CBCS
http://agrobiol.sggw.waw.pl/cbcs

Communications in Biometry and Crop Science
Vol. 4, No. 2, 2009, pp. 40–41

International Journal of the Faculty of Agriculture and Biology,
Warsaw University of Life Sciences, Poland

SOFTWARE TRICKS AND TIPS

SAS Macro Variables and ARRAY Processing

My previous contribution on SAS macro variables (van Santen, 2009) introduced the basic aspects of this time saving tool of the SAS/BASE installation. Macro variables can be deployed whenever there is a need to repeatedly access a certain string in a given program or project. The current contribution gives two examples where macro variables might be useful.

Example 1: Have multiple response variables that need to be accessed repeatedly

```
OPTIONS symbolgen;
/* Symbolgen option writes the resolution of the macro variable to the log window*/
 %LET RESPONSE_VAR = AVAR BAVR CVAR DVAR EVAR FVAR;
/*The maximum length of the string is 64000 characters*/
PROC CORR;
      VAR &RESPONSE_VAR; *no quotation marks needed for macro variable;
RUN;
PROC GLM;
       CLASS A B;
      MODEL &RESPONSE_VAR = A B A*B; *no quotation marks needed;
RUN;
```

Example 2: Linearizing a dataset

One of the strength of SAS is BY processing, where the analysis is conducted for each group of data separately. We can use this to our advantage when the analysis requires procedures that can only handle one response variable at a time, e.g. PROC MIXED or PROC GLIMMIX. If we transform the dataset to the linearized form such that the response variables are stacked and each group is identified by the original variable name. In SAS parlance this is called a long data set in contrast to the original WIDE dataset. This can be achieved through array processing in the data step.

```
OPTIONS symbolgen;
%LET ARRAY_COLUMNS = AVAR BAVR CVAR DVAR EVAR FVAR;
/* Column headers could be copied and pasted directly from an external data file*/
/* This would likely avoid transcription errors*/
DATA Linear; SET All;
ARRAY raw(*) &ARRAY_COLUMNS; *assumes array variables are of the same type;
              DO resp_n=1 TO DIM(raw); *DIM calculates array dimension;
                     response=raw(resp_n); *captures original value;
                     resp_name=VNAME(raw(resp_n));*captures the variable name;
                     OUTPUT;*critical as it writes the values to the new dataset;
              END;
DROP &array_columns;
RUN;
PROC PRINT; RUN;
```

The first eight lines of the resulting LONG dataset would be as follows:

1 CLASS VARIABLES AVAR First_Value_for_AVAR
2 CLASS VARIABLES BVAR First_Value_for_BVAR
3 CLASS VARIABLES CVAR First_Value_for_CVAR
4 CLASS VARIABLES DVAR First_Value_for_DVAR
5 CLASS VARIABLES EVAR First_Value_for_EVAR
6 CLASS VARIABLES FVAR First_Value_for_FVAR
7 CLASS VARIABLES AVAR Second_Value_for_AVAR,

where class variables refers to the columns in the original dataset that identify an individual observation (row), AVAR-FVAR are the names of response variables in the original dataset, and First_Value are the corresponding values in the first line of data.

The new dataset is then sorted by resp_name and any procedure can now be executed by group using resp_name as the BY variable. All datasets created by a given procedure will contain this BY variable and thus identify output groups. Array processing can also be used to reverse the process from LONG to WIDE as might be needed for publication or data summary http://www.biostat.jhsph.edu/bstcourse/bio632/SummerInst/Class5/arrays.pdf verified 6 July 2009).

REFERENCES

van Santen E. (2009). SAS Macro Variables. *Communications in Biometry and Crop Science* 4 (1), 1-2.

contributed by Edzard van Santen

Forage Breeding and Genetics, Dept. of Agronomy and Soils,
Auburn University, AL 36849-5412.
E-mail: vanedza@auburn.edu

Published online: 10 August 2009