SAS® Macros 101

How I learned to stop worrying and love macros
### Abstract
Alex inherited a set of SAS monthly reports from his former coworker, whose user id appears throughout the code. The former coworker went through the SAS code each month manually changing each date reference to reflect the current month end. Alex thinks this is too much work. He is ready to try SAS® Macros for running March 2014 month end represented as ’2014-03-31’. Alex will change the code into a macro called mrpt.

### Objective
- Minimize code changes
- Eliminate hard coding
- Make it easier to hand off code

### Method
- Add %macro mrpt(dtvar); to top
- Add %mend mrpt; to end
- Change all occurrences of month end date ’2014-03-31’ to &dtvar. using a period at the end
- Change all userid references to &&sysuserid. using a period at the end.
- Change single quote to double quote on pathname references.
- Add %mrpt(’2014-03-31’) below %mend.

```sas
%macro mrpt(dtvar);
libname mthly "Z:\&sysuserid\data";
%put dtvar = &dtvar;
/* <code> */
%mend mrpt;
%mrpt(’2014-03-31’)
```

### Results
- Only need to change date value in one place each month.
- Eliminated hard coding of dates
- &&sysuserid always reflects current user id

### Conclusions
- Less manual intervention = Less prep time and fewer errors
- Use of &&sysuserid rather than hard coding user id simplifies hand off
- Easy to run for multiple months

### References
- [SUGI 28: Nine Steps to Get Started Using SAS(r) Macros](https://support.sas.com/publishing/authors/stroupe_jane/index.html) Jane Stroupe
- [SAS(R) 9.3 Macro Language: Reference](https://support.sas.com/publishing/authors/index.html)
Alex's manager thought his macro solution for monthly reporting was totally awesome. He instructed the other team members to convert their SAS code to macros where possible. He told Alex to help them out if they had problems.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Give Alex's co-workers the means to debug their macro code</td>
<td>• Turning options on options symbolgen mprint mlogic mcompilenote=noautocall;</td>
</tr>
<tr>
<td>• Minimize the amount of time he has to spend helping them</td>
<td>• Symbolgen Values assigned to macro variables</td>
</tr>
<tr>
<td></td>
<td>• mprint Macro code</td>
</tr>
<tr>
<td></td>
<td>• mlogic Macro logic</td>
</tr>
<tr>
<td></td>
<td>• mcompilenote Macro compilation message</td>
</tr>
<tr>
<td></td>
<td>• Turning options off options nosymbolgen nomprint nomlogic mcompilenote=none;</td>
</tr>
</tbody>
</table>

/* Macro fin_mthly. Save and format begin date. */

options symbolgen mprint mlogic mcompilenote=noautocall;

symbolgen Values assigned to macro variables
mprint Macro code
mlogic Macro logic
mcompilenote Macro compilation message

%macro fin_mthly(ccyy,dtfmt);
    proc sql noprint;
    select begin_date format=&dtfmt into :date1-date4
    from sasuser.schedule where year(begin_date)=&ccyy.;
    quit;
    %put date2 = &date2 ;
    %put dtfmt = &dtfmt;
%mend fin_mthly;

%fin_mthly(2002,date9.) No semi-colon
### Results

**SYMBOLGEN:** Macro variable `dtfmt` resolves to `date9`.

**MPRINT**(`FIN_MTHLY`): select `begin_date` format=`date9`.
into :`date1`:`date4`

**MLOGIC**(`FIN_MTHLY`): `%PUT &dtfmt mcompilenote=noautocall;`

### Conclusions

- Use the macro debugging options in combination with `%put` statements to debug, fix and test macros.
- Always have options `mcompilenote=noautocall` to confirm code changes to macros have correctly compiled.

### References

- [249-2012: A Tutorial on the SAS® Macro Language](#)  
  John J. Cohen
- [SUGI 28: Nine Steps to Get Started Using SAS(r) Macros](#)  
  Jane Stroupe
- [SAS(R) 9.3 Macro Language: Reference](#)