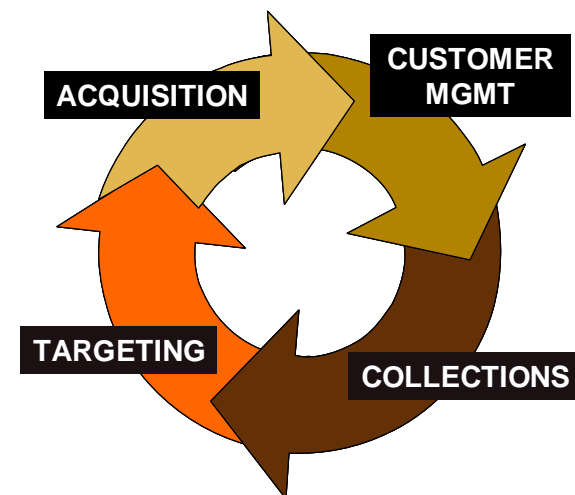

Work Smarter, not Harder

The Beginner's Guide to Using SAS Macros



CRMA – General Overview

- Upon graduating university I joined CBA into their Credit Risk Management and Analytics team. The group is a mix of graduates and managers with CRM experience developed throughout the world
- Customer Scoring, Acquisition Policy, Collections Initiatives and Customer Management
- CRMA's objective is to support business growth by developing strategies and policies that optimise risk and reward at each point of the lifecycle (shown right)



How / Why do we use SAS?

- SAS is our key tool used for data management, analytics and reporting
- We have a structured data-mart on which business queries are run
- Allows efficient data manipulation, processing and output
- Enables the automation of frequently run programs



Macros – What is a Macro?

- A Macro is a SAS program that is prepared and saved, then called into a later SAS program to perform a certain operation or series of tasks
- SAS Syntax is used to manipulate SAS Datasets, SAS Macros are used to manipulate SAS Syntax
- Quite simply a Macro is of the form:

```
%macro macroname(macrovar1,macrovar2,....);
```

```
Loops/Datasteps/Procedures;
```

```
%mend;
```

```
%macroname(variation1 of macrovars);
```

```
%macroname(variation2 of macrovars);
```

```
...
```



How is this Used?

- %macro calls the macro to SAS
- The macroname is the name of the macro
- Macro variables are also named at your discretion
- Call these within the program using an “&”
i.e. data=¯ovar1;
- %mend signifies the end of the macro definition
- Then whatever variation of macro variables required is specified in the statements following %mend;

i.e. %macroname(temp1);



Why are Macros Used?

- Reduce code in programs
- Simplification
- Repetitive operations/programs
- Where an operation is fundamentally the same a macro can be used such that the code only need be written once
- Extension of your SAS knowledge, not a whole new language



You may already be using Macros

- Some very simple macros you may have already used without realising:

```
%let var = x;
```

By using &var in following code you call the character x in its place – universal changes to var can also be made with ease

```
%include "filename";
```

Allows you to include text from the source “filename”



Object Oriented Programming

■ Procedure in a Macro:

```
%macro freq1(var);  
    proc freq;  
        table &var;  
    run;  
%mend freq1;  
  
data one;  
    set sashelp.class;  
    if sex='M';  
run;  
  
%freq1(age);  
  
data two;  
    set sashelp.class;  
    if sex='F';  
run;  
  
%freq1(height);
```



Parametrising / Data Step Interaction

■ Looped Expressions:

```
%macro quarters;
```

```
  %do i=1 %to 6;
```

```
    if monthyr in (1200&i,2200&i,3200&i) then quarter=1200&i;
```

```
    if monthyr in (4200&i,5200&i,6200&i) then quarter=2200&i;
```

```
    if monthyr in (7200&i,8200&i,9200&i) then quarter=3200&i;
```

```
    if monthyr in (10200&i,11200&i,12200&i) then quarter=4200&i;
```

```
  %end;
```

```
%mend;
```

```
%macro halfyears;
```

```
  %do i=1 %to 6;
```

```
    if monthyr in (1200&i,2200&i,3200&i,4200&i,5200&i,6200&i) then  
      halves=1200&i;
```

```
    if monthyr in (7200&i,8200&i,9200&i,10200&i,11200&i,12200&i) then  
      halves=2200&i;
```

```
  %end;
```

```
%mend;
```



Parametrising / Data Step Interaction

```
/*Q = Quarter, H = HalfYear*/  
%macro QorH(required);  
data temp&required;  
    set temp;  
        %if &required=Q %then %do;  
            %quarters;  
        %end;  
        %else %do;  
            %halfyears;  
        %end;  
run;  
%mend;  
%QorH(Q);
```

What could it look like if no macro?

```
data temp1;
  set temp;
  if monthyr in (12001,22001,32001)then quarter=12001;
  if monthyr in (42001,52001,62001)then quarter=22001;
  if monthyr in (72001,82001,92001)then quarter=32001;
  if monthyr in (102001,112001,122001)then quarter=42001;
  if monthyr in (12002,22002,32002)then quarter=12002;
  if monthyr in (42002,52002,62002)then quarter=22002;
  if monthyr in (72002,82002,92002)then quarter=32002;
  if monthyr in (102002,112002,122002)then quarter=42002;
  if monthyr in (12003,22003,32003)then quarter=12003;
  if monthyr in (42003,52003,62003)then quarter=22003;
  if monthyr in (72003,82003,92003)then quarter=32003;
  if monthyr in (102003,112003,122003)then quarter=42003;
  if monthyr in (12004,22004,32004)then quarter=12004;
  if monthyr in (42004,52004,62004)then quarter=22004;
  if monthyr in (72004,82004,92004)then quarter=32004;
  if monthyr in (102004,112004,122004)then quarter=42004;
  if monthyr in (12005,22005,32005)then quarter=12005;
  if monthyr in (42005,52005,62005)then quarter=22005;
  if monthyr in (72005,82005,92005)then quarter=32005;
  if monthyr in (102005,112005,122005)then quarter=42005;
  if monthyr in (12006,22006,32006)then quarter=12006;
  if monthyr in (42006,52006,62006)then quarter=22006;
  if monthyr in (72006,82006,92006)then quarter=32006;
  if monthyr in (102006,112006,122006)then quarter=42006;
run;
```



Some Other Macros

```
options macrogen mprint mlogic;
data _null_;
    x = today();
    y = intnx('month',x,-36);
    call symput("mydatenow",put(x,monyy5.));
    call symput("mydate3yr",put(y,monyy5.));
run;
%put &mydate3yr;
%put &mydatenow;

%macro merging(exclusion,freq1,freq2);
/*Performance sets from this month*/
    proc sort data = perf&mydatenow;
        by acctnmb;
    run;
/*Accts Opened 3 yrs Ago*/
    proc sort data = open&mydate3yr;
        by acctnum;
    run;
```



Some Other Macros Cont'd

```
/*Creating performance at 3 Years for the accts opened*/
data perfat3yrs&mydate3yr;
    merge perf&mydatenow(rename=(acctnmb=acctnum) in=a) open&mydate3yr (in = b);
    by acctnum;
    if a and b;
run;
/*Specific performance items needed*/
proc freq data = perfat3yrs&mydate3yr;
    &exclusion;
    table &freq1*&freq2/missing out = &freq1.by&freq2;
run;
%end;
%mend;
%merging(where score>0 ,score,bad);
%merging(where decision ne 'REFER' ,score,preapp);
%merging(where score>0 and decision ne 'REFER' ,bad,preapp);
```



Troubleshooting

- Use double quotes (“”) around all filenames etc involving a macro variable to be resolved i.e. “C:\excel&year”
- Use a period as a delimiter when adding text to the end of a macro variable i.e. data=&month.&year or data=&library.&month.&year
- Write full version for one variation of code then use SAS Macro to allow for all subsequent variations
- options symbolgen mprint mlogic; These options are useful to track the SAS Macro through its operations



Conclusion / Summary

- Macros significantly reduce the length of programs
- As complexity increases, macros are invaluable in terms of time saving and reducing the extent of code changes



Further Reading

- Texts:

SAS Macro Programming Made Easy – Michele M. Burlew

In the Know...SAS Tips and Techniques from Around the Globe – Phil Mason

- Internet:

Macro Functions – How to Make Them, How to Use Them – Arthur L. Carpenter

<http://www2.sas.com/proceedings/sugi27/p100-27.pdf>



Questions

- Any questions?