USING MACROS TO RUN REGRESSIONS
Reesa Laws, Center for Health Research
Kaiser Permanente, Northwest Region

ABSTRACT:
Anytime a programmer has to repeat code, macros should come to mind. Macros can cut programming time and code by half. This paper describes how macros are used to run regressions and to create intercept and prediction variables in any model a user chooses. The user needs only to select the outcome variable, the model and give the number of variables in that model. With the macro described, the programmer need not change the code when a different model of varying length is selected.

INTRODUCTION:
This paper describes a program that was created to reduce the number of regression programs previously used. The code for the regression run, the creation of intercept and prediction variables, and the subsequent regression run on the validation dataset were the same, but I needed to run this code on many different models of varying lengths. By changing the three %LET statements at the top of the program, I could select the outcome variable and the model, specifying the number of variables in that model without additional changes.

BACKGROUND:
The project task was to estimate medical care costs for an elderly population. The first regression uses half the sample to build models. The main outcome variable was total medical care expenses for the population for 12 months after the baseline data were collected. If we only had part of the year, we weighted the number of months. Then the actual values from the estimation models were plugged into the validation dataset—the other half of the sample—to see how well the actual scores fit the model developed. The code listed below is followed by output from the SAS* log using the options MPRINT and SYMBOLGEN.

PROGRAM DETAILS:
1) The dependent variables are weighted by months of eligibility in the Kaiser Permanente HMO.
   WGT1=MOM_ELIG/12;
   WTOTAL=(TOTAL*WGT1);
   WTOTINP=(TOTINP/WGT1);
   WTOUT=(TOTOUT/WGT1);

2) %LET variables are created for choosing the dependent variable, the model and the number of variables in the model.
   %LET DEP=TOTAL;
   %LET MODNUM=1;
   %LET NUMVARS=10;

3) All models are created using %LET statements. The dataset is split into an estimation dataset and a validation dataset.
   %LET MODEL1=FAGE_65 FAGE_70 FAGE_75 FAGE_80
         FAGE_85 MAGE_70 MAGE_75 MAGE_80 MAGE_85
         FH388;
   %LET MODEL2=FAGE_65 FAGE_70 FAGE_75 FAGE_80
         FAGE_85 MAGE_70 MAGE_75 MAGE_80 MAGE_85
         RFH135 FH388 B_SHSP;
   %LET MODEL3=FAGE_65 FAGE_70 FAGE_75 FAGE_80
         FAGE_85 MAGE_70 MAGE_75 MAGE_80 MAGE_85
         SC_FFPRO PHYSICL SYMPTOM EMOTION SOCIAL
         RCESD20;
   %LET MODEL4=FAGE_65 FAGE_70 FAGE_75 FAGE_80
         FAGE_85 MAGE_70 MAGE_75 MAGE_80 MAGE_85
         FH388 EMOTION PHYSICL SOCIAL SYMPTOM B_SHSP
         RFH135;
   %LET MODEL5=FAGE_65 FAGE_70 FAGE_75 FAGE_80
         FAGE_85 MAGE_70 MAGE_75 MAGE_80 MAGE_85
         EMOTION PHYSICL SOCIAL SYMPTOM;
   IF SAMPLE1=1 THEN OUTPUT S1;
   ELSE IF SAMPLE1=0 THEN OUTPUT S0;

4) PROC REG is run on the estimation dataset, outputting the parameter estimates. Macro variables are used to substitute the appropriate dependent variable and model into the TITLE and MODEL statements.
   PROC REG OUTEST=EST DATA=S1;
   TITLE "W&DEP BY MODEl&MODNUM";
   MODEL W&DEP:&&MODEl&MODNUM;
   WEIGHT WGT1;
   LOG:
   proc reg outest=est data=s1;
   symbolgen: Macro variable DEP resolves to TOTAL
   symbolgen: Macro variable MODNUM resolves to 1
   symbolgen: Macro variable DEP resolves to TOTAL
   symbolgen: Macro variable MODEll resolves to FAGE 65 FAGE 70 FAGE 75 FAGE 80
   symbolgen: & resolves to &.
   symbolgen: Macro variable MODNUM resolves to 1
   symbolgen: Macro variable MODEL1 resolves to FAGE 65 FAGE 70 FAGE 75 FAGE 80
   symbolgen: macro resolved to .
   symbolgen: Macro variable MODNUM resolves to 1

5) The output dataset created from the PROC REG is used to create intercept variables. Macro INT_VARS is used to scan through each variable in the model, creating a new "X" variable from each variable. The macro then scans through each variable, keeping the intercept and the new "X" variables.
DATA EST; SET EST;

%MACRO INT_VARS;
&DVARS=1 %TO &NUMVARS;
%LET VAR&I=%SCAN(&&MODEL&MODNUM,&I);
X&&VAR&I = &&VAR&I;
%END;
KEEP INTERCEP &DVARS=1 %TO &NUMVARS;
%LET VAR&I=%SCAN(&&MODEL&MODNUM,&I);
X&&VAR&I
%END;
%MEND INT_VARS;

%MEND INT_VARS;

LOG:

SYM80LGEN: Macro variable NUMVARS resolves to 10
SYM80LGEN: & & resolves to 6.
SYM80LGEN: Macro variable MODNUM resolves to 1
SYM80LGEN: Macro variable MODEL1 resolves to FAGE_65 FAGE_70 FAGE_75 FAGE_80
FAGE_85 MAGE_70 MAGE_75
MAGE_80 MAGE_85 FH388
SYM80LGEN: Macro variable I resolves to 1
SYM80LGEN: & & resolves to 6.
SYM80LGEN: Macro variable I resolves to 1
SYM80LGEN: Macro variable VAR1 resolves to FAGE_65
SYM80LGEN: & & resolves to 6.
SYM80LGEN: Macro variable I resolves to 1
SYM80LGEN: Macro variable VAR1 resolves to FAGE_65
SYM80LGEN: Macro variable MODEL1 resolves to FAGE_65 FAGE_70 FAGE_75 FAGE_80
FAGE_85 MAGE_70 MAGE_75
MAGE_80 MAGE_85 FH388
SYM80LGEN: Macro variable I resolves to 2
SYM80LGEN: & & resolves to 6.
SYM80LGEN: Macro variable I resolves to 2
SYM80LGEN: Macro variable VAR2 resolves to FAGE_70
SYM80LGEN: & & resolves to 6.
SYM80LGEN: Macro variable I resolves to 2
SYM80LGEN: Macro variable VAR2 resolves to FAGE_70
((this continues through 10 resolutions))

MPRINT(INT_VARS): XFAGE_65 = FAGE_65;
MPRINT(INT_VARS): XFAGE_70 = FAGE_70;
MPRINT(INT_VARS): XFAGE_75 = FAGE_75;
MPRINT(INT_VARS): XFAGE_80 = FAGE_80;
MPRINT(INT_VARS): XFAGE_85 = FAGE_85;
MPRINT(INT_VARS): XMAGE_70 = MAGE_70;
MPRINT(INT_VARS): XMAGE_75 = MAGE_75;
MPRINT(INT_VARS): XMAGE_80 = MAGE_80;
MPRINT(INT_VARS): XMAGE_85 = MAGE_85;
MPRINT(INT_VARS): XFH388 = FH388;
MPRINT(INT_VARS): KEEP INTERCEP XFAGE_65
XFAGE_70 XFAGE_75 XFAGE_80
XFAGE_85 XMAGE_70 XMAGE_75
XMAGE_80 XMAGE_85 XFH388;

6) This new estimation dataset is appended to the validation dataset and prediction variables are created. The macro PRED is used to once again scan through the variables in the model, multiplying the "X" variables by the original variables and adding these up, creating the prediction variable.

CONCLUSION:

Macros offer a highly efficient way of writing code that repeats itself. Originally these regressions were run by creating different programs to run each model a process that was time-consuming and inefficient. With macros, the process requires only one program, making the process easier to follow and much less time intensive.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.