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

Communications in Biometry and Crop Science
Vol. 5, No. 2, 2010, pp. 66–68

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

SOFTWARE TRICKS AND TIPS

PROC SQL vs. DATA Step or Anything you can do, I can do better

The second part of the title is a song from Irving Berlin's 1946 Broadway musical "Annie get you gun", where the two protagonists (male and female) try to outdo each other in performing more and more complicated tasks. Yet both fail in a basic task as neither can bake a cake. The relationship between PROC SQL (**S**tructured **Q**uery **L**anguage) and the DATA step in SAS® is much the same. PROC SQL (SAS, 2010) is very useful in applied data analysis situations to reorder data columns, calculate derived variables and add them to the dataset, and merge information from various datasets. Some tasks are accomplished much more straightforwardly with PROC SQL than the DATA step. This contribution is concerned only with creating tables (data sets), a small subset of the capabilities of PROC SQL.

Let's assume that the task at hand is to read the data from a cultivar trial into a SAS dataset and then prepare it for analysis. We have three pieces of information (1) the actual **RAW DATA** indexed by plot number, (2) the **RANDOMIZATION** table that links data and treatments (entries), and (3) the **TRIAL INFO** that provides information about cutting date and data conversion (see supplementary EXCEL file: http://agrobiol.sggw.waw.pl/~cbcs/articles/5_2_1/). I will create SQL tables from these datasets assuming that we have a much more data.

I am invoking the standard SAS programming window color scheme, where key words are highlighted in blue, user input is black, and comments appear green. The basic sequence of commands to create a table in SQL is **CREATE**, **SELECT**, **FROM**, and **WHERE** in that order. **CREATE** is used to name the new data table, **SELECT** governs what columns (= variables in the data step) are selected from the underlying dataset, **FROM** names the underlying data set, and **WHERE** indicates the conditions by which rows (= observations in the data step) are selected and may be omitted if all observations are to be selected. PROC SQL statements are executed immediately and do not require a **RUN** statement. PROC SQL is terminated when a data step or another PROC is encountered or explicitly by the **QUIT** statement. For good programming practice and clarity an explicit termination is preferred.

STEP 1: A simple SQL to extract data from a larger dataset

```
PROC SQL;
        CREATE TABLE RANDOMIZATION AS
                SELECT *
                FROM TEST.RANDOMIZATION
                WHERE LOC_N= 1;
        CREATE TABLE RAW_DATA AS
                SELECT *
                FROM TEST.RAW_DATA
                WHERE LOC_N= 1;
        CREATE TABLE TRIAL_INFO AS
                SELECT *
                FROM TEST.TRIAL_INFO
                WHERE LOC= "TVS_2010";

QUIT;
```

In our case it would not have been necessary to create this tables because they already existed as SAS datasets (see supplementary SAS program). Now that we have created the needed base tables we want to merge the information into a single table so we can do the necessary calculations.

In the following example I have used the **LEFT JOIN** command, a special case of an outer join, where the first table has rows not present in the second table. It functions much like the (in=) option following the first named dataset when merging datasets during the data step. The **ON** clause serves the same function as the **BY** statement during merging. Key variables need to be explicitly referenced as to the source table because they are present in both underlying tables.

STEP 2: Merging tables

```
PROC SQL;
        CREATE TABLE RANDplusDATA AS
                SELECT RANDOMIZATION.*, CUT, SDW, SGW, PGW
                FROM RANDOMIZATION LEFT JOIN RAW_DATA
                ON RANDOMIZATION.PLOT = RAW_DATA.PLOT;
        CREATE TABLE RANDplusDATAplusTRIAL AS
                SELECT RANDplusDATA. Loc_N, LOC, RANDplusDATA.CUT, PLOT,
                        Entry_N, Entry, ROW, COL, SDW, SGW, PGW, P_DATE,
                        H_DATE, CF, CF_UNITS
                FROM RANDplusDATA LEFT JOIN TRIAL_INFO
                ON RANDplusDATA.LOC_N = TRIAL_INFO.LOC_N;
QUIT;
```

The final step involves expressing forage dry matter yield in lbs per ha (for extension purposes) and in kg per ha for a manuscript. We furthermore wish to express the yield in each cut and block relative to the base population. We will create the coding for the BLOCK variable as well as do the necessary transformation of the raw data. Along the way we will discard all information not needed for analysis.

The table CALCULATED contains all the columns of interest from the underlying table plus calculated values for forage yield in lbs per acre and kg per ha. In table TOTAL we calculated the total seasonal plot yield. The argument **UNIQUE** or **DISTINCT** eliminates duplicates and functions much like the **NODUPKEY** in **PROC SORT**. Notice that we coded the total yield as cut=9, thereby enabling analysis of each cut as well as total in a single run. Creating the table CUTSplusTOTAL introduces the **UNION** command, the **PROC SQL** equivalent of the **SET** command for concatenating tables in the datastep. In the table CONTROL we have extracted the mean of controls (check entries) by block and cut. These will be used for the final table, where we calculate relative yields. Creation of the final table READYforANALYSIS introduces **NATURAL JOIN** as an easy way to join tables that have identical numerical key variables. In this particular case, no **ON** clause is needed.

STEP 3: Data transformation

```
PROC SQL;
        CREATE TABLE CALCULATED AS
                SELECT Loc_N, LOC, CUT, PLOT, INT(PLOT/100) AS BLOCK, Entry_N,
                        Entry, ROW, COL, (SDW/SGW)*PGW*CF*CF_units AS lbs_acre,
                        (SDW/SGW)*PGW*CF*CF_units*1.12 AS kg_ha
                FROM RANDplusDATAplusTRIAL;
        CREATE TABLE TOTAL AS
                SELECT UNIQUE Loc_N, LOC, 9 as CUT, PLOT, BLOCK, Entry_N,
                        Entry, ROW, COL, SUM(lbs_acre) AS lbs_acre,
                        SUM(kg_ha) AS kg_ha
                FROM CALCULATED
                GROUP BY LOC_n, plot;
        CREATE TABLE CUTSplusTOTAL AS
                SELECT * FROM CALCULATED
                UNION
                SELECT * FROM TOTAL;
        CREATE TABLE CONTROL AS
                SELECT UNIQUE LOC_N,CUT, BLOCK, mean(kg_ha) as CONTROL
                FROM CUTSplusTOTAL
                WHERE SUBSTR(ENTRY,1,2)="C0"
                GROUP BY LOC_N, CUT, BLOCK;
        CREATE TABLE READYforANALYSIS AS
                SELECT CUTSplusTOTAL.*, 100*kg_ha/CONTROL AS REL_YIELD
                FROM CUTSplusTOTAL NATURAL JOIN CONTROL
                ORDER BY Loc_N, CUT, PLOT;
QUIT;
```

I should point out that all tables could have been created within a single PROC call. Furthermore, the creation of tables could have also been streamlined using fewer intermediate tables but using extra tables makes the process user-friendlier for the inexperienced SQL user. I have used PROC SQL extensively since 2008 and have come to value it when I have to reshape data sets. Along with my previous tips and tricks articles (van Santen 2008, van Santen 2009 a, b, 2010) this PROC has really enabled me to become more efficient when serving the data analysis needs of my faculty colleagues and our graduate students.

References

SAS (2010). Introduction to the SQL Procedure (http://support.sas.com/documentation/cdl/en/sqlproc/62086/HTML/default/a002536894.htm; verified 20. July, 2010).

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.

van Santen, E. (2010). Data checking with SAS PROC TABULATE. *Communications in Biometry and Crop Science Crop Science* 5 (1), 1–3.

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: 28 July 2010