program execution do

options nodate;
    %let timenow=%sysfunc(time(), time.);
    %let datenow=%sysfunc(date(), date9.);
libname learn "U:\learn";
title "The Date is &datenow - the Time is &timenow";
proc print data=learn.test_scores noobs;
run;
## The Output from Code

The Date is 09MAR2013 - the Time is 10:05:39

<table>
<thead>
<tr>
<th>ID</th>
<th>Score1</th>
<th>Score2</th>
<th>Score3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>95</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>78</td>
<td>77</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
<td>91</td>
<td>92</td>
</tr>
</tbody>
</table>
As seen in the previous code you can assign a value to a macro variable with a `%LET` statement.

It is done in open code, not in inside a DATA or PROC step.

```sas
libname learn "U:\learn";
%let var_list = RBC WBC Chol;

title "Using a Macro Variable List";
proc means data=learn.blood
    n mean min min max maxdec=1;
    var &var_list;
run;
```
The MEANS Procedure
Using a Macro Variable List

The Output from Code

<table>
<thead>
<tr>
<th>Variable</th>
<th>Label</th>
<th>N</th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBC</td>
<td></td>
<td>916</td>
<td>5.5</td>
<td>1.7</td>
<td>8.8</td>
</tr>
<tr>
<td>WBC</td>
<td></td>
<td>908</td>
<td>7043.0</td>
<td>4070.0</td>
<td>10550.0</td>
</tr>
<tr>
<td>Chol</td>
<td>Cholesterol</td>
<td>795</td>
<td>201.4</td>
<td>17.0</td>
<td>331.0</td>
</tr>
</tbody>
</table>
%let n = 3;

data generate;
    do Subj = 1 to &n;
        x = int(100*ranuni(0)+1);
        output;
    end;
run;

title "Data Set with &n Random Numbers";
proc print data=generate noobs;
run;
## Data Set with 3 Random Numbers

<table>
<thead>
<tr>
<th>Subj</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>99</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
</tr>
</tbody>
</table>
%macro gen(n,Start,End);
data generate;
    do Subj = 1 to &n;
        x = int((&End-&Start+1)*ranuni(0)+&Start);
        output;
    end;
run;
proc print data=generate noobs;
title "Randomly Generated Data Set with &n Obs";
title2 "Values are integers from &Start to &End";
run;
%mend gen; * mend = macro end;
%gen(4,1,100)
/* no ; required here, could result in error */
Randomly Generated Data Set with 4 Obs
Values are integers from 1 to 100

<table>
<thead>
<tr>
<th>Subj</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>98</td>
</tr>
</tbody>
</table>
Note the different output in the LOG produced by these code pieces

```sas
%let prefix = abc;
data &prefix123;
   x= 3;
run;
```

and

```sas
%let prefix = abc;
data &prefix.123;
   x= 3;
run;
```

The period in `&prefix.123` is not a valid character in a variable name and thus acts as in indicator where the token `&prefix` ends, so that proper substitution can take place.
ERROR 200-322: The symbol is not recognized and will be ignored.

137 x= 3;
138 run;

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.PREFIX123 may be incomplete.
When this step was stopped there were 0 observations and 1 variables.
WARNING: Data set WORK.PREFIX123 was not replaced because this step was stopped.

NOTE: DATA statement used (Total process time):
  real time 0.00 seconds
  cpu time 0.00 seconds
143  %let prefix = abc;
144  data &prefix.123;
145  x= 3;
146  run;

NOTE: The data set WORK.ABC123 has 1 observations and
NOTE: DATA statement used (Total process time):
    real time          0.00 seconds
    cpu time           0.00 seconds
%let libref = learn; * no quotes around learn;
proc print data = &libref..test_scores;
  title "Listing of Test";
run;

- Here a double period is needed.
- The first indicates the end of the &libref token.
- The second separates the libname and the data set.
Macro values assigned outside of a macro are, by default, global to the SAS session.

This makes them a useful tool for transferring values between data steps.

The next example expresses the RBC and WBC values in the blood data set and expresses them in percentages w.r.t. the means for RBC and WBC.

You can examine the respective results in the WORK library.
Using Macro Values in Transfer between Data Steps

libname learn "U:\learn";
proc means data = learn.blood noprint;
  var RBC WBC;
  output out=means mean = M_RBC M_WBC;
run;
data _null_;  
  set means;
  call symput('AveRBC',M_RBC);
  call symput('AveWBC',M_WBC);
run;

data new;
  set learn.blood(obs=5 keep=Subject RBC WBC);
  Per_RBC = RBC / &AveRBC;
  Per_WBC = WBC / &AveWBC;
  format Per_RBC Per_WBC percent12.4;
run;
PROC MEANS creates data set (named Means, line 4 in code) consisting of M_RBC and M_WBC.

The data _null_; uses CALL SYMPUT to assign a value of a DATA step variable (M_RBC) to a macro variable (’AveRBC’).

The values of M_RBC and M_WBC are made available by set means; prior to that.

Can’t use a %Let because M_RBC and M_WBC are unknown.

The values of &M_RBC and &M_WBC are not available in same data step.

Need an additional DATA step to finish the job.

The SAS format PERCENT adds a % sign and multiplies by 100.
data spirit;
  input gas weight headwind TO_distance;
  TO_DistL10 = log10(TO_Distance);
  weightL10 = log10(weight);
datalines;
  36 2600 7 229
  71 2800 9 287
  111 3050 9 389
  151 3300 6 483
  201 3600 4 615
  251 3900 2 800
  301 4200 0 1023
run;
proc reg data=spirit noprint outest = coefs;
  model weightL10 = TO_distL10;
run;

data _null_;  
set coefs;
  call symput('a',Intercept);
  call symput('b',TO_distL10);
run;

data temp;
  input x1;
  LS_Line = 10**&a * x1**&b;
datalines;
  100
  3000
run;
data combine;
    merge spirit temp;
run;

title "Log10-Log10 Scatter Plot with Regression Line";
proc sgplot data=combine ;
    scatter y = weight x=TO_distance;
    yaxis type=log logstyle=logexpand logbase=10
        min =2000 max=6000;
    xaxis type=log logstyle=logexpand logbase=10
        min = 100 max=3000;
    series x = x1 y= LS_Line; * this connects points;
run;
Log10-Log10 Scatter Plot with Regression Line

weight

TO_distance

weight LS_Line