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

Communications in Biometry and Crop Science
Vol. 5, No. 1, 2010, pp. 1–3

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

SOFTWARE TRICKS AND TIPS

Data checking with SAS PROC TABULATE

My previous contributions on SAS macro variables (van Santen, 2009a,b) introduced the basic aspects of this time saving tool and how it might be employed in data analysis. This contribution builds on the previous two but now the emphasis is on initial data checking.

Statistically sound data analysis rests on the assumption that the data being analyzed have been entered correctly and a particular data point has been accurately assigned to the correct treatment factor-design factor combination. As Mead et al. (1993) pointed out rather succinctly: “One, even more basic assumption that is often forgotten is that the data are, in some sense, correct.” From my personal experience as a consulting statistician I would have to agree that this assumption is more often than not violated.

Our scenario is a blocked 2-factor balanced experiment conducted in several environments, where six response variables were recorded. The data were imported from a spreadsheet and linearized as suggested by van Santen (2009b). The first task at hand is to investigate that the experiment is indeed balanced and check if the range of observed values conforms to expectations. PROC TABULATE is an ideal tool for that task.

PROC TABULATE is a simply procedure that is quickly learned. In its simplest form it consists of the PROC call, a CLASS statement followed by treatment and design factors, a VAR statement followed by the list of response variables, and a TABLE statement that constructs the output table. Because we linearized the data set the variable RESP_NAME, which contains the names of the column headers for the response variables, is listed as a class variable and the VAR statement is simply followed by response, which is the variable name we assigned to all observations during linearization. Again it is clear that linearization of the data set has its advantages in that we don’t have to remember what names we exactly assigned to the response variables.

Example 1: Checking number of observations
```
OPTIONS Symbolgen Nocenter PS=5000 LS=100 Formdlim=”-“;
/* Symbolgen option writes the resolution of the macro variable to the log window
   Nocenter left aligns the output
   PS indicates the number of lines before a hard page return is issued
   LS indicates the number of columns before a linefeed is issued
   Formdlim replaces a hard page return with a line of symbols, in this case dashes*/;
%LET ARRAY_COLUMNS = AVAR BAVR CVAR DVAR EVAR FVAR;
       * a reminder of the response variables, see van Santen(2009b);
 PROC TABULATE NOSEPS FORMCHAR=’           ‘;
       * The options in the PROC call suppress horizontal and vertical lines;
   CLASS RESP_NAME ENV BLOCK FACTOR_A FACTOR_B;
   VAR RESPONSE;
   TABLE n; *Table 1;
   TABLE response*n; *Table 2;
   TABLE RESP_NAME*response*n*f=6.0; *Table 3;
   TABLE FACTOR_B, RESP_NAME*response*n*f=6.0/rts=10;*Table 4;
   TABLE FACTOR_A, FACTOR_B, RESP_NAME*response*n/rts=10; *Table 5;
   TABLE ENV, FACTOR_A*FACTOR_B, RESP_NAME*response*n*f=6.0/rts=20; *Table 6;
 RUN;
```

If the drive letter changes, a simple adjustment in the definition for macro variable 'path' will take care of both the library assignment as well as the spreadsheet location. Macro variables have multiple uses, some of which I will describe in the next issue.

A TABLE has three dimensions separated by commas: "Page, Row, Column", but all dimensions may not be present. The number of commas in the table definition determines the dimensions. It is easiest to understand the TABLE command when considered from right to left. The example below creates a series of tables from simple to involved.

1. The first table will display single cell with the total number of observations in the data set. The statistic "n" is the only one that does not require a variable listed under VAR.
2. The second table is equivalent to the first. The "*" indicates that two items are crossed, i.e., all pairwise combinations are created. Again, only a single number is displayed. Notice that sample numbers in the first two tables are listed with two decimal places, which is the default.
3. In the 3rd table we crossed the class variable RESP_NAME with response leading to a table with a single row and six columns, one for each original response variable. We also added a formatting statement (*f = 6.0) to restrict each variable to six spaces and no decimals.
4. For the 4th table command we added FACTOR_B plus a comma leading to a two-dimensional table where levels of FACTOR_B are listed in rows. We also added the option "rts = 10" to restrict the display of FACTOR_B to 10 spaces. Without that option SAS would allocate 25% of LS given under OPTIONS (25 spaces in our example) to the row header, i.e., FACTOR_B.
5. For the 5th table command we added FACTOR_A plus a comma leading to a three-dimensional table (Pages, Rows, Columns), where each level of FACTOR_A is listed on a separate page, levels of FACTOR_B are listed in rows, RESP_NAME in columns.
6. The 6th table command assigns each FACTOR_A* FACTOR_B combination to a separate row. The factor ENV is listed preceding the 2nd comma from the right and will thus create a separate page for each environment.

Checking data balance is easy. Let's assume we have the following levels for each factor: BLOCK (4), ENV (2), FACTOR_A (2), FACTOR_B (4) and six response variables (AVAR - FVAR). Based on the experimental setup we can predict the number of observations listed in each cell of a given table:

1.	Table 1: 4*2*2*4*6 =	384
2.	Table 2: 4*2*2*4*6 =	384
3.	Table 3: 4*2*2*4 =	64
4.	Table 4: 4*2*2 =	16
5.	Table 5: 4*2 =	8
6.	Table 6: 4 =	4

If these numbers do not conform to the expectation then there is a problem with the data. The numbers in the tables will offer a clue, e.g. if there is a cell in Table 6 with n=5 there might be another cell with n=3, i.e., a treatment combination has not been assigned properly. This often happens when researchers do not use a template for entering data.

The next step in the initial analysis of data integrity might be to check whether the data make sense. This is where the expertise of the researcher plays an important role. If the response variable for example is dry matter content then the values should be between 0 and 1 if expressed as a fraction, or 0 and 100 if expressed as percent. This is easily accomplished because PROC TABULATE can calculate many statistics such as N, NMISS, MINimum, MAXimum, RANGE, MEAN, MEDIAN, various percentiles, etc. (http://support.sas.com/onlinedoc/913/docMainpage.jsp.; verified 20 March 2010). We can thus construct tables to assist us in data checking.

Example 2: Checking data ranges

```
PROC TABULATE;
   CLASS RESP_NAME ENV BLOCK FACTOR_A FACTOR_B;
   VAR RESPONSE;
   TABLE RESP_NAME, response*(MIN MAX)*f=8.0/rts=12; *Table 1;
   TABLE ENV*RESP_NAME, response*(MIN MAX)*f=8.1/rts=20; *Table 2;
   TABLE RESP_NAME*ENV*FACTOR_A*FACTOR_B, response*(MIN MAX)* f=8.2/rts=50; *Table 3;
RUN;
```

1. Table 1: A table with six rows, one for each response variable, and two columns (Minimum, Maximum) is created.
2. Table 2: ENV (2 levels) is crossed with RESP_NAME hence the table has 10 rows and two columns.
3. Table 3: RESP_NAME is crossed with ENV (2), FACTOR_A (2), and FACTOR_B (4) to create rows. The table now has 6*2*2*4 = 96 rows and is now ordered by response variable name for easier checking.

One could of course, use "WHERE" and/or "IF" clauses could be used as well to concentrate on response variables or factors that need further checking. The utility of PROC TABULATE is only limited by the imagination of the user. Lastly, the entire output may be transferred to EXCEL by wrapping an HTML statement around the PROC.

Example 3: Transferring Tables to EXCEL

```
ODS HTML File = "DRIVE:/FOLDER/Table.xls"; * if the approach listed in van Santen (2008) is
used, drive and folder information may be omitted;
PROC TABULATE;
   CLASS;
   VAR;
   TABLE;
RUN;
ODS HTML CLOSE;
QUIT;
```

This file can then be opened in EXCEL. This is also a very quick method to summarize data from an experiment. Finally, checking basic data integrity is only the first step in the process of ascertaining that the data are indeed what they are supposed to be.

REFERENCES

Mead, R., R.N. Curnow, and A.M. Hasted. 1993. *Statistical methods in agriculture and experimental biology*. 2nd ed. Chapman & Hall, New York.

van Santen, E. (2008). Make a project folder home base for SAS. *Communications in Biometry and Crop Science Crop Science* 3 (1), 1–2.

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

van Santen, E. (2009b). SAS Macro Variables and ARRAY Processing. *Communications in Biometry and Crop Science Crop Science* 4 (2), 40–41.

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: 31 March 2010